Actas
do
10º Encontro Português de Computação Gráfica

1 – 3 de Outubro 2001
Lisboa – Portugal

Patrocinadores de Honra

Patrocinadores

Organização

PREFÁCIO

A investigação, o desenvolvimento e o ensino na área da Computação Gráfica constituem, em Portugal, uma realidade positiva e de largas tradições. O Encontro Português de Computação Gráfica (EPCG), realizado no âmbito das actividades do Grupo Português de Computação Gráfica (GPCG), tem permitido reunir regularmente, desde o 1º EPCG realizado também em Lisboa, mas no já longínquo mês de Julho de 1988, todos os que trabalham nesta área abrangente e com inúmeras aplicações.

Pela primeira vez no historial destes Encontros, o 10º EPCG foi organizado em ligação estreita com as comunidades do Processamento de Imagem e da Visão por Computador, através da Associação Portuguesa de Reconhecimento de Padrões (APRP), salientando-se, assim, a acrescida colaboração, e a convergência, entre essas duas áreas e a Computação Gráfica.

Tal como nos Encontros anteriores, o programa está estruturado ao longo de três dias, sendo desta vez o primeiro dia dedicado a seminários por conferencistas convidados e os dois últimos à apresentação de comunicações e de *"posters"*, decorrendo em simultâneo o Concurso para Jovens Investigadores, uma Exibição Comercial e, pela primeira vez, um Atelier Digital. Como novidade essencialmente dedicada aos jovens, realiza-se ainda em paralelo com o Encontro um torneio de jogos de computador.

Em resposta ao apelo às comunicações para este 10º EPCG foram submetidos 38 trabalhos, na sua maioria de grande qualidade, tendo sido seleccionadas pela Comissão de Programa, após um cuidadoso processo de avaliação, apenas 19 comunicações; aos autores de 14 dos restantes trabalhos, considerados suficientemente promissores, foi sugerida a sua reformulação e uma nova submissão como *"posters"*. Posteriormente foram submetidas 10 comunicações curtas (*"posters"*) das quais seleccionámos oito.

As comunicações seleccionadas, que se incluem nestas Actas, aparecem agrupadas em seis sessões técnicas: Síntese de Imagem, Interacção, Modelação e Visualização, Processamento de Imagem, Animação e Multimédia, e Ambientes Virtuais. Tal como nos eventos anteriores, estamos convictos que foi possível, mais uma vez, estabelecer um programa de elevado interesse e qualidade técnico-científica, motivador para todos os participantes no 10º EPCG.

A realização de um evento como o Encontro Português de Computação Gráfica só é possível com o trabalho árduo e a colaboração empenhada de um grande grupo de voluntários e várias instituições, sendo devida uma palavra de sincero agradecimento a todos os que para ele contribuíram.

Os Presidentes da Comissão Organizadora e da Comissão de Programa gostariam de agradecer:

Aos membros da Comissão Organizadora, em particular à Fátima Estevens, o empenho e o entusiasmo postos na concretização das inúmeras tarefas que a organização deste evento implica;
Aos membros da Comissão de Programa e aos revisores externos, a motivação e o cuidado que dedicaram à avaliação e selecção dos trabalhos submetidos;
Aos conferencistas convidados, Augusto Albuquerque, Fausto Bernardini, David Breen, Ken Musgrave, Jivka Ovtacharova e Demetri Terzopoulos, a disponibilidade para estarem presentes e partilharem com os participantes o seu saber e as suas experiências;
Aos autores de todas as submissões, a motivação, o empenho e o esforço que puseram na escrita dos seus trabalhos.

Finalmente agradecemos a todas as instituições que apoiaram a realização do 10º Encontro Português de Computação Gráfica, especialmente à Hewlet Packard, particularmente à Rita Cordovil e ao Luis Monteiro, à Fundação para a Ciência e Tecnologia, à Fundação Luso-Americana para o Desenvolvimento, ao Ministério da Educação, ao Instituto Superior de Ciências do Trabalho e da Empresa, à IBM, em particular à Filipa Valente, à Next Vision, napessoa do Carlos Leitão, à Silicon Graphics, à Micrograf, à Autodesk, ao INSAT, à UniWeb, ao Instituto de Comércio Externo e, ainda, à ADETTI, ao DCTI-ISCTE e ao INESC, co-organizadores com a APRP e o GPCG deste evento.

Lisboa, Outubro de 2001

Joaquim Madeira
Jorge Salvador Marques
Miguel Salles Dias
Joaquim A. Jorge

Comissão Organizadora

José Miguel Salles Dias (ISCTE/ADETTI), *Presidente*
Alcina Prata (Instituto Politécnico de Setúbal)
Bento Correia (INETI)
Fátima Estevens (ADETTI)
Fátima Silva (DCTI-ISCTE)
João Pereira (IST/INESC)
Manuel Gamito (ADETTI)
Manuel Sequeira (ISCTE/ADETTI)
Pedro Faria Lopes (ISCTE/ADETTI)

Comissão do Programa

Joaquim Madeira (UC), Co-Presidente
Jorge Salvador Marques (IST/ISR), Co-Presidente
A. Augusto de Sousa (FEUP/INESC)
Adriano Martins Lopes (UC)
Adérito Fernandes Marcos (UM/CCG)
Armando Pinho (UA/IEETA)
Aurélio Campilho (FEUP)
Beatriz Sousa Santos (UA/IEETA)
Fernando Nunes Ferreira (FEUP/INESC)
Fernando Muge (IST)
Francisco Feito (Univ. Jáen, Espanha)
Fernando Pereira (IST/IT)
Hélder Araújo (UC/ISR)
Isabel Ribeiro (IST/ISR)
João Bernardo (IST)
João Brisson Lopes (IST)
João Duarte Cunha (LNEC)
João Paulo Costeira (IST)
João Pereira (IST/INESC)
Joaquim Jorge (IST/INESC)
José Bulas Cruz (UTAD)
José Carlos Teixeira (UC)
J. Delgado Domingos (IST)
José Dionísio (IST/IDMEC)
José Manuel Rebordão (FCL)
José Miguel Salles Dias (ISCTE/ADETTI)
José Santos Victor (IST/ISR)
Jorge Padilha (FEUP)
Juan Carlos Torres (Univ. Granada, Espanha)
Manuel Gamito (ADETTI)
Manuel Próspero dos Santos (UNL)
Mário Rui Gomes (IST/INESC)
Mário Martins (UM)
Nuno Correia (UNL)
Pedro Faria Lopes (ISCTE/ADETTI)
Pere Brunet (Univ. Pol. Catalunya, Espanha)
Rafael Bidarra (Univ. Téc. Delft, Holanda)
Rogério Caldas Pinto (IST)
Vasco Branco (UA/INESC)
Xavier Pueyo (Univ. Girona, Espanha)

Recensores Externos

Michal Koutek - Univ. Téc. Delft, Holanda
Hiroshi Masuda - Univ. Tóquio, Japão
Wu Shin-Ting - UNICAMP, Brasil

Lista de Artigos

Lista de Autores

Programa

Sessão 1

Síntese de Imagem

Sessão 2

Interacção

Sessão 3

Modelação e Visualização

Sessão 4

Processamento de Imagem

Sessão 5

Animação e Multimédia

Sessão 6

Ambientes Virtuais

Sessão 1

SÍNTESE DE IMAGEM

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Holodeck: uma técnica para Armazenamento e Busca Eficientes de Raios Luminosos em Síntese de Imagem

Ana Cláudia Nogueira
ISMAI/INESC Porto
Porto
anogueira@ismai.pt

A. Augusto Sousa
FEUP/INESC Porto
Porto
aas@fe.up.pt

António Cardoso Costa
ISEP/INESC Porto
Porto
acc@dei.isep.ipp.pt

Sumário

Os métodos de síntese de imagem baseados em modelos físicos sofisticados, com preocupações de produzir resultados fisicamente válidos, tendem a consumir recursos de processamento muito pesados, em particular, tempos de processamento incompatíveis com visualizações interactivas. Neste artigo apresenta-se uma implementação da técnica Holodeck que, com base num motor de radiância fisicamente válido, cria uma estrutura que permite armazenar a radiância dos raios que atravessam a cena para posterior reutilização. Desta forma, sempre que se procede ao cálculo de um raio luminoso, consulta-se a estrutura a fim de se obter rapidamente uma resposta adequada e procede-se ao uso do motor de radiância apenas quando não existir resposta satisfatória. Os resultados obtidos, embora preliminares, permitem prever que, através desta técnica, poderão ser realizadas visualizações interactivas em cenas geometricamente complexas, com níveis elevados de realismo, fundamentais em certos tipos de aplicação (arquitectura, design de iluminação, etc).

Palavras-chave

Traçagem de raios, síntese de imagem, visualização interactiva, cálculo/armazenamento/pesquisa de radiância.

1. INTRODUÇÃO

Quando se observa uma imagem criada através de um processo de síntese de imagem baseado no método de traçagem de raios (*Ray-Tracing* [Whitted80]), na realidade apreende-se um conjunto de pontos coloridos, cada um deles localizado numa grelha rectangular. Para a obtenção dessas tonalidades é necessária uma grande quantidade de cálculos: em cada um desses pontos são projectados raios que partem do ponto de observação (câmara) em direcção à cena; cada um desses raios pode atingir vários objectos da cena em diversos pontos; a partir do ponto mais próximo obtido são gerados novos raios que podem atingir outros objectos, dando assim origem a novos raios, etc. Deste modo, quando o número de objectos é elevado, a geração de novos raios pode atingir níveis muito elevados. No fim deste processo, tipicamente, o tempo empregue para obter o resultado final é muito elevado.

Este problema torna-se mais grave quando se trata de criar dinamicamente imagens, por exemplo, durante uma navegação na cena (*walkthrough*), dado que os tempos de processamento, demasiado altos, são incompatíveis com a cadência necessária para produção de imagens com fins de visualização.

Em [Larson98] apresenta-se uma nova solução: pré-processar e armazenar alguma informação relacionada com os raios luminosos, a fim de ser utilizada mais tarde. Assumindo-se que o observador se pode deslocar livremente na cena, tem de se procurar garantir que a informação armazenada alberga raios representativos de todas as direcções de vista possíveis.

Este tipo de abordagem não tem sido aprofundado, apesar das suas potencialidades parecerem ser enormes. Por exemplo, depois da informação dos raios estar armazenada, esta pode ser transportada e fazer-se a reconstituição da cena (visualização) noutro local. Além disso, reduz-se ou até se elimina o processo de geração de novos raios, pelo que o tempo de processamento poderá tornar-se significativamente inferior.

Neste documento referem-se alguns trabalhos realizados na área da síntese de imagem e aqueles que mais se relacionam com o tema de visualização em *walkthroughs*. Em seguida, faz-se a descrição da aplicação protótipo implementada, baseada na técnica *Holodeck* apresentada em [Larson98], dando-se especial ênfase à sua estrutura de raios, a qual é criada para utilização pelo processo de visualização. O protótipo, seguindo as linhas gerais apontadas por [Larson98], permite avaliar quais os passos mais críticos do método, fornecendo indicadores importantes sobre as suas evoluções mais adequadas. Em sequência, apresenta-se uma avaliação dos resultados obtidos e, por último, resumem-se as principais conclusões sobre o trabalho realizado, discutem-se

algumas limitações actuais e apresentam-se sugestões para trabalho futuro.

2. TRABALHOS RELACIONADOS

A produção de imagens realistas é um dos objectivos da computação gráfica. Actualmente, o realismo e a interacção em tempo real têm que ser balanceados, isto é, tem que ser encontrado um equilíbrio entre ambos. Para se atingir uma cadência elevada de visualização de imagens, tem que se perder em termos de realismo. Por outro lado, quando se aumenta o realismo de imagem pretendido, perde-se em interactividade.

Recentemente, surgiram três abordagens relacionadas com a visualização interactiva de imagens produzidas pelo método de traçagem de raios. Uma abordagem actualiza a visualização à medida que os raios são calculados para uma dada vista. A visualização aproximada pode ser implementada através do desenho progressivo de triângulos cada vez mais pequenos [Painter89] ou através de triangulação de *Delaunay* ou mapas de textura [Pighin97]. A segunda abordagem faz um pré-processamento da cena para um conjunto pré-definido de vistas [Levoy96] [Gortler96], sendo outras vistas obtidas por interpolação das vistas iniciais. A terceira abordagem, designada por *Holodeck* [Larson98], combina o armazenamento e pesquisa de raios luminosos anteriormente calculados com o cálculo interactivo de novos raios.

A primeira abordagem tem como grande desvantagem o facto da informação relativa a uma vista ser perdida a partir do momento em que o observador se desloca na cena (cálculo da nova vista). A principal desvantagem da segunda abordagem é o facto de não permitir uma actualização dinâmica dos dados. A abordagem do *holodeck* combina as vantagens das duas anteriores sem apresentar as suas desvantagens, na medida em que visa atingir cadências interactivas de visualização através da reutilização de raios previamente calculados, complementada com o cálculo de novos raios.

3. O CONCEITO DE *HOLODECK*

O método de cálculo de radiância de raios luminosos em iluminação global utilizado neste trabalho é o método de traçagem de raios (*Ray-Tracing*).

Dos vários métodos de *Ray-Tracing* existentes para se realizar o cálculo da radiância dos raios luminosos, foi escolhido um dos que possui validade física mais comprovada: o software *Radiance* [Ward94], [Larson97]. Trata-se de um método lento, dado que o número de raios luminosos a processar tende a ser muito elevado e envolve cálculos complexos.

Como se pretende permitir a circulação na cena (*walkthrough*), o tempo de visualização de cada imagem da cena é crucial. Surgiu então a ideia de se armazenar a informação relacionada com os raios por forma a não ser necessário o cálculo repetido dos mesmos, sempre que tal seja solicitado na visualização da cena. Para se atingir este objectivo, [Larson98] desenvolveu uma técnica que designou por *holodeck* e que se baseia numa estrutura de informação que armazena raios de luz, a fim de serem utilizados posteriormente (secção 5).

O método assenta no pressuposto de que a radiância de um raio de luz é constante ao longo de todo o seu percurso[1] [Glassner95]. Assim, dentro de uma certa tolerância em termos de direcção de um qualquer raio, será possível aproximá-lo por um outro que seja conhecido de cálculos anteriores, independentemente do facto de este último ter diferentes pontos de origem e de destino.

A vantagem do método capitaliza-se então no facto de se poderem evitar cálculos complexos de intersecção de raios com objectos, reaproveitando, tanto quanto possível, cálculos de radiância anteriormente efectuados com raios semelhantes. Em resumo, reutiliza-se a radiância de um raio anteriormente calculado, que tenha comprovadamente uma posição espacial geometricamente próxima do raio actual. No caso de não ser encontrado qualquer raio naquelas condições, não há informação recuperável e o novo raio deverá ser processado pelo método normal, previsto no motor de radiância do traçador de raios.

A estrutura de informação do *holodeck* destina-se a armazenar raios, à medida que estes vão sendo processados. A operação fundamental sobre a estrutura consiste em, face a um raio novo em processamento, pesquisar um raio conhecido, que mais se lhe aproxime. A radiância do novo raio pode então ser aproximada pela radiância do raio encontrado na estrutura.

4. DESCRIÇÃO DA IMPLEMENTAÇÃO

Em seguida apresentam-se os processos que constituem o sistema implementado de síntese de imagem.

4.1 Diagrama de processos

Com base no trabalho *holodeck* proposto em [Larson98], o sistema implementado constitui-se de quatro processos principais (Figura 1): visualização da imagem, gestor do cálculo de raios, gestor do *holodeck* e motor de radiância.

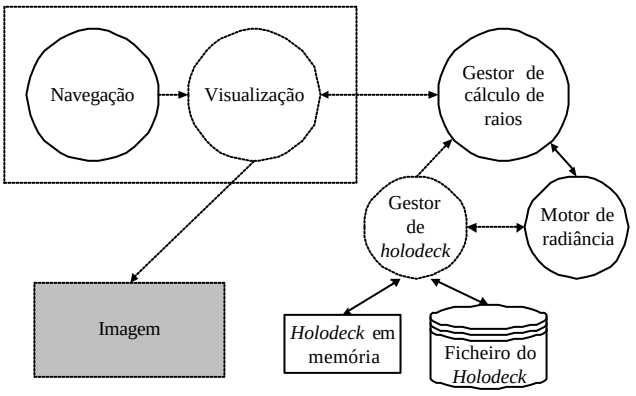


Figura 1: Diagrama de processos do sistema implementado

[1] Na ausência de meios participantes.

O processo gestor do cálculo de raios encarrega-se de coordenar as operações de chamada ao motor de radiância e as operações de consulta ao processo gestor de *holodeck*.

O processo do motor de radiância recorre à aplicação *Radiance*, a partir dos dados enviados pelo gestor do *holodeck* (pontos de partida e direcções dos raios a processar) e implementa o método de *Ray-Tracing* de forma fisicamente válida. O resultado dos cálculos efectuados é gravado em ficheiro para ser analisado pelo gestor do *holodeck*.

O gestor de *holodeck* organiza a pesquisa e/ou o armazenamento da informação dos raios calculados pelo motor de radiância, em memória ou em ficheiro físico. Sempre que seja possível encontrar uma resposta no *holodeck*, o motor de radiância não é utilizado.

As operações de visualização da imagem são comandadas por um processo de navegação, que altera os parâmetros de visualização de acordo com a actuação de dispositivos de interacção.

O processo de visualização da imagem cria uma janela virtual [Larson98] a partir da qual pode ser visualizada a cena. Para o efeito, cria-se um conjunto de raios, passando cada um deles por um *pixel* e com origem na câmara. Envia estes raios para o gestor de cálculo de raios que lhe devolve os respectivos valores de radiância para realizar a representação da imagem.

O processo de visualização é muito dependente da eficiência da pesquisa realizada sobre a estrutura do *holodeck*. A estrutura de dados definida para armazenar a informação do *holodeck* foi concebida para permitir a máxima eficácia nessa pesquisa.

A descrição pormenorizada da estrutura do *holodeck* é apresentada na secção seguinte.

5. A ESTRUTURA DE DADOS DO *HOLODECK*

O valor de radiância de um raio ao longo de uma trajectória desobstruída é constante [Glassner95]. Assim sendo, pode envolver-se parte da cena num volume que é atravessado pelos diversos raios necessários para o representar numa imagem. Este volume serve de suporte ao armazenamento dos valores de radiância dos raios dessa parte da cena.

Assim, o *holodeck* é a representação de um paralelipípedo imaginário, integrado na cena, através de uma estrutura de dados adequada. As faces do paralelipípedo são definidas por forma a serem perpendiculares aos eixos principais da cena, o que simplifica o cálculo de intersecções de raios. As seis faces do paralelipípedo são subdivididas em rectângulos, todos de dimensões iguais em cada face. A cada rectângulo resultante da divisão das faces do *holodeck* dá-se o nome de célula ou quadrícula. O número de divisões pode não ser igual nas três dimensões, como se observa na Figura 2.

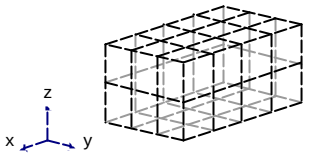


Figura 2: Vista exterior do *holodeck*

À medida que se vai analisando a cena, os raios vão sendo processados para se determinar o objecto mais próximo e avaliar a radiância correspondente. Assim, alguns raios podem intersectar o *holodeck* e, quando tal acontece, os cálculos de radiância podem ser evitados (se já existir um raio na estrutura de *holodeck* que esteja conforme as especificações discutidas nas secções seguintes), economizando-se em tempo de cálculo.

Um raio que atravesse o paralelipípedo, intersecta-o em dois pontos, sendo que cada um se situa numa das faces do *holodeck* e, em particular, numa quadrícula. Consideram-se face e quadrícula de entrada, aquelas que contêm o primeiro ponto de intersecção e face e quadrícula de saída, aquelas que contêm o segundo ponto de intersecção do raio em causa, segundo a sua direcção e sentido.

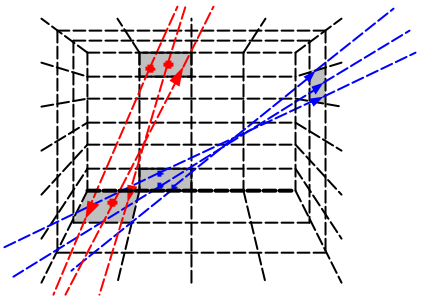


Figura 3: Vista interior do *holodeck*

O interesse da determinação das quadrículas de entrada e de saída dos raios reside no facto de todos os raios que partilham as mesmas quadrículas, de entrada e de saída, poderem ser agrupados e classificados de forma semelhante. O armazenamento destes raios "quase paralelos" é feito de forma a facilitar a sua pesquisa.

Os raios são catalogados pelo par de quadrículas, de entrada e de saída, segundo a sua direcção mas não segundo o seu sentido. Isto significa que, dois raios quase paralelos mas com sentidos contrários, que partilhem as mesmas quadrículas de entrada e de saída, são classificados da mesma forma (como exemplo veja-se, na Figura 3, que o conjunto de raios assinalados a traço grosso partilha o mesmo par de quadrículas, mas que o sentido de um deles é inverso dos restantes). Uma variável interna do *holodeck* toma então, nos dois casos, valores de sinais opostos.

Conforme descrito na secção anterior, o *holodeck* é representado por um paralelipípedo. Na aplicação implementada, cada uma das faces é identificada através de uma numeração sequencial de 1 a 6, de acordo com a

Figura 4. Nesta figura pode visualizar-se, em a) o *holodeck* visto do exterior, sendo visíveis as faces 1, 2 e 3. Em b) estão representadas apenas as faces invisíveis, o que corresponde a observar o *holodeck* a partir do seu interior.

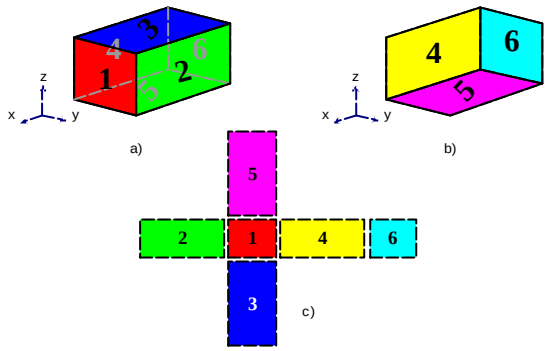


Figura 4: Identificação de cada uma das faces do *holodeck* a) Vista frontal do *holodeck*. b) Vista interior do *holodeck*. c) Planificação do *holodeck*.

Para se armazenar um raio, é necessária a determinação do par de faces que são intersectadas pelo raio. Dado que as faces são planas, não é possível que um raio entre e saia na mesma face. Na concepção da estrutura de dados adequada à representação do *holodeck*, consideram-se todas as combinações possíveis para "faces de entrada" e "faces de saída". Cada uma das seis faces pode combinar-se com as restantes cinco, pelo que existem trinta combinações possíveis de pares "face de entrada"/"face de saída". Dado que o sentido dos raios não é tido em conta, o número de combinações reduz-se para metade. Assim, a estrutura de dados do *holodeck* baseia-se num vector com quinze posições (Tabela 1).

	Faces		...
1	1	2	...
2	1	3	...
3	1	4	...
4	1	5	...
5	1	6	...
6	2	3	...
7	2	4	...
8	2	5	...
9	3	6	...
10	3	4	...
11	3	5	...
12	3	6	...
13	4	5	...
14	4	6	...
15	5	6	...

Tabela 1: Combinações de "faces de entrada"/"faces de saída" e o vector do *holodeck*.

Cada elemento do vector contém apontadores para uma estrutura que representa o par de quadrículas em que o raio intersecta as faces do *holodeck*.

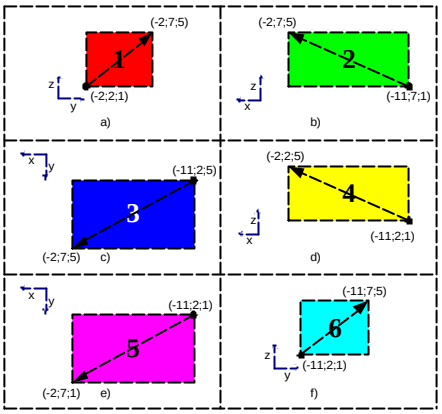


Figura 5: Projecção das faces do *holodeck* a) Face 1 b) Face 2 c) Face 3 d) Face 4 e) Face 5 f) Face 6

A conversão da representação de um paralelipípedo em três dimensões, para a representação das faces em duas dimensões, apresenta-se na Figura 5. Junto com cada face, representa-se o plano correspondente do sistema de coordenadas. Assim, observa-se que as faces 1 e 6 são paralelas ao plano *yz*, as faces 2 e 4 são paralelas ao plano *xz* e as faces 3 e 5 são paralelas ao plano *xy*.

Na Figura 6, o número de quadrículas segundo as direcções dos eixos coordenados são 6, 5 e 4, respectivamente para *x*, *y* e *z*. Cada uma das divisões é numerada sequencialmente, seguindo o sentido crescente das coordenadas dos respectivos eixos.

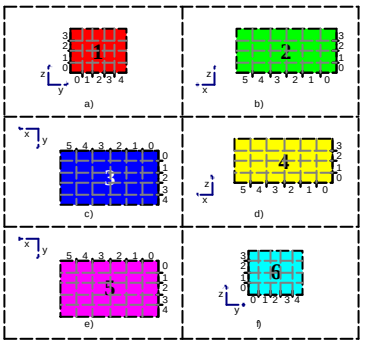


Figura 6: Projecção das faces do *holodeck* com as respectivas divisões a) Face 1 b) Face 2 c) Face 3 d) Face 4 e) Face 5 f) Face 6

Em resumo, no "sistema de duas coordenadas", considera-se que cada uma das quadrículas de uma face é identificada pelos valores representativos da coluna e da linha em que se encontram na respectiva face.

Quanto à identificação das faces obtém-se, como referido, pelo índice no vector. Dado que, numa quadrícula, podem passar vários raios, torna-se necessário caracterizar a posição destes dentro dessa quadrícula.

Assim, tendo também em atenção a informação de radiância que caracteriza o raio, a informação que é necessário armazenar por raio, na estrutura anterior é:

- Componente de radiância vermelha
- Componente de radiância verde
- Componente de radiância azul
- Distância em x ao canto superior esquerdo da quadrícula de entrada
- Distância em y ao canto superior esquerdo da quadrícula de entrada
- Distância em x ao canto superior esquerdo da quadrícula de saída
- Distância em y ao canto superior esquerdo da quadrícula de saída

5.1 Utilização do *holodeck*

A informação contida no *holodeck* é pesquisada sempre que se necessita de determinar a radiância de um raio que o atravessa. Tenta-se encontrar, na sua estrutura de dados, um raio previamente calculado que seja suficientemente semelhante, do ponto de vista geométrico do raio a calcular.

Assim sendo, o primeiro passo consiste na determinação da intersecção do raio a calcular com o volume do *holodeck*. Se a intersecção não existe, o raio pode ser enviado para processamento no motor de radiância, pois a estrutura do *holodeck* não tem, garantidamente, nenhuma informação relevante para fornecer.

Um raio é também enviado para o motor de radiância quando intersecte o *holodeck* num só ponto (caso das arestas e dos vértices) ou quando as duas quadrículas, de entrada e de saída, partilhem uma aresta do *holodeck*, (quadrículas contíguas mas de faces distintas). Desta forma evitam-se, no primeiro caso, erros de representação numérica e, no segundo, perdas de tempo de pesquisa em situações que, na maioria das vezes, não encontram raios semelhantes (dado que a grande variabilidade de direcção dos raios armazenados naquelas circunstâncias raramente torna possível a adopção de um deles como aproximação ao raio em questão).

Assim, assegura-se que, quando o raio a calcular intersecta o *holodeck*, a intersecção ocorre em dois pontos, pelo que se podem determinar as duas faces que os contêm, assim como as respectivas quadrículas.

Consulta-se a estrutura, com base nas faces e quadrículas, para determinar quais os raios calculados aí existentes. Estes são analisados para se determinar se existe, e qual é, o raio "mais semelhante" ao raio em processamento.

O raio em processamento recebe o valor da radiância do raio encontrado na pesquisa ou, na ausência deste último, é passado para o motor de radiância, para o seu cálculo exaustivo por *Ray-Tracing*.

6. PREENCHIMENTO DO HOLODECK

O sistema será tanto mais rápido quantos mais raios estiverem armazenados na estrutura de dados do *holodeck*, uma vez que, não havendo sucesso na consulta ao *holodeck* para avaliação de um raio, se torna necessário recorrer ao motor de radiância.

Assim, o motor de radiância serve para fornecer novos valores de radiância pedidos pelo processo de visualização. Estes serão então inseridos na estrutura do *holodeck*, para posterior utilização.

Numa situação de utilização em *walkthrough*, o utilizador desloca-se na cena virtual por pequenas distâncias entre duas imagens consecutivas. Desta forma, se uma percentagem significativa de radiâncias calculadas na imagem anterior tiverem sido armazenadas, existe uma grande probabilidade de estas serem encontradas na estrutura de dados, evitando-se o recurso ao motor de radiância.

Extrapolando o exposto, conclui-se da utilidade do preenchimento do *holodeck*, previamente ao processo de construção da imagem, ou seja, como pré-processamento.

No entanto, durante a síntese de uma imagem e sempre que uma visita ao *holodeck* resultar sem sucesso, o raio assim calculado pelo motor de radiância deve ser armazenado e devidamente inserido no *holodeck*. Trata-se portanto de outra forma de preenchimento, dinâmico, da respectiva estrutura de dados.

O processo de preenchimento do *holodeck* baseia-se na geração de um certo número de raios que intersectem o *holodeck*. Para efeitos de preenchimento por pré-processamento, definem-se dois pontos que pertençam ao *holodeck* e, em seguida, faz-se passar um raio por esses pontos. A informação de radiância é calculada pelo motor de radiância e é armazenada no *holodeck*.

A geração de cada raio processa-se da seguinte forma:

1. Selecciona-se aleatoriamente um par de faces (Figura 7 a);
2. Determina-se qual das duas faces corresponde à "face de entrada" e à "face de saída", respectivamente;
3. Determina-se a qual dos planos do sistema de eixos (*xy*, *xz* ou *yz*) a "face de entrada" é paralela (Figura 7 b);
4. Idem para a "face de saída";
5. Atendendo ao número de divisões de cada face, gera-se aleatoriamente uma quadrícula para a primeira e posteriormente para a segunda dimensão da "face de entrada" (Figura 7 c);
6. Idem para a "face de saída";
7. Determinam-se dois valores aleatórios, pertencentes ao intervalo [0;1[, para cada quadrícula. Estes valores

servem como coordenadas normalizadas locais, de um ponto situado em cada uma das quadrículas;

8. Traça-se um raio entre os dois pontos assim definidos, no sentido da quadrícula de entrada para a de saída (Figura 7 d);

9. Envia-se a informação do raio para o gestor do *holodeck*, que a armazena de acordo com as faces e quadrículas correspondentes, juntamente com a informação de coordenadas locais normalizadas e com a informação de radiância, esta última determinada pelo respectivo motor.

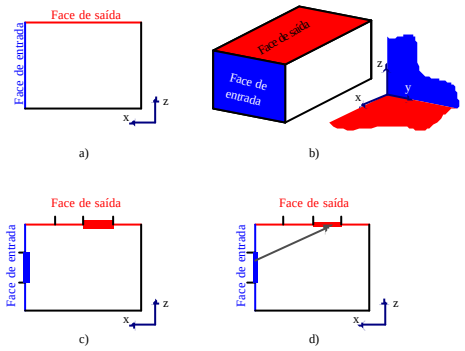


Figura 7: Preenchimento do *holodeck* em pré-processamento a) Selecção aleatória de duas faces. b) Determinação de paralelismo entre faces e planos principais. c) Determinação aleatória da divisão para cada dimensão das faces. d) Traçar raio.

Um outro método de preenchimento, sequencial, pode ser utilizado quando se pretende o preenchimento total do *holodeck*. Neste caso, não se seleccionam as quadrículas do *holodeck* de forma aleatória, pois quando é ultrapassada uma certa percentagem de preenchimento, a probabilidade de acertar nos pares de quadrículas que não estejam preenchidas torna-se muito baixa. Assim sendo, percorre-se sequencialmente toda a estrutura, para garantir que todas as quadrículas são visitadas. Por cada par de quadrículas geram-se todos os raios pretendidos.

Uma variante deste método permite o preenchimento parcial do *holodeck*, baseando-se na estratégia de "moeda ao ar". Quando se selecciona um par de quadrículas, de acordo com a mesma sequência do método anterior, gera-se um número aleatório entre 0 e 100. O valor deste é comparado com um valor de limiar (igual à percentagem de *holodeck* a preencher) e decide-se se é ou não gerado o raio correspondente.

De realçar que qualquer dos métodos anteriores pode ser utilizado para efeitos de preenchimento do *holodeck* em pré-processamento. Em qualquer situação, a estrutura do *holodeck* é actualizada durante o processo de visualização, à medida que se navega na cena virtual.

7. RESULTADOS

A cena de teste (fonte [NRC]) é formada por uma célula de escritório, a qual inclui uma mesa em forma de L, uma cadeira, armários, computador e outros objectos mais pequenos, dentro de uma sala com 6 luminárias no tecto. Um *holodeck* foi colocado na cena de forma a envolver toda a célula de trabalho.

Os resultados apresentados dependem de parâmetros que caracterizam a estrutura do *holodeck*: o número de divisões das três dimensões do *holodeck*, o número de raios que atravessam cada par de quadrículas, a resolução da imagem e o valor máximo de tolerância admitido para a semelhança entre raios a calcular e raios existentes.

Considerando N o número de divisões de cada dimensão do *holodeck* e F o número de raios passantes por cada par de quadrículas, conclui-se que o número total de raios inseridos no *holodeck* é:

$$K = 30 * N^4 * F \qquad (1)$$

No caso ideal em que todos os raios a calcular, para uma dada imagem, existam no *holodeck*, a visualização dos *pixels* respectivos depende apenas do tempo de acesso directo aos raios previamente calculados e armazenados. Ou seja, nesta situação, não é necessária qualquer traçagem de raios, sendo a imagem criada por um processo semelhante a um refrescamento.

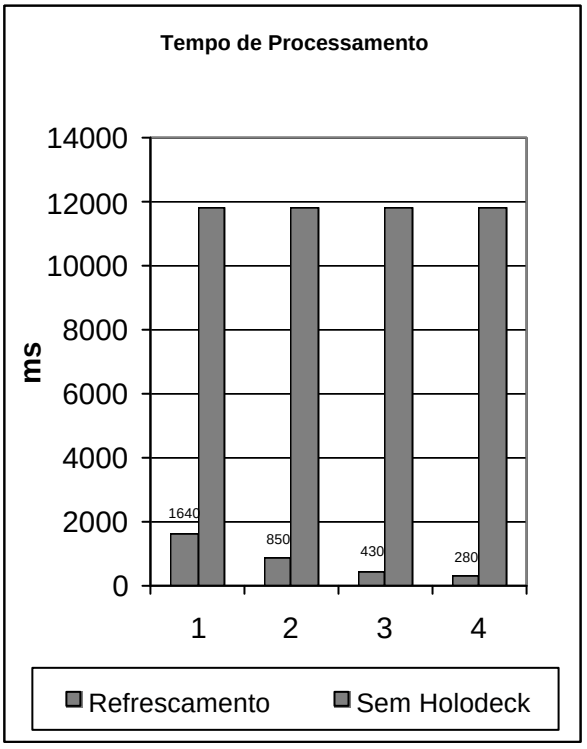


Figura 8: Tempos de processamento com *holodeck* ideal e sem *holodeck*

Na Figura 8 apresentam-se os tempos de refrescamento de uma imagem (*holodeck* ideal) em comparação com os respectivos tempos de cálculo completo pelo motor de radiância, em quatro situações que apresentam a mesma quantidade total de raios no *holodeck*:

- Caso 1: N=5, F=16384
- Caso 2: N=10, F=1024
- Caso 3: N=15, F=203
- Caso 4: N=20, F=64

A análise dessa figura mostra que o refrescamento com base no *holodeck* é altamente vantajoso em comparação com a traçagem total de raios.

Também se vê que é preferível aumentar o número de quadrículas, em detrimento do número de raios por par de quadrículas.

A utilização do *holodeck* processa-se em três fases:

- Fase "reutilização+falta de raios" – esta fase inicial pesquisa no *holodeck* os raios a calcular; caso existam raios armazenados semelhantes (ie, dentro de uma tolerância especificada pelo utilizador), efectua-se a sua reutilização; se não existirem raios calculados no par de quadrículas correspondente ao raio a calcular (falta de raios), é efectuada a traçagem desse raio e o resultado é inserido no *holodeck*.
- Fase "raios inválidos" – nesta fase processam-se os raios a calcular para os quais, no par de quadrículas correspondente, existem raios calculados, mas que não satisfazem a tolerância máxima definida; o processamento é realizado através de uma traçagem de raios, sendo os resultados inseridos no *holodeck*.
- Fase "refinamento de raios" – esta fase destina-se a melhorar a qualidade da imagem obtida nas fases anteriores e só é accionada se o utilizador o pretender (por exemplo, após parar a sua navegação pela cena). A fim de se obter um refinamento progressivo da imagem, esta fase é iterativa; em cada iteração processam-se os *pixels* da imagem cujos raios foram substituídos por raios semelhantes, mas que ultrapassam um certo valor percentual da tolerância máxima (parâmetro definido pelo utilizador); este valor percentual é decrementado a cada nova iteração.

Na aplicação desenvolvida, assume-se que a navegação na cena pode fazer-se de três formas:

- Rápida – neste tipo de navegação, o utilizador pretende explorar rapidamente a cena e não necessita de ver imagens com elevada qualidade; assim sendo, a imagem resultante da fase reutilização+falta de raios é suficiente para a compreensão espacial da cena. Caso se pretenda uma rapidez muito elevada, pode visualizar-se uma imagem simplificada, interpolada a partir dos *pixels* correspondentes a raios reutilizados, usando-se para tal uma técnica do tipo *flat-shaded cones* [Larson98]).
- Intermédia – se o utilizador pretende navegar na cena e, simultaneamente, visualizar a imagem da cena com algum detalhe, então deverão ser usados os resultados das fases reutilização+falta de raios e raios inválidos; esta visualização será mais lenta mas, ainda assim, com tempos de processamento bastante inferiores aos que se obtêm por traçagem total de raios.
- Detalhada – caso o utilizador pretenda visualizar uma imagem com elevado nível de qualidade, então poderá accionar a fase refinamento de raios para esse efeito; para evitar um bloqueio indesejável na navegação, esta fase é progressiva, apresentando-se sucessivamente imagens de maior qualidade.

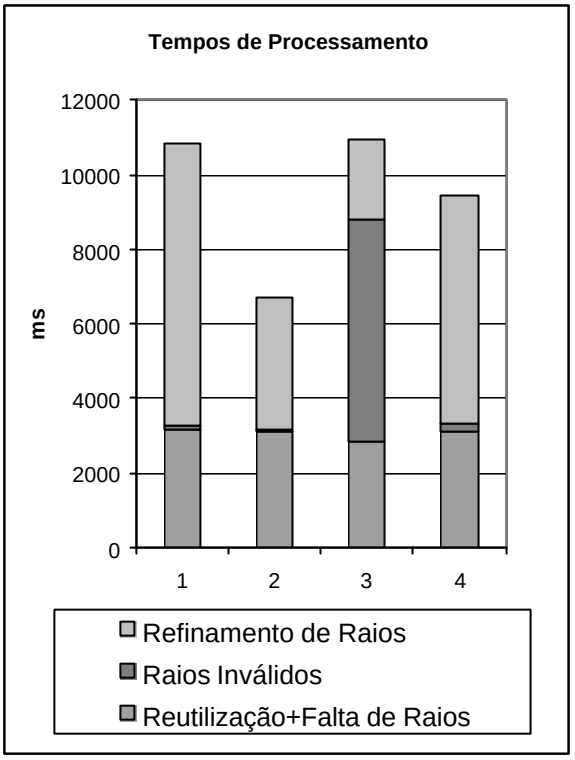


Figura 9: Tempos parciais de processamento (passo 2 cm)

Nas Figuras 9 e 10 apresentam-se tempos de navegação na cena sem pré-processamento do *holodeck*. As dimensões da cena são 12m x 10m x 2.5m. Os parâmetros gerais definidos pelo utilizador foram:

- Tolerância máxima 50%
- Redução iterativa de tolerância 50%

Consideraram-se as mesmas 4 situações de valores de N versus F anteriores:

Dado que a nova informação calculada vai sendo armazenada no *holodeck* para posterior utilização, coloca-se o problema de coerência espaço-temporal. Efectivamente, os novos raios poderão ser reutilizados nas imagens seguintes, se a posição do observador não se alterar demasiado face à posição actual. Pelo contrário, se o observador se movimenta em passos muito largos, os novos raios têm uma posição espacial muito diferente dos que estão armazenados e, portanto, há necessidade de recorrer mais regularmente ao processamento de raios pelo motor de radiância. Os resultados apresentados seguidamente, foram obtidos com duas velocidades diferentes do observador (correspondentes a passos de

avanço na cena de 2 cm e de 5 cm, respectivamente) entre imagens consecutivas.

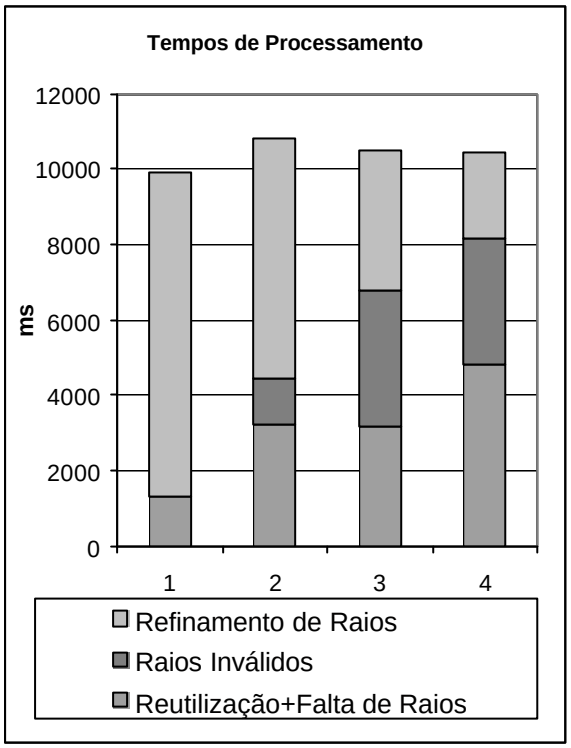


Figura 10: Tempos parciais de processamento (passo 5 cm)

A Figura 9 apresenta os tempos de processamento para cada uma das fases anteriormente descritas (incluindo três iterações da fase refinamento de raios) com passo de avanço na cena igual a 2 cm.

Tendo em conta que uma traçagem total de raios demora cerca de 11000ms, a Figura 9 mostra claramente os benefícios da navegação usando técnicas de reutilização de informação e de melhoramento progressivo da imagem. A navegação rápida consome, em média, cerca de 30% de traçagem total e, em conjunto com a navegação intermédia, consome cerca de 35%.

A Figura 10 apresenta os tempos equivalentes, mas com passo de avanço na cena igual a 5 cm. Neste caso, a navegação rápida consome entre 12% e 44% da traçagem total. Caso se considere acumulação com a navegação intermédia, os tempos de processamento variam entre 12% e 74% dos tempos de traçagem total.

Os tempos correspondentes à fase reutilização+falta de raios na Figura 10 crescem, nos quatro casos apresentados, com o número de quadrículas do *holodeck*. Tal observação parece contrariar a interpretação dos resultados apresentados na Figura 8. Realmente, na Figura 10 (e também na Figura 9), aquela fase inclui o processamento de raios quando se detecta que estes estão em falta. Pelo contrário, na Figura 8, não há lugar a cálculo de raios, dado que se trata de um refrescamento com informação já existente.

Assim, os resultados apresentados nas Figuras 9 e 10 são bastante mais realistas, dado que a situação dita "ideal", da Figura 8, necessita de um *holodeck* com um número incompatível de raios.

A Figura 11 apresenta a evolução da imagem ao longo das fases descritas anteriormente, para o caso do passo de avanço na cena igual a 2 cm.

a) Fase reutilização+falta de raios

b) Fase raios inválidos

c) Fase refinamento de raios (iteração 1)

d) Fase refinamento de raios (iteração 2)

e) Fase refinamento de raios (iteração 3)

Figura 11: Evolução das imagens nas fases de processamento (passo 2 cm)

Da observação da Figura 11 constata-se que a imagem inicial (fase reutilização+falta de raios) apresenta uma qualidade aceitável, sendo mostrados através de *pixels* a branco aqueles para os quais não é possível obter resultados dentro da tolerância especificada. Estes *pixels* desconhecidos são preenchidos pela fase seguinte (raios inválidos), com a qual se obtém uma imagem de qualidade quase final. Os resultados das iterações da fase refinamento de raios notam-se essencialmente nos *pixels* correspondentes a arestas de objectos.

A Figura 12 apresenta um conjunto de imagens que mostram a evolução das mesmas fases com o passo de avanço na cena igual a 5 cm. De salientar que a imagem inicial apresenta um grande número de *pixels* por colorir (representados por *pixels* a branco) devido ao facto do passo de avanço, neste caso, ser superior ao da figura anterior e, consequentemente, o número de raios armazenados que não estão dentro da tolerância máxima tender a ser superior (ie, haver menos coerência entre imagens de passos sucessivos).

Da observação da Figura 12 retira-se que, após as duas fases iniciais, a imagem obtida é suficientemente compreensível para ajudar o utilizador a ter uma boa percepção da cena. A fase de refinamento de raios actua apenas para eliminar artefactos visuais em regiões da imagem nas quais existem variações bruscas de visibilidade (muito perceptíveis, por exemplo, nas arestas de objectos) ou de iluminação (pouco perceptíveis).

8. CONCLUSÕES

Este artigo apresentou uma implementação de avaliação do método *holodeck*, proposto por [Larson98]. Relativamente a outras soluções, este método tem a vantagem de permitir a utilização, em ambiente de *walkthrough*, de um motor de radiância fisicamente válido, numa plataforma computacional comum (PC). Para além disso, apresenta um faseamento explícito de refinamento da imagem.

Os resultados obtidos permitem concluir que o método *holodeck* apresenta tempos de geração de imagem bastante inferiores aos tempos necessários quando se utiliza exaustivamente o motor de radiância.

Tal redução de tempo de processamento ocorre à custa da redução de qualidade de imagem. Presume-se no entanto que, durante o movimento do observador, durante um *walkthrough*, essa redução seja aceitável, na condição de sem perda de tempo relativamente ao cálculo completo de uma imagem, ser possível incrementar essa qualidade sempre que o utilizador o deseje, ao fixar-se numa posição.

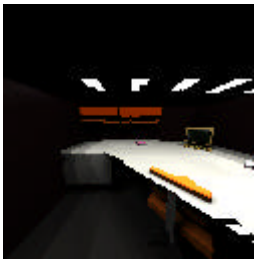
O trabalho encontra-se numa fase preliminar e prevê-se que se possa dar-lhe continuidade em algumas direcções de investigação. Assim, em termos de trabalho futuro, anteveêm-se desde já algumas melhorias.

O motor de radiância usado para a traçagem de raios funciona como uma "caixa negra" (recebe raios e devolve radiâncias), pelo que o emprego de técnicas de paralelização na traçagem desses raios poderá aumentar o desempenho interactivo desta aplicação.

a) Fase reutilização+falta de raios

b) Fase raios inválidos

c) Fase refinamento de raios (iteração 1)



d) Fase refinamento de raios (iteração 2)

e) Fase refinamento de raios (iteração 3)

Figura 12: Evolução das imagens nas fases de processamento (passo 5 cm)

O pré-preenchimento do *holodeck* com base na geometria da cena, a fim de lançar maior número de raios provenientes de locais mais prováveis de passagem, tais como corredores, portas, escadas, etc., torna-se muito interessante para futura implementação.

A utilização de vários *holodecks* numa cena mostra-se, face aos testes efectuados, bastante promissora, na medida em que estes podem ser colocados em regiões de geometria ou iluminação complicada, assim se evitando traçagens demoradas de raios. Um dos objectivos de trabalho futuro é o suporte de vários *holodecks* numa cena, a fim de se melhorar a navegação interactiva. A localização destes *holodecks* poderá resultar de um processo semi-automático de análise da complexidade da cena.

9. REFERÊNCIAS

[Glassner95] Glassner, Andrew S., *Principles of Digital Image Synthesis*, Morgan Kaufmann Publishers, Inc., 1995.

[Gortler96] Gortler, Steven, Radek Grzeszczuk, Richard Szeliski, Michael Cohen, *The Lumigraph*, Computer Graphics Proceedings, Annual Conference Series, 1996.

[Larson97] Larson, Greg Ward, Rob Shakespeare, *Rendering with Radiance. The Art and Science of Lighting Visualization* Morgan Kaufman, 1997.

[Larson98] Larson, Greg Ward, *The Holodeck: A parallel Ray-caching Rendering*, 2nd Eurographics Workshop on Parallel Graphics and Visualization, 1998.

[Levoy96] Levoy, Marc and Pat Hanrahan, *Light Field Rendering*, Computer Graphics Proceedings, Annual Conference Series, 1996.

[Painter89] Painter, James and Kenneth Sloan, *Antialiased Ray-Tracing by Adaptive Progressive Refinement*, Computer Graphics Proceedings, Annual Conference Series, 1989.

[Pighin97] Pighin, Frédéric, Dani Lischinski, David Salesin, *Progressive Previewing of Ray-Traced Images Using Image-Plane Discontinuity Meshing*, 8th Eurographics Workshop on Rendering, Saint-Etienne, France, June 1997.

[Ward94] Ward, Greg The RADIANCE *Lighting Simulation and Rendering System*, Computer Graphics Proceedings, Annual Conference Series, 1994.

[Whitted80] T. Whitted, *An Improved Illumination Model for Shaded Display*, Communications of the ACM, vol. 23, no. 6, June 1980, pp. 343-349.

[NRC] Institute for Research in Construction do National Research Council, Canadá, www.nrc.ca/irc/

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Síntese de Imagem para Ambientes Virtuais Experiências com a técnica RENDERCACHE

Alexandre Valle de Carvalho
INESC Porto
Porto
alexandre.carvalho@inescn.pt

António Augusto Sousa
FEUP/INESC Porto
Porto
aas@fe.up.pt

António Cardoso Costa
ISEP/INESC Porto
Porto
acc@dei.isep.ipp.pt

Sumário

Um dos grandes objectivos da Computação Gráfica no domínio dos 3D é a síntese de imagens realistas. Os últimos avanços conseguidos na área, dotados de modelos matemáticos rigorosos dos principais fenómenos da física, garantem um realismo fotográfico mas, infelizmente, à custa de grandes esforços computacionais, impeditivos da sua utilização em ambientes interactivos típicos dos sistemas de realidade virtual. A técnica RENDERCACHE, apresentada em [Walter99], é uma das mais promissoras em termos de utilização nestes contextos, sobre arquitecturas banais baseadas em computadores pessoais. Faz uso de técnicas simplificadas de síntese de imagem, como a reprojecção e a interpolação, e permite o reaproveitamento de informação entre fotogramas, na forma de amostras da cena, mantidas numa estrutura do tipo cache e geridas de acordo com o envelhecimento das mesmas. Este documento apresenta uma implementação experimental da técnica RENDERCACHE, que serve como banco de ensaios para alguns melhoramentos, dos quais se destacam o tratamento perceptual dos resultados visuais e a utilização do software RADIANCE, a funcionar em arquitectura paralela, como motor de informação.

Palavras-chave

Image-based rendering, reprojecção, coerência, arquitecturas paralelas, perceptualidade.

1. INTRODUÇÃO

A visualização de imagens sintéticas fotorrealísticas, fisicamente correctas, tem sido uma das metas principais da Computação Gráfica. Porém, este objectivo atinge-se à custa de cálculos complexos que consomem grandes quantidades de tempo de cálculo. Daí que, durante algum tempo, se tenha considerado que a interactividade era um objectivo mutuamente exclusivo da produção de imagens sintéticas de qualidade elevada. Este conflito de objectivos traduz-se pelo tempo que é necessário dispender na produção de cada imagem, quando se contemplam os fenómenos da iluminação global fisicamente correcta. Desta forma, o enorme esforço computacional envolvido sempre foi considerado um elemento comprometedor da interactividade e restringiu esta à produção de imagens de baixa qualidade, através da utilização de algoritmos acelerados por *hardware* como modelo de linhas *(wireframe)* ou *scan-conversion* [Hearn94], [Harrington97].

Porém, a recente banalização dos recursos computacionais, em simultâneo com o desenvolvimento de novas técnicas e algoritmos, permitiu a criação de algumas soluções mais eficientes do que a utilização massiva de *hardware* ou de estações gráficas especializadas, ambas bastante dispendiosas e fora do alcance da esmagadora maioria dos utilizadores. Algumas soluções novas atingem a interactividade mantendo, dentro dos limites possíveis, a qualidade de um motor radiométrico mais complexo, no que concerne ao realismo da imagem. Estas soluções de síntese de imagem interactiva utilizam um nível de poder computacional modesto e, consequentemente, de baixo custo. Em contextos de navegações em ambientes virtuais com qualidade elevada e da produção de animações, a principal característica destas soluções é terem a capacidade de reutilizar informação, isto é, utilizar a informação anteriormente determinada para um conjunto de fotogramas na produção do fotograma actual. A reutilização da informação é possível porque estas soluções exploram a coerência espacial e temporal existente entre fotogramas.

2. MOTIVAÇÃO

Neste artigo apresenta-se o problema de produzir imagens sintéticas, com informação fotorrealista, para contextos onde se permite ao utilizador navegar em cenas virtuais. Neste âmbito identificam-se as duas classes principais de aproximações a este problema, baseada em imagem e baseada em geometria, e apresentam-se as vantagens e desvantagens de cada uma. Relativamente às aproximações baseada em imagem, descrevem-se os princípios em que assenta, nomeadamente a coerência espaço-temporal e a reprojecção da informação. Desta

última, focam-se alguns dos problemas que resultam da sua utilização.

Em seguida apresentam-se algumas das técnicas existentes, que recorrem a síntese de imagem *image-based* para navegações virtuais. Destas, aprofunda-se a técnica de RENDERCACHE [Walter99], da qual resultou o desenvolvimento de um sistema experimental que possibilitou dar continuidade a algum do trabalho futuro proposto por Walter *et al*., nomeadamente a utilização do *software* RADIANCE como motor de cálculo e a inclusão de tratamento perceptual dos resultados visuais. Com base nos resultados obtidos, tecem-se conclusões e apresentam-se perspectivas de trabalho futuro.

3. SÍNTESE DE IMAGEM

As aproximações ao problema de síntese de imagem em navegações virtuais apresentam-se, na actualidade, divididas em duas classes distintas: as que trabalham no espaço imagem, designadas por síntese de imagem baseada em imagem (síntese de imagem *image-based*) e as que trabalham ao nível do modelo geométrico que descreve a cena, designadas por síntese de imagem baseada na geometria (síntese de imagem baseada na geometria).

Segundo McMillan [McMillan97] ambas as aproximações apresentam vantagens e desvantagens: as baseadas em imagem destacam-se pela capacidade de produzir resultados fotorrealistas com um esforço computacional que é proporcional ao número de *pixels* da imagem e totalmente independente das características geométricas da cena. No entanto, uma vez que constituem representações adimensionais, são sobretudo recomendadas para efeitos de visualização. Já as aproximações baseadas em geometria possibilitam uma aproximação modular ao problema de converter uma especificação geométrica numa imagem [McMillan97]. Porém, devido à procura incessante de realismo, os modelos geométricos têm vindo a ficar progressivamente mais complexos, acarretando um esforço computacional que, nas aproximações baseadas em geometria, é proporcional ao número de primitivas da cena, de onde resulta que esta classe de aproximações não constitui a solução mais apropriada para ambientes interactivos.

3.1 Aproximações Baseadas em imagem

As aproximações da classe síntese de imagem *image-based* baseiam-se na existência de um conjunto de imagens, designadas por imagens-chave, geradas a partir de diferentes localizações da cena, ou capturadas a partir de perspectivas distintas do mundo real.

Com base neste conjunto de imagens são produzidas as imagens intermédias, que em contextos de animação ou navegações virtuais se designam por fotogramas, através de um conjunto de técnicas das quais se destacam a reprojecção e a interpolação.

No entanto, a utilização de imagens como fonte de informação apresenta algumas dificuldades, das quais se destaca a dependência do ponto de vista para o qual cada imagem foi produzida ou capturada. Isto significa que é necessário empregar uma técnica que, com base em informação dependente de determinadas localizações, produza imagens a partir de quaisquer pontos de vista. Esta técnica, designada por reprojecção [Chen93], permite, considerando um ponto de visualização arbitrário, determinar qual a contribuição de uma imagem-chave para essa localização.

A utilização da reprojecção apresenta um requisito, nem sempre fácil ou possível de cumprir: para se reprojectarem correctamente os *pixels* de uma imagem-chave é indispensável o conhecimento, por cada *pixel*, da sua informação de profundidade. O cálculo desta, conhecida a geometria da cena, constitui um procedimento trivial, semelhante ao que se executa em algoritmos de *ray-tracing* [McMillan97]. Assim, a aproximação baseada em imagem ao problema da visualização independente do ponto de vista, resume-se a uma operação onde se tenta maximizar a exploração da coerência espaço-temporal existente entre imagens.

3.1.1 Coerência Espaço-temporal

A coerência espaço-temporal consiste num conjunto de características que se verificam ser idênticas entre dois fotogramas consecutivos de uma animação. Por exemplo, muitos dos objectos da cena visíveis no primeiro fotograma mantêm-se visíveis no segundo fotograma, e em localizações semelhantes. Desta forma, quanto mais pequenas forem as diferenças dos parâmetros da câmara entre dois fotogramas consecutivos, maior tende a ser a informação que é comum. Por outro lado, o tempo que decorre entre dois fotogramas consecutivos é diminuto. Partindo deste princípio, assume-se que as diferenças na geometria, nos materiais ou na iluminação da cena devem ser pequenas.

A exploração da coerência espaço-temporal que existe entre dois fotogramas consecutivos possibilita, em princípio, a redução do esforço computacional associado à produção de fotogramas novos. Um algoritmo que explore esta coerência poderá extrair, do primeiro fotograma ou de fotogramas anteriores, um conjunto de informação que é relevante ao contexto do fotograma em produção.

Sendo possível explorar a coerência espaço-temporal, torna-se desejável reaproveitar a informação de fotogramas previamente determinados na produção do fotograma actual.

3.1.2 Reprojecção

A reprojecção (*image warping*) é uma das técnicas mais utilizadas em síntese de imagem *image-based*, com o objectivo de acelerar a cadência de produção de imagens novas, através da redução do número e da complexidade dos cálculos efectuados. A redução do número de cálculos obtém-se pela eliminação da necessidade de calcular toda a informação da imagem nova. Isto significa que muita da informação do fotograma em produção consiste na reutilização de informação de imagens-chave

ou de informação calculada no contexto de fotogramas que, por sua vez, resultaram de reprojecção.

Relativamente à complexidade dos cálculos, considera-se que a operação de reprojectar um elemento de informação acarreta cálculos mais simples do que a determinação desse mesmo elemento recorrendo a um motor de síntese de imagem baseado em cálculos dispendiosos de intersecção de raios com objectos.

Genericamente, a reprojecção permite determinar a correspondência de *pixels* de uma imagem inicial que representa a cena vista a partir de uma determinada localização, numa outra imagem que representa a mesma cena, mas visualizada de uma localização distinta da primeira. Para que a reprojecção seja possível é necessário conhecer os parâmetros da câmara e o mapa de profundidade associados a cada um dos fotogramas fonte.

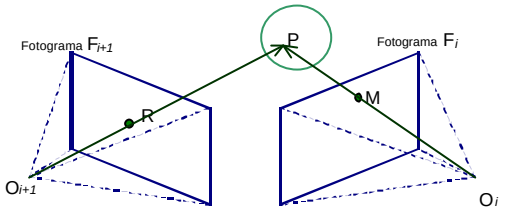


Figura 1: Reprojecção do *pixel* M, obtido no fotograma F_i, no *pixel* R do fotograma F_{i+1}.

Considerem-se os dois fotogramas consecutivos F_i e F_{i+1}, da sequência de fotogramas de uma animação, apresentados na Figura 1. Assumindo que os movimentos da câmara que permitem a obtenção destes dois fotogramas são reduzidos e que o modelo geométrico da cena permanece invariável, é possível supor que são pequenas as diferenças da informação entre os fotogramas F_i e F_{i+1} [Jevans92]. Desta forma, um número elevado de *pixels* do fotograma F_i será aproveitado na produção do fotograma F_{i+1}. Como se pode observar na Figura 1, a reprojecção consiste no mecanismo que permite determinar, para cada *pixel* M do fotograma F_i, qual a sua correspondência no fotograma F_{i+1}, ou seja, as coordenadas de R. O conjunto de operações executado para determinar esta correspondência é o seguinte:

1. Para cada *pixel* M do fotograma F_i, supõe-se ser conhecida a distância do observador O_i ao objecto visível nesse *pixel* (informação constante no mapa de profundidades mencionado anteriormente). As coordenadas do ponto P de um *pixel* M do fotograma podem assim ser obtidas pela multiplicação do versor O_iM pelo valor da profundidade nesse *pixel*.
2. No fotograma F_{i+1}, para cada ponto P obtido na etapa anterior, determina-se o segmento de recta que passa por O_{i+1} e por esse mesmo ponto. A intersecção desta recta com o plano imagem de F_{i+1} permite a determinação do ponto R do exemplo da Figura 1.
3. O ponto R, que se encontra em coordenadas mundo, é mapeado para as coordenadas 2D do plano da imagem em produção e atribui-se-lhe a informação de iluminação que consta no ponto M.

Apesar da reprojecção ser um instrumento rápido de determinação de fotogramas novos, esta técnica acarreta algumas desvantagens, nomeadamente ao nível do mapeamento da informação e da iluminação.

As inconsistências de mapeamento resultam do facto de nem sempre existir correspondência unívoca de um *pixel* de F_i para um *pixel* de F_{i+1}: um *pixel* do novo fotograma pode ser alvo de múltiplos *pixels* do fotograma inicial [Walter99] ou, pelo contrário, não corresponder a nenhum destes *pixels*. Enquanto que a correspondência de múltiplos *pixels* pode ser facilmente resolvida pela utilização de um algoritmo *Z-Buffer* [Hearn94] [Harrington97] ao nível da informação de profundidade de cada *pixel* mapeado, já se torna mais difícil resolver a falta de mapeamento de *pixels* iniciais em *pixels* do fotograma novo. Assim, a solução pode ter de passar pela utilização de técnicas de reprojecção inversa [McMillan97], computacionalmente muito dispendiosas.

Ao nível da iluminação, a utilização de objectos com materiais especulares associados resulta na variação dos valores de luminância com a posição do observador [Foley90]. Daqui resulta que o efeito especular varia na superfície dos objectos de acordo com a localização das luminárias e do ponto de observação actual. Em consequência, surgem artefactos de iluminação no fotograma em produção, resultantes da reprojecção de reflexões especulares, inconsistentes com o ponto actual de observação da cena.

4. SÍNTESE DE IMAGEM PARA AMBIENTES INTERACTIVOS

Enquanto que a maior parte da investigação em síntese de imagem *image-based* incide sobre o cálculo de imagens isoladas, alguns investigadores procuram encontrar soluções para contextos de animação e navegações virtuais interactivos. Aqui, a utilização de uma aproximação puramente baseada em imagem é por vezes considerada penalizadora, do ponto de vista do espaço de armazenamento dos fotogramas-chave e da liberdade de movimentação na cena. Isto justifica-se pela necessidade de calcular ou capturar um conjunto bastante alargado de fotogramas-chave, que são fulcrais para a caracterização da cena e que permitem a produção de imagens novas, interpoladas, independentes do ponto de observação. Da constatação desta desvantagem resulta o facto de algumas das soluções de visualização de cenas em contextos de navegações virtuais interactivas recorrerem a aproximações baseadas simultaneamente em imagem e em geometria. Estas soluções híbridas caracterizam-se pela utilização preferencial da aproximação baseada em imagem, computacionalmente menos exigente, na produção de fotogramas novos. A aproximação baseada em geometria só é utilizada quando, para cada fotograma em produção, são encontradas regiões onde a informação reprojectada é esparsa ou inexistente. Nesta situação, a

informação em falta é determinada por um motor de síntese de imagem baseado na geometria da cena.

Do conjunto de soluções de síntese de imagem para ambientes interactivos apresentam-se, nesta secção, as técnicas de *Post-Rendering Warp 3D*, HOLODECK e RENDERCACHE.

A técnica *Post-Rendering Warp 3D* [Mark97] é uma aproximação baseada em imagem ao problema de visualização interactiva em que a cadência de imagens se pretende superior à do cálculo do motor de síntese de imagem. Apenas algumas imagens-chave são calculadas e entre cada par de imagens-chave são interpoladas várias imagens através de reprojecção. Esta técnica apresenta uma grande desvantagem: é necessário prever, com alguma antecedência, as coordenadas e orientações da câmara virtual, de modo a calcular as imagens-chave que são necessárias para a interpolação. Enquanto que a previsão dos parâmetros da câmara é trivial no contexto de uma animação em que se conhece o percurso da câmara, a utilização num contexto interactivo, como é o caso de uma navegação virtual, é extremamente difícil.

O HOLODECK [Larson98], outra aproximação ao problema da visualização interactiva, é um método que permite navegações virtuais em cenas complexas e foi desenvolvido inicialmente sobre o *software* de síntese de imagem RADIANCE [Ward94]. Este método, do tipo baseado em geometria, possibilita a determinação de informação geométrica e de iluminação em pré-processamento, sendo possível o seu armazenamento em ficheiros para utilização posterior, o que constitui uma vantagem para determinadas aplicações.

Segundo [Larson98], a estrutura de dados do HOLODECK consiste num cubo que contém a cena e que está dividido em regiões 3D, designadas por secções. As faces de cada secção estão divididas em células. O preenchimento da estrutura consiste em, para cada raio que atravessa uma secção, armazenar as coordenadas 2D das células de entrada e de saída, o valor de radiância do raio e o seu comprimento. O pré-processamento do HOLODECK consiste em lançar raios aleatoriamente para dentro do cubo referido, armazenando-se a informação respectiva. Durante o cálculo de um fotograma e ao lançar-se um raio para dentro da cena, determina-se a secção e o par de células do cubo que são intersectadas. Nestas células, o algoritmo pesquisa um raio previamente armazenado que mais se aproxime espacialmente do raio agora lançado. Só na situação em que, para uma dado par de células, não existe um raio suficientemente próximo do raio lançado, se torna necessário efectuar a determinação da informação geométrica e de iluminação para este último, através dos métodos usuais [Larson98], [Ward94].

As principais desvantagens do HOLODECK são, de acordo com Walter *et al*. [Walter99], a quantidade de memória que a estrutura do HOLODECK necessita e a incapacidade de trabalhar com cenas dinâmicas, ao nível da geometria, das luminárias ou dos materiais.

A técnica RENDERCACHE constitui uma das soluções híbridas para navegações virtuais, vocacionada para cenários onde o motor de síntese de imagem é demasiado complexo e, consequentemente, demasiado lento para produzir continuamente fotogramas completos, com informação de iluminação fisicamente correcta e em ambiente interactivo.

Esta capacidade obtém-se pela execução, para cada fotograma a produzir, de duas tarefas:

1. A reutilização da informação previamente determinada, e que possibilitou a produção dos últimos fotogramas. Esta tarefa, tipicamente baseada em imagem, assenta na operação de reprojecção.
2. A requisição de informação nova a um motor de síntese de imagem. O recurso a esta fonte de informação, que o RENDERCACHE tenta minimizar em virtude do elevado peso computacional associado aos cálculos de intersecção de raios com a geometria da cena, acontece quando o algoritmo detecta regiões no fotograma actual que não são possíveis de preencher com a informação previamente armazenada.

A principal característica desta técnica consiste na capacidade de proporcionar interactividade, mesmo quando a cadência da determinação de informação pelo motor de síntese de imagem é demasiado lenta. Nestas circunstâncias, o fraco desempenho deste motor é compensado pelo mecanismo de interpolação, que tenta colmatar as regiões do fotograma para as quais não existe qualquer informação. Adicionalmente, a técnica RENDERCACHE apresenta-se flexível quanto ao funcionamento com diversos motores de síntese de imagem. O requisito mínimo que estes devem cumprir consiste na capacidade de calcular raios individualmente ou *pixels* isolados. Neste sentido, os motores de síntese de imagem ditos orientados ao raio, como por exemplo os *ray-tracers*, são excelentes candidatos.

5. SISTEMA EXPERIMENTAL RENDERCACHE

Com o intuito de dar continuidade ao trabalho futuro proposto por Walter *et al*., desenvolveu-se um sistema experimental, baseado na técnica RENDERCACHE [Walter99], onde se executam as seguintes funções:

1. Geração dos fotogramas: é o conjunto de etapas que permite a determinação de fotogramas, ou seja, de aproximações à imagem correcta, baseada no ponto de observação actual e na informação que se encontra armazenada nas estruturas de dados.
2. Amostragem de pedidos de amostra: é o conjunto de etapas que permitem identificar quais as amostras que devem ser solicitadas ao motor de síntese de imagem. Uma vez que o número de amostras solicitadas se encontra limitado pelos requisitos de interactividade do sistema, é necessário identificar quais as amostras que se supõem maximizar o aspecto visual do fotograma em produção.

3. Gestão das estruturas de informação do sistema: conjunto de funções que gerem a integração de informação nova nas estruturas do sistema, sempre que necessário e apropriado, e a eliminação da informação considerada desactualizada relativamente ao contexto actual de visualização. Do conjunto de estruturas do sistema destacam-se as amostras, elementos que albergam a informação geométrica, temporal e de iluminação para um ponto de um objecto da cena, e a *cache* de amostras (ou apenas *cache*), um vector de amostras que detém todas as amostras disponíveis no sistema para a produção de fotogramas.

5.1 Geração de fotogramas

Inicialmente, o sistema começa por reprojectar, de acordo com a posição do observador, a informação na *cache* de amostras. O resultado desta etapa é um fotograma parcialmente preenchido, podendo existir várias inconsistências de oclusão. Estas têm origem na movimentação de objectos (em cenas dinâmicas) ou à própria deslocação do ponto de observação, fenómeno intrínseco a uma navegação virtual.

Com o intuito de resolver as inconsistências de oclusão utilizou-se a heurística proposta por Walter *et al.* [Walter99], combinada com um filtro digital. Para cada *pixel* do fotograma determina-se se a sua informação de profundidade está inconsistente com a dos *pixels* vizinhos. Importa afirmar que se verificou, tal como Walter *et al.* afirmara, que tanto a heurística como o filtro digital utilizados são simples e rápidos, contribuindo para manterem modestos os requisitos computacionais do sistema e resolvendo a maioria dos problemas de oclusão. No entanto, como resultado da simplicidade da heurística, algumas ligações entre *pixels* e amostras associadas são incorrectamente removidas. Verificou-se que isto acontece sobretudo em *pixels* cuja informação geométrica da amostra associada pertence à aresta de uma superfície da cena. Nesta situação, algumas arestas da cena podem representar descontinuidades de profundidades relativamente ao ponto actual de observação. Porém, as avaliações incorrectas são, na sua maioria, camufladas pela execução da etapa seguinte, de Interpolação. Nesta procede-se ao preenchimento das regiões do fotograma, desde que de pequena dimensão, sem qualquer informação associada. Para este efeito utilizou-se um filtro digital de interpolação / suavização, de dimensão 3x3 que consiste na média pesada dos valores de iluminação da vizinhança de cada *pixel* sem informação associada, após o que o fotograma actual se encontra produzido.

5.2 Amostragem de Pedidos de Amostra

Após a finalização da etapa de interpolação, o sistema inicia a etapa de amostragem, elemento crucial no processo de determinação de amostras novas. Nesta etapa pretende-se determinar os pedidos que devem ser solicitados ao motor de síntese de imagem, de modo a poder integrar informação nova na *cache* de amostras, que caracterize o contexto actual de visualização da cena e permita produzir novos fotogramas com informação consistente e actualizada. Porém, torna-se essencial limitar o número de solicitações a efectuar ao motor de síntese de imagem pela sua incapacidade em determinar um número elevado de amostras em tempo útil. A selecção dos pedidos a efectuar, proposta por Walter *et al.* verifica quais são as regiões do fotograma que contêm menos informação ou informação menos actual e que, por este motivo, devem ser mais amostradas. Desta forma, desenvolveu-se a etapa de amostragem com base na sugestão do autor, utilizando-se um algoritmo de *dithering* [Floyd76]. Isto possibilitou a obtenção de uma boa distribuição espacial dos pedidos a solicitar e impediu a concentração em demasia de pedidos numa determinada região do fotograma actual. Como critério de amostragem utilizou-se o valor de prioridade de cada *pixel*, que é baseado na idade da amostra associada e na execução de heurísticas ao longo das etapas de geração do fotograma. Este valor, determinado para cada *pixel*, representa a urgência relativa em solicitar ao motor de síntese de imagem uma amostra nova. Esta urgência encontra-se, segundo Walter *et al.* [Walter99], limitada entre zero, que corresponde à prioridade mínima, e 255, o valor da prioridade máxima.

A imagem de prioridades, isto é o conjunto da informação de prioridade dos *pixels* do fotograma actual é calculada por heurísticas que funcionam da seguinte forma:

1. Durante a reprojecção, sempre que se estabelece uma associação entre um *pixel* do fotograma e uma amostra da *cache*, a prioridade do *pixel* é inicializada com a idade da amostra. Esta inicialização pretende dar mais prioridade aos *pixels* cuja informação se obtém a partir das amostras com mais idade e, provavelmente, menos consistentes com o estado actual de observação da cena.
2. Na etapa de interpolação, todos os *pixels* que não possuem uma amostra associada recebem o valor máximo de prioridade. O valor de prioridade dos *pixels* restantes é obtido a partir do número de *pixels* vizinhos com amostras associadas e da idade destas. Aqui, pretende-se dar mais prioridade a regiões do fotograma que não contêm qualquer informação sem, no entanto, ignorar as restantes regiões e sem descurar a actualidade das amostras subjacentes.

Com base na imagem de prioridades e no número máximo de pedidos que podem ser solicitados ao motor de síntese de imagem, determina-se o valor que representa a prioridade com a qual se faz, nesta etapa, a binarização da Imagem de Prioridade. Este processo consiste na execução do algoritmo de *dithering* de Floyd-Steinberg [Floyd76] e resulta na determinação do conjunto de pedidos a solicitar ao motor de síntese de imagem.

5.3 Determinação de Amostras Novas

A componente de determinação de amostras desenvolvida resulta do trabalho futuro que Walter *et al.* Este autor propõe em [Walter99], a utilização do *software*

RADIANCE [Ward94] como motor de cálculo de amostras. Esta componente é, no sistema que se desenvolveu, um módulo que se executa local ou remotamente e que consiste na implementação sobre uma arquitectura paralela do tipo *farm* de processos, resultando estes de desenvolvimentos feitos sobre a aplicação *rtrace* do *software* RADIANCE. O recurso a uma arquitectura paralela resulta de se constatar que a utilização de uma única instância *rtrace* introduz limitações ao número de pedidos solicitados pela etapa de amostragem, o que pode comprometer a qualidade dos resultados visuais.

Nesta componente, desenvolvida sobre o *software* PVM [Geist94], contemplaram-se dois tipos de processos, que realizam tarefas distintas: um *farmer*, que recebe da componente RENDERCACHE os pedidos de amostra e um ou mais processos *workers*, que consistem nas instâncias *rtrace* e são os responsáveis pela determinação das amostras.

Desta forma, quando o *farmer* recebe um pacote de pedidos de amostra, distribui-os, segundo o modelo *farm* de processos, pelos vários *workers* disponíveis que se encarregam de determinar as amostras correspondentes e de as devolver ao *farmer*. Quando este detecta que todas as amostras de um pacote de pedidos se encontram determinadas, devolve os resultados à componente RENDERCACHE que imediatamente se encarrega de as integrar na *cache* de amostras.

Nas experiências efectuadas, verificou-se que a componente de cálculo de amostras apresenta uma eficiência razoável, para um número pequeno de *workers*, mas também se constatou que essa eficiência diminui quando o número de *workers* é elevado. Supõem-se que tal se deve à incapacidade do *farmer* em gerir atempadamente o fluxo de dados/resultados que envia e recebe de cada *worker*. A confirmar-se esta causa, uma das soluções mais imediatas passa pelo desenvolvimento de uma hierarquia de *farmers*.

5.4 Gestão das Estruturas de Informação do Sistema

Embora seja desejável armazenar o universo de amostras determinadas desde o inicio da sessão interactiva, os elevados custos de manutenção deste volume de informação inviabilizam este tipo de abordagem. É esta a principal razão pela qual se limita a dimensão da estrutura de armazenamento de amostras, que detém apenas as amostras determinadas mais recentemente. Walter *et al.* sugere que a *cache* de amostras possa armazenar um número de amostras igual a 150% do número de *pixels* da janela de visualização de fotogramas. De facto, verificou-se que a utilização de uma *cache* com dimensão muito inferior acarreta uma rotatividade excessiva das amostras na *cache*, ou seja, a substituição prematura de amostras mais antigas, mas ainda consistentes com o estado actual de visualização da cena. Por outro lado, uma *cache* com dimensão muito superior implica muito tempo dispendido na etapa de reprojecção, dado que se reprojectam amostras cuja informação se encontra desactualizada e nada contribui para a produção do fotograma actual).

Porém, a limitação da dimensão da *cache* acarreta a necessidade de um mecanismo expedito, responsável pela integração de amostras novas e consequente libertação de amostras com mais idade. Esta tarefa apresenta-se mais complexa do que parece inicialmente, uma vez que a manutenção de uma lista de idades das amostras, embora elegante, se apresenta computacionalmente dispendiosa. Desta forma, optou-se por seguir a sugestão de Walter *et al.*, onde se considera a *cache* de amostras dividida em grupos de oito elementos. Sempre que é necessária a integração de uma amostra nova, o mecanismo pesquisa, no grupo actual, um elemento que ainda não alberge uma amostra. Quando este elemento existe, é imediatamente ocupado pela amostra nova e o mecanismo termina a sua execução. Em simultâneo com a pesquisa anterior, é guardado o índice que corresponde ao elemento do grupo cuja amostra possui o maior valor de idade. Assim, quando todos os elementos do grupo estão ocupados, a amostra nova substitui a amostra cujo valor de idade é o maior. Verificou-se que este mecanismo permite simultaneamente a inclusão de novas amostras de uma forma rápida, libertando eficazmente amostras mais antigas.

Outro mecanismo de gestão da informação é aquele que se encarrega de manter a consistência da idade das amostras na *cache*, ao longo da produção de fotogramas. Este mecanismo opera ao nível da integração de amostras novas, onde se inicializa em zero a idade da amostra integrada, e após a etapa de amostragem, onde ao percorrer a *cache*, se incrementa a idade de cada amostra.

5.5 Inclusão de Perceptualidade

No sistema desenvolvido, após a etapa de amostragem e em paralelo com a determinação das amostras novas, procede-se ao tratamento perceptual do fotograma produzido, de modo a permitir a sua correcta visualização pelo observador humano. Esta etapa surge pela incapacidade apresentada pelos dispositivos de visualização tradicionais em reproduzirem eficazmente a gama alargada de luminâncias da cena. Daí que se utilizem métodos, designados por operadores de *tone mapping* e vulgarmente englobados no tema "Perceptualidade", para retirar partido do facto da visão humana ser sensível a valores relativos de luminância e não aos valores absolutos [Larson97]. Estes operadores aproximam-se da solução óptima de mapear a gama de luminâncias da cena para a gama do dispositivo de visualização.

O operador de *tone mapping* implementado no sistema desenvolvido segue as directivas do operador proposto por Larson *et al.* [Larson97]. A escolha deste operador baseou-se no facto de este respeitar a manutenção da consistência de visibilidade dos objectos no fotograma, e a correspondência entre a visualização da cena real e a experiência subjectiva que é a visualização do fotograma.

No entanto, optou-se por omitir as etapas finais do trabalho de Larson *et al*., dado o seu elevado custo computacional, que comprometeria o desempenho do sistema desenvolvido. Desta forma, as etapas que se desenvolveram foram as seguintes:

- Inicialização de variáveis;
- Cálculo do Histograma de Luminâncias;
- Cálculo da Distribuição Cumulativa;
- Ajuste Linear do Histograma de Luminâncias;
- Ajuste do Histograma Baseado na Sensibilidade Humana ao Contraste.

Na inicialização de variáveis procede-se à determinação dos valores máximos e mínimos das luminâncias da cena e dos valores correspondentes do dispositivo de visualização.

Em seguida procede-se à determinação do histograma de luminâncias. Para este contribui a informação de iluminação de cada *pixel* do fotograma, devidamente convertida para valores de luminância.

Na etapa seguinte, do cálculo do Histograma de Luminâncias, considera-se individualmente a informação de radiância de cada pixel do fotograma, a qual, uma vez convertida em luminância, permite determinar a célula correspondente do histograma.

Uma vez determinado o Histograma de Luminâncias, procede-se ao Cálculo da Distribuição Cumulativa. Com este intuito determina-se, para cada célula do histograma, o valor correspondente da frequência cumulativa [Larson97], que se utiliza na etapa de Equalização do Histograma. Desta última resulta a obtenção de uma primeira aproximação aos valores de luminância prontos a serem colocados no dispositivo de visualização. No entanto, a utilização directa dos resultados até agora obtidos produz imagens com regiões onde o contraste é demasiado elevado. Desta forma, para se assegurar que o contraste de qualquer região do fotograma não excede o resultado produzido por um operador linear, torna-se necessário proceder ao Ajuste Linear do Histograma de Luminâncias. Deste resulta a possibilidade de visualizar o fotograma com um aspecto mais natural, mas onde não são contempladas as características do sistema visual humano. De modo a introduzir estas características na informação dos fotogramas, executa-se a etapa de Ajuste do Histograma de Luminâncias Baseado na Sensibilidade Humana ao Contraste, onde se assegura que o contraste de qualquer região do fotograma não ultrapassa a resposta do operador de Ferwerda *et al.* [Ferwerda96].

Finalmente, é com base no Histograma de Luminâncias final que se procede-se à visualização dos resultados.

No contexto do sistema desenvolvido, o tratamento perceptual incide exclusivamente sobre os resultados visuais, imediatamente antes da sua visualização. No entanto, supõe-se ser desejável e vantajoso dar utilização aos resultados perceptuais, no controlo dos pedidos a efectuar ao motor de síntese de imagem, uma vez que certas regiões da cena podem sofrer variações de iluminação difíceis ou mesmo impossíveis de detectar pelo sistema visual humano.

6. AVALIAÇÃO DO SISTEMA

A avaliação do protótipo implementado baseia-se em testes efectuados em dois ambientes distintos. O primeiro ambiente, designado por sequencial, compõe-se de um computador Intel PENTIUM III Celeron 500 Mhz com 128 Kb de memória *cache* de nível 1 e 128Mb de memória principal. Os testes realizados neste ambiente pretendem demonstrar que o protótipo consegue funcionar satisfatoriamente mesmo quando os recursos computacionais disponíveis são escassos. Neste cenário, as componentes RENDERCACHE, de tratamento perceptual dos resultados visuais e de determinação de amostras, partilham a mesma unidade de processamento.

O segundo ambiente de teste, designado por paralelo, é composto por um conjunto de vinte e dois nós, a funcionar numa arquitectura paralela *Beowolf* [Becker95]. Neste ambiente, as componentes RENDERCACHE e de tratamento perceptual partilham um nó, enquanto que a componente de determinação de amostras utiliza os nós restantes. Nos dois ambientes de teste utilizou-se o sistema operativo Linux Red Hat 6.2, com o KDE como gestor de X11. O *software* que dá suporte à vertente paralela do protótipo desenvolvido é o PVM [Geist94] na sua versão 3.4.3. Nos testes realizados utilizou-se uma janela de visualização de fotogramas com dimensões de 150x150 *pixels*.

O modelo de cena utilizado nos testes efectuados foi um dos modelos do *software* RADIANCE, que possui uma complexidade geométrica de 13.100 polígonos, aproximadamente. Considera-se que este número de polígonos é adequado para os testes de visualização interactiva de imagens fotorrealistas, em virtude dos compromissos da cadência de fotogramas e na ausência de aceleração por *hardware*.

6.1 Critérios de Avaliação do Sistema

Os critérios de avaliação do protótipo nos ambientes de teste especificados são, designadamente, a cadência de fotogramas produzidos, o tempo de reacção do sistema aos comandos de navegação e a qualidade visual dos resultados. Considera-se que a cadência de fotogramas medida em número de fotogramas produzidos por segundo, é um dos critérios mais relevantes para a avaliação do desempenho do protótipo desenvolvido. Num cenário ideal, esta cadência deveria ser de pelo menos 25 a 30 fotogramas por segundo [Foley90].

A avaliação do tempo de resposta do sistema aos comandos de navegação é também importante, uma vez que se trata de um sistema interactivo, onde o atraso de reacção do sistema face às acções do utilizador deve ser muito reduzido. Neste contexto, espera-se que o tempo de resposta do sistema face aos comandos de navegação introduzidos pelo utilizador seja mínimo, permitindo ao

sistema reflectir, no mais curto intervalo de tempo, as alterações resultantes da mudança do ponto de observação da cena.

Finalmente, considera-se que a qualidade visual dos resultados é outro critério de avaliação importante, uma vez que influencia de modo crítico a percepção que o utilizador tem da cena em que está a navegar.

No que respeita à paralelização, considera-se que a avaliação da arquitectura paralela da componente de determinação de amostras possui critérios específicos de avaliação do desempenho, nomeadamente o *speedup* e a eficiência da arquitectura paralela [Hwang93]. A avaliação que se faz da arquitectura paralela é a que corresponde à avaliação de carga fixa, isto é, de Amdahl [Amdahl67], onde a dimensão do trabalho a realizar é constante e se faz variar a capacidade computacional. No entanto, existem outras variantes, por exemplo, de Gustafson [Gustafson88], ou de Sun & Ni [Sun93], que podem ser utilizadas, desde que se proceda a testes consistentes com estas avaliações. Os valores de *speedup* foram obtidos de acordo com o que se apresenta na Equação 1. Nesta equação, T_1 corresponde ao tempo que o sistema demora a executar um certo trabalho, quando dispõe de apenas uma unidade de processamento e T_n corresponde ao tempo que esse mesmo sistema demora a executar o mesmo trabalho, quando dispõe de *n* unidades de processamento.

$$Speedup = \frac{T_1}{T_n} \qquad (1)$$

6.2 Ambiente Sequencial de Testes

No ambiente sequencial de testes, a avaliação do protótipo é feita com base nos resultados obtidos, correspondentes ao primeiro minuto de visualização da cena e que correspondem à produção de 192 fotogramas.

A recolha dos tempos, efectuada ao longo de uma sessão de utilização do protótipo indica que a geração de um fotograma varia entre 127 e 748 mili-segundos, sendo o valor médio, ponderado, de 320 mili-segundos. Quando se activa o processamento perceptual dos resultados visuais, é ainda necessário acrescentar um valor constante de 30 mili-segundos.

Face aos valores apresentados, conclui-se que a cadência média de visualização é de 2.9 e 3.2 fotogramas por segundo, respectivamente, sem e com tratamento perceptual.

Embora se reconheça que estas cadências se encontram longe das ideais, de 25 a 30 fotogramas por segundo, salienta-se que o melhor resultado corresponde à gestão, por parte da componente de determinação de amostras, de uma complexidade geométrica de 41.920 polígonos por segundo (13.100 polígonos * 3.2 fotogramas). Enquanto que este número de polígonos é considerado irrisório quando se utilizam acelerações por *hardware*, corresponde a uma geometria com complexidade satisfatória, face aos objectivos de utilizar exclusivamente *software*, executado em arquitecturas computacionalmente banais.

Na Tabela 1 apresentam-se os tempos obtidos em cada etapa da componente RENDERCACHE do sistema.

Operação	Média (ms)	Média (%)
Inicialização	8.5	2.7
Reprojecção	187.8	60.2
Depth Culling	16.1	5.2
Interpolação	52.6	16.9
Amostragem	19.4	6.2
Envelhecimento da *cache*	27.6	8.8

Tabela 1: Resultados temporais e relativos de execução de tarefas da componente RENDERCACHE

Nesta tabela observa-se que a operação temporalmente mais dispendiosa é a etapa de reprojecção de amostras, que ocupa 187 mili-segundos. De facto, a complexidade computacional desta etapa é significativamente maior do que a de qualquer outra. Isto resulta da necessidade de executar operações de cálculo vectorial na reprojecção, ao invés das outras etapas, onde se utilizam operações aritméticas e lógicas simples. No entanto, destaca-se que, embora a reprojecção seja a etapa mais demorada, o tempo dispendido é bastante inferior àquele que se dispenderia com a utilização massiva de um motor de síntese de imagem baseado em geometria.

A componente responsável pela determinação de amostras determinou 27.841 amostras, com as quais se produziram 2.764.800 *pixels*. Isto significa que o protótipo gerou 192 fotogramas com apenas 1.007% de informação.

Durante a execução dos testes verificou-se que a capacidade de reacção do protótipo aos comandos do utilizador não apresenta qualquer atraso. Esta característica explica-se pela ausência de sincronismo entre a componente RENDERCACHE, responsável pela geração de fotogramas e a componente responsável pela determinação de amostras. Dado que a geração de um fotograma é feita unicamente com base na informação da *cache*, não necessita de aguardar pelos resultados da componente de determinação de amostras.

Nas experiências realizadas, constatou-se que o envio de vários comandos ao protótipo, temporalmente pouco espaçados, não tem repercussões na cadência de produção de fotogramas mas que afecta a qualidade dos resultados visuais, diminuindo-a significativamente. Por exemplo, um avanço muito rápido na cena desactualiza rapidamente as amostras da *cache*. Isto acontece porque as amostras ficam descontextualizadas, como resultado da baixa cadência de produção de amostras.

6.3 Ambiente Paralelo de Testes

Nos testes realizados neste ambiente verificou-se uma ligeira redução do tempo de determinação de fotogramas, aumentando-se ligeiramente a geração de fotogramas, de

192 para 287, durante o primeiro minuto de execução do sistema. Este aumento relaciona-se com o facto de as componentes RENDERCACHE e de tratamento perceptual estarem fisicamente separadas da componente de cálculo, ficando-lhes associada uma maior capacidade de processamento. Relativamente à cadência de fotogramas, conclui-se que o desempenho obtido, de 4.7 fotogramas por segundo (212 mili-segundos por fotograma), constitui uma pequena melhoria relativamente ao valor obtido no ambiente sequencial de teste, mas que ainda se encontra longe da cadência ideal [Foley90]. Porém, destaca-se que a mais valia deste ambiente não se encontra exclusivamente ao nível da cadência de geração de fotogramas, mas também ao nível da qualidade da informação com que de determinam dos resultados visuais.

Ao nível do reaproveitamento da informação, que constitui a essência da técnica RENDERCACHE, verificou-se que o conjunto de fotogramas foi produzido com apenas 3.5% da informação total: o sistema produziu 146.083 amostras para visualizar 4.132.800 *pixels*. Quando se compara com o ambiente sequencial de teste, verifica-se que a solicitação de amostras é 3.5 vezes superior neste ambiente, proporcionando a obtenção de resultados visuais com mais informação.

Relativamente ao tempo de reacção do sistema face aos comandos do utilizador, não se verifica qualquer atraso, embora, nas situações em que a cadência de comandos é elevada, o utilizador nota uma diminuição acentuada da qualidade visual dos fotogramas gerados.

Quanto à avaliação da arquitectura paralela da componente de determinação de amostras, elaboraram-se cenários experimentais, onde se fez variar o número de *workers*, nomeadamente, 1, 5, 10, 15 e 20, sendo cada *worker* atribuído a um processador diferente. Para cada um destes cenários elaboraram-se ainda sub-cenários, onde se fez variar a granularidade do trabalho atribuído a cada *worker*, designadamente, 10, 20, 25, 30 e 40 pedidos de amostras. Com base nos resultados temporais obtidos verificou-se que, com poucos *workers*, o desempenho do sistema se encontra condicionado pelo próprio número de *workers* existentes. Porém, quando se consideram os cenários de 15 e 20 *workers*, observa-se que o valor de granularidade se torna o elemento mais crítico para a obtenção dos melhores valores de *speedup*.

Relativamente à eficiência desta arquitectura, verificou-se que decai consideravelmente à medida que se adicionam mais *workers*, chegando ao valor de 46% quando existem 20 *workers* na componente de determinação de amostras. Constatou-se que o melhor valor de *speedup* da arquitectura paralela foi obtido para o cenário de 20 *workers*, com uma granularidade de 20 pedidos de amostra (Figura 2).

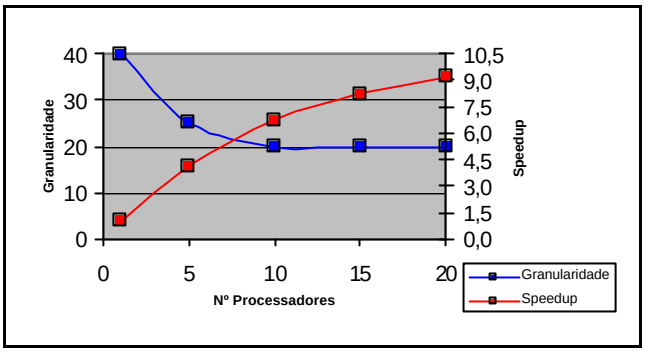


Figura 2: Melhores valores de granularidade encontrados e respectivos speedups, em função do nº de processadores

6.4 Qualidade dos Resultados Visuais

Em ambos os ambientes de teste verificou-se que a qualidade dos fotogramas depende sobretudo da consistência da informação existente na *cache* de amostras, relativamente ao contexto actual de visualização da cena e, por sua vez, esta depende da capacidade de determinar, atempadamente, amostras novas.

Constatou-se que acções como rotações e deslocamentos rápidos ou passagens através de superfícies opacas na cena (por exemplo paredes), degradam significativamente a qualidade dos fotogramas imediatamente produzidos.

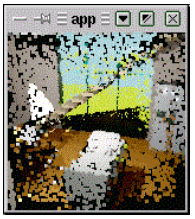


Figura 3: Resultado do deslocamento do observador para trás.

No fotograma apresentado na Figura 3, que corresponde à fase final de um movimento de recuo do observador na cena, observa-se que, ao invés da periferia, a região central do fotograma se encontra quase preenchida. Isto resulta da existência, na *cache*, das amostras necessárias ao preenchimento da região central. Ao contrário, a periferia do fotograma apresenta uma degradação substancial da qualidade visual, resultante da escassez ou mesmo inexistência de amostras produzidas para essa região. Nas situações em que o utilizador recua, a informação do centro dos fotogramas é, durante esse percurso, reprojectada, enquanto a informação da periferia é solicitada à componente de determinação de amostras e, portanto, determinada pela primeira vez. Comprovou-se ainda que a etapa de Interpolação é essencial para colmatar as falhas de informação e contribui para a recuperação rápida da qualidade dos fotogramas. Segundo Walter *et al*. [Walter99], cada fotograma torna-se perceptível a partir do momento em que 10% da *cache* de amostras se encontra preenchida com amostras relevantes. Embora este valor concreto não tenha sido provado, observou-se que, com uma percentagem muito pequena de amostras relevante, é possível obter

fotogramas totalmente preenchidos. Porém, salienta-se que, nestas situações em que a interpolação desempenha o papel preponderante na produção de fotogramas, o detalhe visual e o rigor dos resultados ficam severamente penalizados.

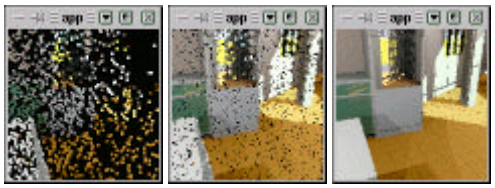


Figura 4: Limites da qualidade dos fotogramas produzidos pelo sistema desenvolvido.

De uma forma geral, dependendo do bom comportamento do utilizador na navegação pela cena, a qualidade média dos fotogramas é a que se apresenta na imagem central do conjunto da Figura 4.

7. CONCLUSÕES

A principal conclusão que se retira das experiências realizadas é que o sistema experimental RENDERCACHE possibilita atingir o objectivo da visualização interactiva de cenas virtuais, ainda que com baixa cadência de fotogramas, através de imagens de baixa resolução e quando os recursos computacionais não incluem acelerações por *hardware*.

Se o objectivo da aplicação da técnica RENDERCACHE consistir em aumentar a cadência de fotogramas, a solução deve passar pela paralelização das etapas de geração de fotogramas. Porém, quando o objectivo é o aumento da informação de base, isto é, do número de amostras para a geração de fotogramas com mais qualidade visual, a solução deve passar pela utilização de uma arquitectura paralela na componente de determinação de amostras. Desta forma, relaxa-se o elemento do sistema com a maior parcela de tempos consumidos. Verificou-se que a introdução do modelo computacional *farm* de computadores no protótipo desenvolvido permitiu apresentar resultados visuais com maior qualidade, uma vez que permitiu determinar em maior quantidade, e num intervalo de tempo mais reduzido, a informação geométrica e de iluminação solicitada. No entanto, constata-se que, a partir de um determinado número de unidades de processamento atribuídas ao Módulo de Cálculo, o desempenho desta arquitectura não corresponde ao esperado, face ao número de nós envolvidos. Isto resulta da incapacidade do *farmer* em gerir prontamente a atribuição de trabalho e a recolha de resultados dos *workers*.

Também foi possível concluir que a incapacidade de obtenção de uma cadência de fotogramas mais elevada, mesmo quando se está a usufruir da arquitectura paralela da componente de determinação de amostras, se deve ao desequilíbrio computacional que existe no sistema implementado, entre a elevada capacidade de determinação de informação geométrica e de iluminação e a baixa capacidade computacional associada à componente RENDERCACHE.

Do conjunto de etapas da componente RENDERCACHE, verifica-se que a reprojecção de amostras é a mais dispendiosa, mas que apresenta um saldo positivo quando comparada com o custo da determinação de informação nova a partir de um motor de síntese de imagem.

A amostragem do RENDERCACHE, orientada aos *pixels* do fotograma, baseada na idade de cada amostra e na consistência da sua informação, revelou ser adequada à amostragem das regiões mais críticas do fotograma e que contribuem para a maior convergência dos resultados visuais. Realmente, com apenas um conjunto mínimo de amostras determinadas, é possível a obtenção de resultados visuais satisfatórios.

Relativamente ao tratamento perceptual efectuado sobre os fotogramas produzidos pelo sistema verificou-se que o contraste dos resultados visuais é maximizado de acordo com as características do sistema visual humano, sem no entanto comprometer a visibilidade dos objectos ou a resposta visual. Daqui resulta a produção de fotogramas com qualidade visual satisfatória face aos objectivos propostos.

Verificou-se ainda que acções como deslocamentos ou rotações rápidas, isto é, acções que comprometem a coerência espaço-temporal da informação armazenada na *cache* tendem a reduzir a qualidade dos resultados visuais. Porém, assim que o observador se imobiliza, a qualidade dos fotogramas aumenta progressivamente. Nesta situação, os resultados visuais apresentam um grau de realismo que tende para o elevado, semelhante ao que se obtêm com a utilização directa de um motor de iluminação global para a produção de imagens estáticas.

8. PERSPECTIVAS DE TRABALHO FUTURO

Embora os resultados obtidos sejam satisfatórios, salienta-se que existem vários pontos neste trabalho que podem ser mais explorados, nomeadamente, o refinamento das heurísticas utilizadas na amostragem da componente RENDERCACHE, a inclusão dos resultados de perceptualidade na estratégia de amostragem, a paralelização das etapas de geração de fotogramas, a paralelização da componente RENDERCACHE e o desenvolvimento de um algoritmo mais robusto de visualização de fotogramas.

Apesar dos valores numéricos utilizados nas heurísticas da estratégia de amostragem, obtidos através de cenários experimentais, terem permitido uma calibração razoável da componente RENDERCACHE, supõe-se ser possível uma ainda melhor exploração. Duas vias podem desde já ser apresentadas: um melhor aproveitamento da informação temporal e a utilização das características dos materiais envolvidos na cena. Considera-se que o aperfeiçoamento destas heurísticas pode contribuir para melhorar a etapa de amostragem, resultando numa solicitação de pedidos

mais adaptada à componente de determinação de amostras.

Salienta-se ainda que, no sistema desenvolvido, o tratamento perceptual actua exclusivamente para efeitos de visualização dos fotogramas produzidos. Porém, a utilização dos resultados perceptuais na etapa de amostragem parece ser uma via de investigação importante, dado que pode evitar o cálculo de amostras em zonas onde o processamento perceptual não detecte alterações.

Finalmente, é desejável a inclusão de um algoritmo de visualização dos fotogramas mais robusto, capaz de preencher as regiões do fotograma que possuem informação esparsa. Como exemplo, refira-se o algoritmo de visualização baseado em polígonos de Voronoi, utilizado na técnica HOLODECK [Larson98].

9. REFERÊNCIAS

[Amdahl67] Amdahl, G.M., *Validity of Single-Processor Approach to Achieving Large-Scale Computing Capability*. Proceedings of AFIPS Conference, Vol. 30, 1967.

[Becker95] Becker, Donald J., Thomas Sterling, Daniel Savarese, John E. Dorband, Udaya A. Ranawak, Charles V. Packer, *Beowulf: A Parallel Workstation for Scientific Computation*, Proceedings, International Conference on Parallel Processing, 1995

[Chen93] Shenchang Eric Chen, Lance Williams.*View Interpolation for Image Synthesis*. Vol. 27, nº 2, 279-288, 1993.

[Ferwerda96] Ferwerda, Pattanaik, Shirley, Greenberg, *A Model of Visual Adaptation for Realistic Image Synthesis*. Proceedings of SIGGRAPH, 1996.

[Floyd76] R.W. Floyd, L. Steinberg. *An Adaptive Algorithm for Spatial Grayscale*. Proceedings of the Society for Information Display, Vol. 12, 75-77, 1976.

[Foley90] James D. Foley, Andries van Dam, Steven K. Feiner, John F. Hughes. *Computer Graphics, Principles and Practice. Second Edition*, Addison-Wesley, 721-733, 1990.

[Geist94] Al Geist, Adam Beguelin, Jack Dongarra, Weicheng Jiang Robert Manchek, Vaidy Sunderam. *PVM, A Users' Guide and Tutorial for Networked Parallel Computing*. MIT Press, Cambridge, Massachusetts, 1994.

[Gustafson88] Justafson, J.L. *Reevaluating Amdahl's Law*. Communications of ACM, Volume 31, nº 5, 1988.

[Harrington97] Steven Harrington. *Computer Graphics: a programming approach*. McGraw-Hill International Editions, 1987.

[Hearn94] Donald Hearn, M. Pauline Baker. *Computer Graphics. Second Edition*, Prentice Hall International Editions, 1994.

[Hwang93] Kay Hwang. *Advanced Computer Architecture: Parallelism, Scalability, Programmability*. McGraw-Hill, 1993.

[Jevans92] D.A.Jevans. *Object Space Temporal Coherence for Ray-Tracing*. Proceedings of Computer Graphics Interface, 176-183, 1992.

[Larson97] Gregory Ward Larson, Holly Rushmeyer, Christine Piatko. *A visibility Matching Tone Reproduction Operator for High Dynamic Range Scenes*. IEEE Transactions on Visualization and Computer Graphics, Vol. 3, nº 4, 1997.

[Larson98] Gregory Ward Larson. *The Holodeck: A Parallel Ray-caching Rendering System.* Proceedings of the Second Eurographics Workshop on Parallel Graphics and Visualization, Setembro 1998.

[Mark97] William R. Mark, Leonard McMillan, Gary Bishop. *Post-Rendering Warping*. Proceedings of Symposium on Interactive 3D Graphics, 1997, 7-16.

[McMillan97] Leonard McMillan Jr.. *An Image Based Approach to Three Dimensional Computer Graphics*. Chapel Hill,1997.

[Sun93] Sun, X., Ni, L.. *Scalable Problems and Memory-Bounded Speedup*. Journal of Parallel and Distributed Computing, nº 19,1993.

[Walter99] Bruce Walter, George Drettakis, Steven Parker. *Interactive Rendering using the RENDERCACHE*. Rendering techniques, Proceedings of the 10th Eurographics Workshop on Rendering, 1999.

[Ward94] G. J. Ward. *The RADIANCE Lighting Simulation and rendering system.* ACM SIGGRAPH, Conference Proceedings, 459-472, 1994.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Dynamic Algorithm Selection: a New Approach to the Real-Time Rendering of Complex Scenes Problem

Pedro Pires
IST & INESC-ID
Lisbon - Portugal
pedro.pires@acm.org

João Madeiras Pereira
IST & INESC-ID
Lisbon - Portugal
jap@inesc.pt

Abstract

This paper presents a novel approach to the real-time rendering of complex scenes problem. Up to the present date, a huge number of acceleration techniques have been proposed, although most are geared towards a specific kind of scene. Instead of using a single, or a fixed set of rendering acceleration techniques, we propose the use of several, and to select the best one based on the current viewpoint. Thus, dynamically adapting the rendering process to the contents of the scene, it is possible to take advantage of all these techniques when they are better suited, rendering scenes that would otherwise be too complex to display at interactive frame rates.

We describe a framework capable of achieving this purpose, consisting on a pre-processor and an interactive rendering engine. The framework is geared towards interactive applications were a complex and large scene has to be rendered at interactive frame rates. Finally, results and performance measures taken from our test implementation are given.

Keywords

Real-Time Rendering, Scene Management, Visibility Culling, Levels-of-Detail, Image-Based Acceleration Techniques.

1. INTRODUCTION

Real-time rendering of complex scenes is an old problem in computer graphics. Applications such as computer aided design (CAD), architectural visualization, flight simulators, virtual environments and computer games need to display scenes comprising many thousands, if not millions of polygons. Even today's hardware accelerated rendering pipelines cannot handle all the polygons that compose such scenes. Due to our everlasting quest for visual realism, the gap between processor power and scene complexity, is expected to maintain itself in the future. Thus, it is necessary to employ rendering acceleration techniques, i.e., techniques that reduce the number of polygons sent to the graphics pipeline, at each frame.

Over the years, several techniques that address this problem where proposed. They can be categorized into the following areas: (1) visibility determination techniques, and (2) geometric complexity reduction techniques. Area (1) encompasses topics such as view-frustum culling techniques and occlusion culling techniques, where as area (2) includes geometric detail reduction techniques and image-based acceleration techniques.

Most of these techniques have one thing in common: they are special purpose, i.e., they are built to explore characteristics of certain types of scenes. As an example, consider a city model: if the camera were located inside a building, in a room, it would be wise to use techniques that take advantage of the room's specific geometric arrangement – walls, which occlude other areas, and doors through which one can see – namely: portal based techniques [Airey90,Teller91,Teller92,Leubke95]. If the user were walking along a street, it would no longer be possible to use portal techniques, instead, more generic occlusion culling techniques should be used, such as shadow frusta [Coorg97,Hudson97] or hierarchical occlusion maps [Zhang97, Zhang98]. If the user where flying high above the city, it would be useless to use occlusion culling techniques, instead detail reduction techniques should be used, for most of the city would be visible.

As shown by this example, the best techniques to use depend on where the user is, and where he is looking at. In this paper we propose a new approach to the real-time rendering of complex scenes problem: we propose to select the set of rendering acceleration techniques based on the current viewpoint and view-direction and we present a framework capable of achieving this purpose.

This paper is organized as follows: in section 2, we review the state of the art in rendering acceleration techniques, specifically: on visibility culling, scene simplification and image-based acceleration techniques; in section 3 we present our general framework; on section 4 we provide an overview of our test

implementation; on section 5 the results are introduced; on section 6 we present our conclusions; and on section 7 we outline future research directions.

2. PREVIOUS WORK

Rendering scenes as complex and as fast as possible has always been a major goal in computer graphics. Over the time, many different techniques and rendering algorithms where proposed. Today, rendering pipelines are implemented in hardware, and are standardized around the z-buffer algorithm. Thus research focus has shifted towards techniques that try to reduce the number of polygons passed to the graphics pipeline. Here, apart from a small exception, we will concentrate on such techniques only, and not in rendering techniques that can no longer be used.

Specifically, we will cover visibility determination techniques, including view-frustum culling and occlusion culling techniques, and geometric complexity reduction techniques, including geometric detail reduction techniques, and image-based acceleration techniques.

2.1 Visibility Determination

2.1.1 View-Frustum Culling

View-frustum culling techniques try to quickly determine which objects are inside the current view-frustum and pass only those to the graphics pipeline. Either a bounding volume hierarchy, or a spatial partitioning hierarchy, is used, and each volume is hierarchically tested for containment within the frustum planes [Moller96].

This is a simple and effective test, which is performed by most applications, and implemented in any retained mode API.

2.1.2 Occlusion Culling

Occlusion culling techniques try to eliminate objects (occludees) that are hidden behind other objects (occluders).

Greene et. al. proposed the use of a hierarchical z-buffer [Greene93]. A scene in augmented with a hierarchical spatial partitioning tree and it is rendered in front-to-back order. Polygons will be tested against pyramid of z-buffers, and for densely occluded scenes. The last, and thus the furthest, will be quickly rejected. Until recently this technique was limited to software rendering, but the latest graphic accelerators already implement hierarchical z-buffers in hardware.

Zhang et. al. proposed another object space occlusion culling technique – Hierarchical Occlusion Maps (HOMs) [Zhang97, Zhang98]. For a given viewpoint, several occluders are rendered into an image, using hardware acceleration. Several lower resolutions of this image are created, also using the standard hardware, in order to form an image pyramid. The bounding box of the remaining objects are projected onto the screen and checked against the image pyramid. Any object whose bounding box is covered by the projection of the occluders is hidden behind them.

Object space occlusion techniques were also proposed [Coorg97,Hudson97]. The basic idea is that if the viewpoint is considered to be a light source, any object in the shadow of an occluder is invisible from the same viewpoint. A shadow frusta, i.e., a set of planes bounding this area can be built and each object's bounding box is tested against these planes.

Another set of techniques, instead of trying to determine what is *invisible* from a *point* in space, try to determine what is *visible* from a given *volume* in space – its Potentially Visible Set (PVS).

For static interior scenes, portals and cells are the preferred solution. The terms portals and cells where first coined by Jones [Jones71] in the context of hidden line removal, back in 1971. In his algorithm, models were manually subdivided into convex cells and convex portals. Rendering begins with the walls and portals of the cell containing the user. As each portal is drawn, the cell on the opposite side is recursively rendered and its contents clipped against its portal. Later, Airey [Airey90] focused on the problem of determining cell-to-cell visibility, proposing several solutions, including point sampling and shadow volumes. Teller [Teller91,Teller92] took the concept further and found an analytic solution to the portal-to-portal visibility problem based on linear programming techniques. Luebke and Georges [Leubke95] suggested dynamically calculating the PVSs at run-time.

For static exterior scenes techniques were proposed that calculate the PVS for a volume in space. They are conservative volume techniques since they calculate a as tight as possible superset of the objects which can be seen from any point inside a given volume in space [Cohen-Or98,Durand00,Durand99,Schaufler00].

2.2 Geometric Complexity Reduction

2.2.1 Geometry-Based Techniques

Geometry based techniques aim to reduce the number of polygons used to represent an object, based on certain criteria: typically, its distance to the viewpoint. The reason behind this is that objects further away from the viewpoint occupy a smaller screen space area, and thus can be represented with less detail.

Static levels of detail, first introduced by Clark [Clark74], use several representations of the same object at different resolutions and switch between them based on the distance to the user. Funkhouser and Séquin studied the problem of selecting the set of object representations that maximizes the visual benefit while maintaining a target frame rate [Funkhouser93].

Progressive meshes, introduced by Hoppe [Hoppe96,Hoppe98] represent an object as a coarse base mesh, plus a record of each operations required to transform it into the full resolution mesh. These operations are invertible, and each adds a new vertex. Thus, it is possible to progressively increase or decrease detail, depending on the distance to the viewpoint. The

addition and removal of vertexes can be smoothly animated.

Techniques capable of selectively refining the models, where also proposed [Xia96,Xia97,Hoppe97,Luebke97]. For large models, it is desirable to increase details in the parts that are closer to the viewpoint, and not the object as a whole.

Special purpose algorithms, capable of reducing the detail in a view dependent way were also proposed for height field (terrain) data [Lindstrom96,Duchaineau97,Hoppe98b].

Another possibility is the use of parametric surfaces to represent scene content [Kumar96,Kumar97]. They can be tessellated on the fly, with more or less detail, depending on their distance to the user.

2.2.2 Image-Based Acceleration Techniques

Image-based acceleration techniques, try to replace portions of a scene with images. An image can represent as many objects as necessary, and its rendering time depends only on its resolution. On the other hand, it can only represent static content and it is only valid when seen from the point where it was created.

Marciel and Shirley where the first to use images in this context. They replaced objects with pre-rendered images (a single polygon texture mapped with an image), and coined the name impostors for this new primitive [Maciel95].

Shade et.al. [Shade96] and Schaufler and Stürzlinger [Schaufler96] introduced the idea of image caches. Impostors are generated on the fly to replace parts of the scene, and remain valid for a couple of frames. As the viewpoint moves, and the error exceeds a user specified number of pixels, the impostor(s) will be invalidated and regenerated.

Techniques to extend the life of impostors were proposed by Scilion et. al.[Scillion97] and Decoret et. al. [Decoret99].

Images have also been used in portal environments. Aliaga and Lastra introduced the concept of portal textures [Aliaga97]. They replaced portals with pre-rendered images of the cells behind it and showed how to smoothly transition from images to geometric based representations of a portal, when the user approaches it. They also introduced a method to automatically place images in order to guarantee a minimum frame rate [Aliaga99].

2.3 Discussion

Most of the above-mentioned techniques are special purpose ones, i.e., they are best suited for specific kinds of scenes. Some can only handle static scenes; others are suited for both static and dynamic ones. Some require that the scene be represented as a set of disjoint objects; others do not impose any particular structure. Some are only usable for interior scenes; others are geared towards exterior ones. Some offer advantages when used with geometry located close to the viewpoint, others are preferred for objects located far away. Given these criteria, Table 1 (located at the end of this document) summarizes the requirements/appropriateness of each above-mentioned technique, and groups them into categories.

TECHNIQUES				Dynamics	Structure	Organization	Distance
Visibility	Point Visibility	Image Space Techniques	Hierarchical z-buffer	DYN/STA	IN/OUT	UNSTR	
			HOMs	DYN/STA	IN/OUT	STR	
		Object Space Techniques	Shadow Frusta	DYN/STA	IN/OUT	STR	
	Conservative Volume Visibility	Interiors	BSP+PVS	STA	IN	UNSTR	
			Portals	STA	IN	STR	
		Exteriors	Octrees+PVS	STA	OUT	STR	
Detail	Geometry Based Techniques	Static LODs		DYN/STA	IN/OUT	STR	NEAR
		View-Dependent Progressive Meshes		DYN/STA	IN/OUT	STR	NEAR
		View-Independent Progressive Meshes		DYN/STA	IN/OUT	STR	NEAR
		Parametric Surfaces		DYN/STA	IN/OUT	STR	NEAR
		Terrain		DYN/STA	OUT	STR	
	Image Based Techniques	Impostors		STA	IN/OUT	UNSTR	FAR
		Image Caches		STA	IN/OUT	UNSTR	FAR
		Portal Textures		STA	IN	UNSTR	FAR
		Pure Image Based		STA	IN/OUT	UNSTR	FAR

Table 1 – Categorization of existing Rendering Acceleration Techniques and situations where they should be used. DYN stands for dynamic scenes, STA for static scenes, IN for interior scenes, OUT for exterior scenes, UNSTR for unstructured scenes (poly soup), STR for structured scenes (objects/meshes), NEAR for scene content close to the viewpoint and FAR for scene content far from the viewpoint.

Point visibility occlusion techniques are suitable for both interior/exterior and static/dynamic scene content. Among these, Shadow Frusta and Hierarchical Occlusion Maps (HOMs) handle static occluders better, but occludees can still be dynamic. They also require that the scene be organized as a set of objects that have bounding volumes assigned. Hierarchical Z-buffer imposes no organization on the scene.

Conservative volume visibility evaluation techniques are done off-line, thus are only applicable to static scene content. Portals and BSP trees are special purpose and suitable only to interior scenes. While Portals require the scene to be subdivided into cells and portals, BSP trees impose no such structure, since they are able to create it from polygon soup models. Octree-based conservative visibility evaluation techniques are suited for exterior scene content.

The distinction between complexity reduction techniques is cleaner. With the exception of portal textures, which are more special purpose, they are all suited to both interior and exterior scenes. Geometric detail techniques require the scene to be organized as a set of objects some with special purpose representations, while image-based techniques do not impose any special structure on object representation. Geometric detail techniques are suited for both static and dynamic objects, while image-based techniques still have difficulty to cope with dynamic scenes, be it moving objects or changing illumination conditions. Images are also better suited to represent scene content that is far from the viewer, given that the error induced as the viewpoint moves away from the

viewpoint from where it was originally created is smaller, if the image is placed far away from the user. Geometric detail techniques are usually better to represent scene content closer to the user, since that after some point in the distance an object will contribute very little to the user perception of the scene and its lowest detail can be used, without having to change it further.

Moreover, it is useless to apply occlusion-culling techniques if a substantial part of the scene is not occluded from a given viewpoint. Similarly, it is useless to employ complexity reduction techniques, if there are not enough polygons visible.

As a conclusion, it is safe to say that: *the best acceleration techniques are dependent on the chosen viewpoint and view-direction*.

3. SELECTING THE BEST RENDERING ACCELERATION ALGORITHM

In order to select the best rendering acceleration techniques, it is necessary to analyze the scene and determine which of the available techniques are best suited for the current viewpoint. This process takes its time, which should be minimized. Specifically, the time to select an algorithm α_i, amortized by the number of frames it remains valid, plus the time to render the scene using it, should be inferior to the time to render the scene with a single algorithm β.

$$\frac{t_{select}(\alpha_i)}{valid\_frames} + t_{render}(\alpha_i) < t_{render}(\beta)$$

Another important criterion is that it is undesirable to switch between algorithms every frame. In other words, spatial coherency hints that the best techniques for a given viewpoint remain the same over a given neighborhood thereof.

To keep the selection time low, we propose to analyze the scene off-line, and store information that can be used at runtime to select the appropriate algorithm, once the viewpoint is known.

To explore spatial coherency, we suggest augmenting the scene with a spatial subdivision hierarchy, and marking each node with the techniques to use, when the viewpoint is located within the volume it represents. This tree should be kept separate from the usual view-frustum culling hierarchy, since it fits a different purpose: the former groups objects based on their spatial location, and the later categorizes zones of space. Even if the same type of data structure is used, each will benefit from different subdivision criteria. Possible choices for such a tree are an octree, a k-dtree, a bsp-tree or a similar data structure. The advantages and disadvantages of each are well documented elsewhere.

Since the best techniques are liable to change with the view-direction, it is necessary to consider several view-directions per node. Ignoring the camera tilt, its orientation can be expressed by two angles: θ (azimuth) and φ (elevation). Dividing the space of all possible orientations into n discrete intervals of the form $[\theta_i, \theta_{i+1}[\times [\varphi_i, \varphi_{i+1}[$, it is possible to determine the best techniques when the view-direction is inside each interval. This information can be stored at each node using a bi-dimensional table, indexed using the current view direction (θ, φ).

Such representation takes advantage of space coherency since that the contents of a scene, as seen from two points close by in space, using more or less the same view-direction, are likely to be very similar. Thus the best techniques are also likely to be the same in both situations.

Summarizing: During the preprocessing step the scene is divided into a set of volumes, using a spatial subdivision hierarchy. Each volume is analyzed in order to determine the best techniques to use when the viewpoint is therein, and looking in a given direction. This information is written in the corresponding tree node. Additional preprocessing required for each particular algorithm is done and stored at this time. At run time, once the viewpoint is known, the spatial subdivision hierarchy is used to rapidly access the pre-computed information and to determine the techniques to use.

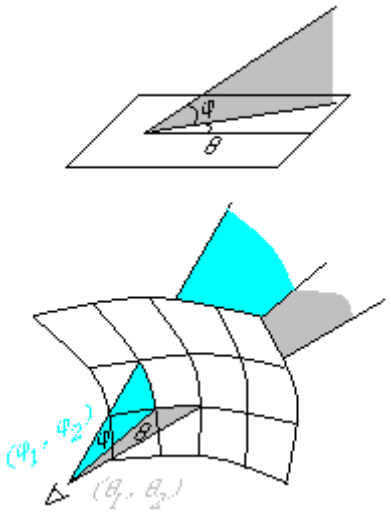


Figure 1 - Dividing the space of all possible view-directions into discrete intervals

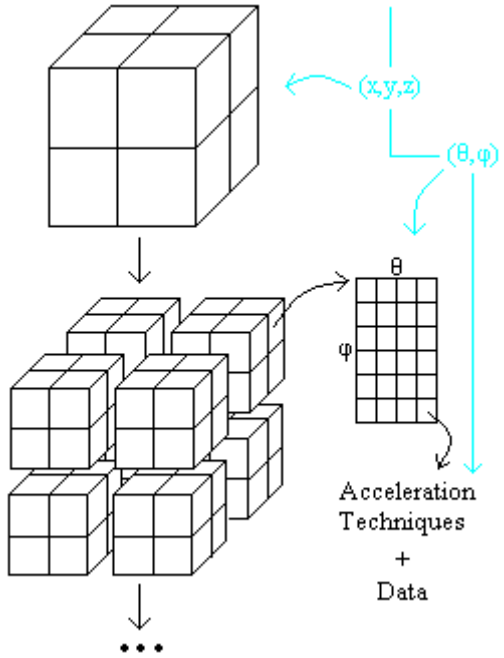


Figure 2 – The scene is portioned using a Hierarchical Space Partition Tree (an octree in this example) and each node stores a table with the techniques to use for each view-direction. At runtime the viewpoint (x,y,z) is used to determine the current tree node and the view-direction (θ,φ) is used to index the corresponding table, obtaining the acceleration techniques and associated data.

In the next two sections both these steps will be addressed.

3.1 Scene Preprocessing

The preprocessor's goal is to build the hierarchical spatial partition tree, analyze the scene as seen from each node and view-direction, determine the techniques that are better suited in each case, store this information at each node and write the hole data structure to disk.

In order to determine if acceleration techniques are required, it is necessary to measure how complex is a scene, as seen from a given viewpoint. Then there are two characteristics that have to be analyzed for each viewpoint and view-direction: first, in order to justify the use of occlusion techniques, it is necessary to determine the set of objects that can be seen, and the occlusion relationships between these objects. Second, to determine if detail reduction techniques are required, it is necessary to determine how complex is this set of objects.

3.1.1 Measuring Geometric Complexity

How complex a scene is, is a relative concept. A scene can be too complex for one machine, and fairly easy to render using another. The simplest way to determine the time it takes to render a given object is to simply render it and measure. Since this is done as a preprocessing step, it means that the scene has to be preprocessed in the same type of workstation that will be used for the visualization. This may not always be desirable. Thus, it is necessary to find an abstract and adjustable model of each machine's graphics pipeline. Back in 1993, Funkhouser and Séquin [Funkhouser93] proposed one such model. Surprisingly it is still valid today.

The cost of rendering an object or a set of polygons, i.e., the time it takes to do it, increases with the number of *per-vertex* and *per-fragment operations*, the hardware has to perform. Per-vertex operations include moving the vertexes to the GPU, through the system's bus, vertex transform, lighting and clipping, or whatever vertex program is specified in today's programmable graphic pipelines. Per-fragment operations include rasterization, texture mapping, depth compares, stencil operations, pixel-shader operations, and so on.

Funkhouser and Séquin argued, and showed experimentally, that the per-vertex cost and the per-pixel cost change linearly, with machine dependent proportionality constants α_v and α_p, with the number of triangles and the number of pixels drawn, respectively. Both these proportional factors can be taken from manufacturer's data sheets or evaluated experimentally on each platform. The cost of rendering an object, or a set of triangles is the higher of these costs, since these operations run in parallel, in a pipeline, whose weakest link determines the overall cost.

$$\max\{\alpha_v \times num\_vertexes, \alpha_p \times num\_pixels\}$$

Calculating the number of pixels occupied by each triangle, is time consuming and unnecessary. Instead the number of pixels occupied by an object can be approximated by the number of pixels occupied by the projection of its bounding box on the screen. To make calculations even simpler, it can be approximated by the number of pixels occupied by the 2D bounding box of the screen projection of the object's 3D bounding box.

3.1.2 Selecting Visibility Techniques

Since interior areas are usually modeled separately, we assume that they are marked as such in the scene model. Thus, it is not necessary to determine which areas are interiors and which are exteriors automatically. Interior areas are represented as cells with portals which link to other cells, or to the tree structure (for example, an octree) used to represent the outside areas (windows or doors can lead outside or to other rooms). Portals leading into inside areas will be represented in the nodes of this tree structure, corresponding to the volume where they are located.

Thus, if an implementation wishes to use portal based techniques, its hierarchical structure will be a combination of a cell graph, and an octree. The only requirement is that given a point in space, it must be possible to determine in which cell it is or in which octree node it is located.

A cell will assume the same role as an octree node. It will also have a table, which is indexed using the current view direction. If justifiable, a cell can be further divided (for instance, complex factory models are known to exhibit

large, and highly intricate interior rooms, with pipes and machinery occluding each other). For all purposes, a portal is treated as any other object, except that when it is rendered, the cell behind it will be displayed instead.

During the preprocessing step this hierarchy is built, using the usual techniques. If memory constraints permit it, the PVS for the interior cells can be computed and stored at this time [Airey90,Teller91,Teller92]. Alternatively visibility can be evaluated at run-time using Leubke and Georges' technique [Leubke95].

The first thing to do at each viewpoint is to cull all the objects outside the view-frustum, and determine if the remaining ones can be rendered under the allotted frame time. If they can, it is not necessary to use any special techniques. If they cannot, it is necessary to determine if there are objects that can be used as occluders. Object space occlusion culling techniques require a reduced number of large close by occluders. On the other hand, Zhang's hierarchical occlusion maps can handle several smaller occluders. More specific measures on how to select ocluders for each technique are given in [Coorg97,Hudson97] and [Zhang97, Zhang98] respectively. Using our technique, the accelerations techniques are chosen based on the number and type of possible occluders, and not the other way around.

Whatever technique gets chosen, the set of occluders to use for this viewpoint and direction are stored at the corresponding node.

3.1.3 Selecting Complexity Reduction Techniques

Given a viewpoint, a view-direction and the set of visible objects, the preprocessor determines if they can be rendered in the allotted frame time, without complexity reduction techniques. If they cannot, it tries to use different techniques for different parts of the scene.

The relevant criterion here is proximity: Image-based techniques are better suited for representing far away portions of the scene, since they can group lots of small objects into a single image, and the further away the image is, the further one can move away from the point where it was created without noticing errors. On the other hand, after some point in the distance, the object will be too small and it can be rendered using the lowest detail possible. Thus geometric detail techniques are better suited for objects that are closer to the viewpoint.

Furthermore, since most image-based techniques can induce visual artifacts, we have chosen to try geometric detail techniques first and image-based ones only if necessary.

The visible objects are placed in a list sorted by distance to the viewpoint. Two passes are made through this list, starting from the further object to the closest one. In the first, each object is selected to be rendered using a geometric detail reduction approach. In the second each object is selected to be rendered using an image-based approach. Note that in the later case, all selected objects can be represented using the same image-based primitive, since images can effectively combine several objects. After each selection, the time to render the scene is evaluated, and this process stops as soon as this time is under the required maximum frame time.

Note that a portal is treated as any other object. It can be displayed rendering the geometry of the cell behind it, or using a portal texture [Aliaga97,Aliaga99].

The required images will be stored at the corresponding tree node table, and cannot be reused at other nodes since they are view-dependent.

Data required for the geometric detail reduction techniques, will be stored in a common pool and a pointer to the relevant information will be kept at the node, since this data can usually be shared by other nodes (for instance, if it is decided to use a progressive mesh to render an object, the same mesh will probably be used for many other positions and orientations.

3.2 Interactive Rendering

Once the preprocessing is done, rendering the scene is a simple matter. The user's position in space is used to index the spatial structure and the node corresponding to the volume where the user is. The view-direction is used to index the scene and determine the techniques to use.

Once again, if a portal is visible it will be treated as any other object, and will be rendered with the technique assigned during the preprocessing step.

4. TEST IMPLEMENTATION

In order to validate our ideas, we have developed a prototype preprocessor and a rendering engine. Since it would be impossible to implement all the techniques proposed in the literature in any reasonable amount of time, we decided to focus on a subset of thereof. In addition we chose the techniques that would be faster to implement.

We chose to use a quadtree to index exterior scene content and portals to represent interiors. For simplicity we do not calculate the PVS for each cell, instead we use Leubke and Georges' technique [Leubke95]. The techniques to use are kept in a separate structure, a 3D grid, since an octree would use more memory.

We use hierarchical view-frustum culling to eliminate objects exterior to the view volume and shadow frusta to handle occlusions. As a geometric complexity reduction technique, we used a static level of detail approach.

At this time, we have not experimented with image-based acceleration techniques. We plan to use simple impostors for the further objects when necessary. As we are not considering warping techniques, [McMillan95] there will surely be artifacts when switching between impostors, but that is a characteristic of the chosen acceleration technique, and it does not invalidate the ideas presented at this paper, namely the possibility of preprocessing a scene in order to extract enough information to permit a fast real-time selection of the best algorithm for any viewpoint and view-direction.

5. RESULTS

We tested our approach using a model of our university's campus and surrounding area. It contains about 75k polygons spread by 3500 objects. In order to increase its complexity, it was instanced 49 times (in a 7x7 square), adding up to 3,6M polygons.

To reduce the number of possible viewpoints, the user was only allowed to fly within the campus up to a height of 70m above the ground, at which time he can view the whole model.

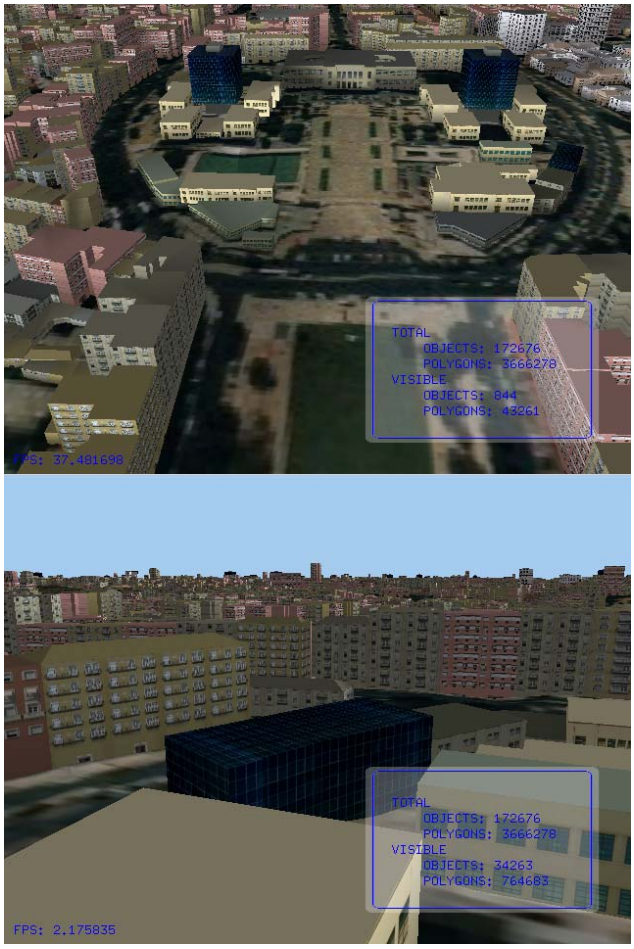


Figure 3 - Two snapshots from the test model: one showing the campus and the other showing the surrounding area as seen from inside the campus. The model was provided by SAE/IST[1] and is a part of the Virtual Lisbon project.

The system was implemented entirely in C++ and OpenGL. The testes were run on a Pentium III 450MHz with a GeForce256 graphics accelerator and the frame rate was locked to the monitor refresh rate, 75Hz, yielding a maximum of 75fps. Without any acceleration techniques, the model could only be rendered at 0,6fps in this system. From within the campus, using only view frustum culling, the worst-case views were rendered at 2fps.

[1] Environment and Energy Group at Instituto Superior Técnico.

Depending on the user position and orientation, frame rate varied significantly. In interior areas 30fps to 75fps where easily achieved. In exterior areas, close to ground level, the results depended on how well the preprocessor selected the occluders. In the presence of large objects close to the camera, the Shadow Frusta algorithm was activated and we achieved frame rates between 10fps and 30fps (depending on the number of occluders and how much of the scene they occluded). In open areas, with many small occluders close to the horizon and when flying above the campus, these were not the most appropriate algorithms, thus frame rate plummeted to 2fps. It is our belief that if we use Hierarchical Occlusion Maps and/or image based techniques, speedups will be possible in these areas, although we need to implement them, and perform further tests in order to make that claim.

6. CONCLUSIONS

We have presented a novel approach to the real-time rendering of complex scenes problem. We proposed to select the best rendering techniques for each viewpoint and view-direction. In order to minimize the time to select new algorithms, we analyze the scene offline and store information to permit a rapid selection at run-time.

We described a general framework capable of achieving this, composed of a preprocessor and an interactive rendering engine.

We provided an overview of the state of the art in real time rendering acceleration techniques, and identified criteria that permit the selection of each given a specific type of scene.

Given the encompassing nature of this work, we chose a small subset of these techniques, and implemented a prototype, which, although in its initial stages, validated our ideas: in situations for which appropriate algorithms were available, interactive frame rates where achieved. For other situations, further tests are needed.

We have discovered that, as with all acceleration techniques, there are tradeoffs involved. In this case we trade space for speed.

7. FUTURE WORK

The first issue we would like to explore is the use of cache management techniques to mitigate the memory requirements problem. This subject was already studied by Funkhouser [Funkhouser96], in the context of portal based scenes. He predicted the cells where the user could be in the next frames and tried to load their PVSs from persistent storage, before the user reached them. Thus, the required data will always be in memory and the system will not block, i.e., will not stop generating images, due to cache misses. Their ideas could be adapted to handle our structure.

Another issue to explore is how many spatial intervals to consider. At this time the 3D grid divides space evenly. The space of possible directions is also divided into a pre-specified number of intervals. It would be interesting to use an octree and automatically decide when to stop subdividing the octree and what intervals to consider for

the space of possible orientations, in order to reduce the amount of positions to consider and still obtain the highest frame possible. A possible approach is to use brute force. We can start with a very fine interval resolution, preprocess the scene and then merge intervals where there are no changes in the used techniques.

8. ACKNOWLEDGEMENTS

The main test model was provided by the Environment and Energy Group at Instituto Superior Técnico (SAE/IST) (see http://visualis.ist.utl.pt). It is a small part of the Virtual Lisbon Project, which aims to build and display a 3D representation of the city of Lisbon. We would like to thank the team behind this project, for allowing us to use a part of their model.

9. REFERENCES

[Airey90] John Airey. Increasing Update Rates in the Building Walkthrough System with Automatic Model-Space Subdivision and Potentially Visible Set Calculations. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, TR#90-027, 1990.

[Aliaga97] Daniel G. Aliaga and Anselmo A. Lastra. Architectural Walkthroughs Using Portal Textures, IEEE Visualization '97, pp. 355-362 (November 1997). IEEE. Edited by Roni Yagel and Hans Hagen. ISBN 0-58113-011-2.

[Aliaga99] Daniel G. Aliaga and Anselmo Lastra. Automatic Image Placement to Provide a Guaranteed Frame Rate, Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series, pp. 307-316 (August 1999, Los Angeles, California). Addison Wesley Longman. Edited by Alyn Rockwood. ISBN 0-20148-560-5.

[Cohen-Or98] Daniel Cohen-Or, Gadi Fibich, and Dan Halperinand Eyal Zadicario. Conservative visibility and strong occlusion for viewspace partitioning of densely occluded scenes. Computer Graphics Forum, 17(3):243–254, 1998. ISSN 1067-7055.

[Coorg97] S. Coorg and S. Teller. Real Time Occlusion Culling for Models with Large Occluders. In Simposium on Interactive 3D Graphics, pages 83–90, April 1997.

[Decoret99] Xavier Decoret, François Sillion, Gernot Schau er, and Julie Dorsey. Multi-layered impostors for accelerated rendering. Computer Graphics Forum, 18(3):61–73, September 1999. ISSN 1067-7055.

[Durand99] Frédo Durand. 3D Visibility: analytical study and applications. PhD thesis, Université Joseph Fourier, Grenoble I, July 1999.

[Durand00] Frédo Durand, George Drettakis, Joëlle Thollot, and Claude Puech. Conservative visibility preprocessing using extended projections. Proceedings of SIGGRAPH 2000, pages 239– 248, July 2000. ISBN 1-58113-208-5.

[Duchaineau97] Mark A. Duchaineau, Murray Wolinsky, David E. Sigeti, Mark C. Miller, Charles Aldrich, and Mark B. Mineev-Weinstein. Roaming terrain: Real-time optimally adapting meshes. IEEE Visualization ’97, pages 81–88, November 1997. ISBN 0-58113-011-2.

[Funkhouser93] Thomas A. Funkhouser and Carlo H. Séquin. Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. *Proceedings of SIGGRAPH 93*, pages 247–254, August 1993. ISBN 0-201-58889-7.

[Funkhouser96] Thomas A. Funkhouser. Database management for interactive display of large architectural models. Graphics Interface ’96, pages 1–8, May 1996. ISBN 0-9695338-5-3.

[Greene93] N. Greene, M. Kass, and G. Miller. Hierarchical Z-Buffer Visibility. In Computer Graphics (SIGGRAPH 1993 Proceedings), pages 231–238, August 1993.

[Hoppe96] [Hop96] Hugues Hoppe. Progressive meshes. Proceedings of SIGGRAPH 96, pages 99–108, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[Hoppe97] Hugues Hoppe. View-dependent refinement of progressive meshes. Proceedings of SIG-GRAPH 97, pages 189–198, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[Hoppe98a] Hugues Hoppe. Efficient implementation of progressive meshes. Computers & Graphics, 22(1):27–36, February 1998. ISSN 0097-8493.

[Hoppe98b] Hugues H. Hoppe. Smooth view-dependent level-of-detail control and its application to terrain rendering. IEEE Visualization ’98, pages 35–42, October 1998. ISBN 0-8186-9176-X.

[Hudson97] T. Hudson, D. Manocha, J. Cohen, M. Lin, K. Hoff, and H. Zhang. Accelerated Occlusion Culling Using Shadow Frusta. In Symposium on Interactive 3D Graphics, pages 1–10, June 1997.

[Jones71] C.B. Jones. A New Approach to the ’Hidden Line’ problem. The Computer Journal, 14(3):232, August 1971.

[Kumar96] Subodh Kumar and Dinesh Manocha. Hierarcical visibility culling for spline models. Graphics Interface ’96, pages 142–150, May 1996. ISBN 0-9695338-5-3.

[Kumar97] Subodh Kumar, Dinesh Manocha, Hansong Zhang, and III Kenneth E. Hoff. Accelerated walkthrough of large spline models. 1997 Symposium on Interactive 3D Graphics, pages 91–102, April 1997. ISBN 0-89791-884-3.

[Leubke95] D. Luebke and C. Georges. Portals and Mirrors: Simple, Fast Evaluation of Potentially Visible Sets. In Symposium on Interactive 3D Graphics, pages 105–106 and 212, April 1995.

[Luebke97] David Luebke and Carl Erikson. View-dependent simplification of arbitrary polygonal environments. Proceedings of SIGGRAPH 97, pages

199–208, August 1997. ISBN 0-89791-896-7. Held in Los Angeles, California.

[Lindstrom96] Peter Lindstrom, David Koller, William Ribarsky, Larry F. Hughes, Nick Faust, and Gre-gory Turner. Real-time, continuous level of detail rendering of height fields. Proceedings of SIGGRAPH 96, pages 109–118, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[McMillan95] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. Proceedings of SIGGRAPH 95, pages 39–46, August 1995. ISBN 0-201-84776-0. Held in Los Angeles, California.

[Moller96] Tomas Moller and Eric Haines. Real Time Rendering. A K Peters, 1996.

[Maciel95] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. 1995 Symposium on Interactive 3D Graphics, pages 95–102, April 1995. ISBN 0-89791-736-7.

[Shade96] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose, and John Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. Proceedings of SIGGRAPH 96, pages 75–82, August 1996. ISBN 0-201-94800-1. Held in New Orleans, Louisiana.

[Schaufler96] Gernot Schaufler and Wolfgang Stürzlinger. A three dimensional image cache for virtual reality. Computer Graphics Forum, 15(3):227–236, August 1996. ISSN 1067-7055.

[Schaufler00] Gernot Schaufler, Julie Dorsey, Xavier Decoret, and Fran¸cois X. Sillion. Conservative volumetric visibility with occluder fusion. Proceedings of SIGGRAPH 2000, pages 229–238, July 2000. ISBN 1-58113-208-5.

[Sillion97] François X. Sillion, G. Drettakis, and B. Bodelet. Efficient impostor manipulation for real-time visualization of urban scenery. Computer Graphics Forum, 16(3):207–218, August 1997. ISSN 1067-7055.

[Teller91] S. Teller and C. Séquin. Visibility preprocessing for interactive walkthroughs. In Computer Graphics (SIGGRAPH 1991 Proceedings), pages 61–70, August 1991.

[Teller92] Seth Teller. Visibility Computation in Densely Occluded Polyhedral Environments. PhD thesis, Department of Computer Science, University of California Berkeley, TR#92/708, 1992.

[Xia97] Julie C. Xia, Jihad El-Sana, and Amitabh Varshney. Adaptive real-time level-of-detail-based rendering for polygonal models. IEEE Transactions on Visualization and Computer Graphics, 3(2), April - June 1997. ISSN 1077-2626.

[Xia96] Julie C. Xia and Amitabh Varshney. Dynamic view-dependent simplification for polygonal models. IEEE Visualization ’96, pages 327–334, October 1996. ISBN 0-89791-864-9.

[Zhang97] H. Zhang, D. Manosha, T. Hudson, and K. Hoff. Visibility Culling Using Hierarchical Occlusion Maps. In Computer Graphics (SIGGRAPH 1997 Proceedings), pages 77–88, August 1997.

[Zhang98] Hansong Zhang. Effective Occlusion Culling for Interactive Display of Arbitrary Models. PhD thesis, Department of Computer Science, University of North Carolina at Chapel Hill, 1998.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Procedural Landscapes with Overhangs

Manuel N. Gamito
ADETTI
Edifício ISCTE
Av. das Forças Armadas, 1600 Lisboa
mag@iscte.pt

F. Kenton Musgrave
Pandromeda.com
15724 Trapshire Ct.
Waterford, VA 20197-1002, USA
musgrave@pandromeda.com

Abstract

Overhangs have been a major stumbling block in the context of terrain synthesis models. These models resort invariably to a heightfield paradigm, which immediately precludes the existence of any type of overhang. This article presents a new technique for the generation of surfaces, with the ability to model overhangs in a procedural way. This technique can be used generally to model landscape elements, not only terrains but also the surface of the sea. The technique applies non-linear deformations to an initial heightfield surface. The deformations occur after the surface has been displaced along some specified vector field. The method is conceptually simple and enhances greatly the class of landscapes synthesized with procedural models.

Keywords

Surface overhangs, Procedural models, surface deformation, ray-tracing, adaptive level of detail.

1. INTRODUCTION

The representation of terrains with mathematical models began with the seminal work of Mandelbrot on fractional Brownian motion [Mandelbrot83]. Mandelbrot realized that the output of an fBm process was visually similar to the ragged outline of a mountain range. He then proceeded to visualize a wire-frame model of a terrain, using a two-dimensional fBm process.

Several models have since been proposed for the synthesis of digital terrains, always relying on the concepts of fBm for the basic framework. Fournier and co-workers developed a polygon subdivision model, causing a triangular mesh to converge to a true fBm surface [Fournier82]. Voss used spectral synthesis to generate his terrains in the Fourier domain, based on the fBm power spectrum distribution [Voss85]. Saupe used Perlin's turbulence function, knowing that it had fractal properties [Saupe89]. This was the first procedural approach to terrain synthesis. Musgrave then pointed out that fBm, being a statistically homogeneous process, had drawbacks for terrain generation. In a true digital terrain map, gravity and erosion play an important role. Valleys, for instance, are always much smoother than mountain peaks due to the downward transport of sediments. This kind of subtleties is impossible to generate with an fBm model. Musgrave proposed a multifractal model, where each scale of the Perlin's turbulence function was weighted by the previously accumulated terrain height [Musgrave89]. Although this multifractal function has not yet been studied for its statistical properties, it is clear that it generates terrains with a much greater degree of realism.

Despite all the advances in the terrain generation techniques, terrains have always been represented with a heightfield paradigm. A heightfield representation is both flexible and intuitive with the only important drawback that it cannot represent overhangs. Overhangs are an ubiquitous feature in many actual terrains and can be present in a wide range of scales, from canyons and wind eroded rock formations to microstructures on the surface of rocks. Perhaps, the only known example of overhangs in the literature is related to breaking waves. It appears in the context of wave interactions with shorelines, with the consequent swelling and breaking of the wave trains [Fournier86].

It is clear that overhangs are an important issue that must be tackled in the context of landscape synthesis. The need for new algorithms with the ability to incorporate overhangs has already been acknowledged in the literature [Musgrave94]. This paper presents an attempt to achieve such a goal in a way that is as generic as possible. The model, here presented, can be used to generate overhangs, both in the traditional form of terrains and also in the form of breaking waves, following the work of Fournier. The main idea behind the technique consists in warping an initial heightfield surface along a vector field. The structure of the paper is as follows: In section 2 we present the algorithm, using the common framework of terrain models. In session 3 we discuss rendering issues. In section 4 we present some results and adapt the model to breaking waves. Section 5 gives some final remarks.

2. TERRAIN WARPING

Traditionally, terrains have been modelled by height functions $h = f(x,y)$: $\mathbf{R}^2 \alpha \mathbf{R}$ that associate an altitude h to each point (x,y) on the plane. These heightfields are then stored, either in memory or in disk, as two-dimensional altitude grids $h_{i,j} = f(x_i,y_j)$ for an underlying subdivision of the plane into equally spaced samples. Figure 1 shows an example of one heightfield terrain and also one terrain that cannot be represented with this paradigm. The problem with terrains exhibiting overhangs is that for some (x,y) values there is more than one altitude value. It is clear that the problem will be solved if we can somehow represent the terrain by some parametric function $\mathbf{S}(u,v)$: $\mathbf{R}^2 \alpha \mathbf{R}^3$. What remains to be seen is how this function can be generated.

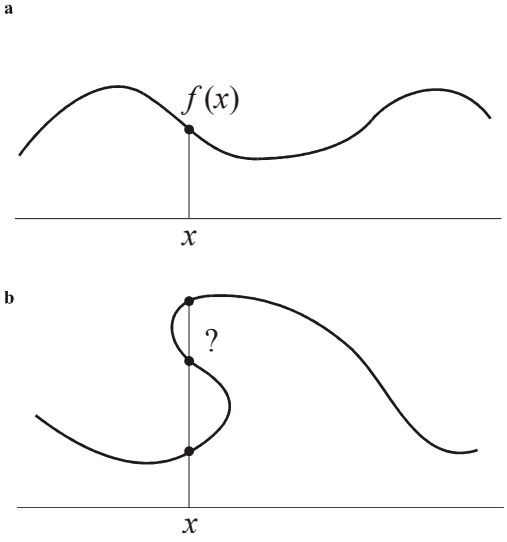


Figure 1. An heightfield terrain in (a) and a terrain with an overhang in (b).

Any algorithm that tries to generate non-heightfield terrains must obey two fundamental principles:

1. The terrain must be simply connected. This means that it must have a unique surface boundary separating an inside region from an outside region.
2. The terrain must not self-intersect. This constraint is obvious but is not that simple to implement.

The most naïve idea to implement non-heightfield terrains consists in the generation of a three-dimensional fractal field and the subsequent extraction of an isosurface from it. Voss used this method to obtain his images of "fractal flakes" [Voss85]. This technique violates condition 1 above, although it preserves condition 2. A more clever approach could use hypertextures as defined by Lewis and Perlin [Lewis89,Perlin89]. The problem with hyper textures is that the user must supply some initial volume density function. This function is subsequently "turbulated" and a fractal isosurface is extracted. It does not seem feasible to define explicitly a density function for every new terrain, unless some very specific geological feature is desired. Other hypotheses for terrain generation could include some simple deformation operations over an initial heightfield terrain (like deformation matrices or FFD lattices, for example) [Barr84,Sederberg86]. These techniques, however, are not general enough and have the significant drawback that condition 2 becomes very hard to hold. One would have to use some sort of collision detection algorithms to avoid self-intersections and thus control the maximum allowable amount of deformation.

A simple and elegant solution to this problem can be obtained by studying the apparently unrelated problem of fluid flow. Let's consider some flow field (a vector field $\mathbf{v}(\mathbf{x})$, expressing the velocity $\mathbf{v}$ at each point $\mathbf{x}$ in some n-dimensional space). Let's consider further a connected set of points inside the fluid at some initial time instant. This set can have a topological dimension of one (a line) or two (a surface). As time progresses, this set will be dragged by the surrounding fluid and will evolve into new shapes. This is usually called a *material set* since it is constituted by material particles that are dragged or *advected* by the flow [Batchelor67]. Figure 2 illustrates the idea. The two remarkable properties concerning the motion of material sets are that *they remain simply connected* if they were simply connected at the beginning and *they never self-intersect*, although different parts of the set can become arbitrarily close to each other. The conditions on the flow field for these two properties to hold are quite mild; all that is required is that the flow $\mathbf{v}(\mathbf{x})$ is continuous in the portions of space where the material set is supposed to evolve.

The advection of an initial surface through the action of some specified flow will be used to define the non-heightfield terrain. This operation will be called *terrain warping*, since it effectively warps the terrain from its initial shape to some new form. The initial shape of the terrain should be simple. It can be a heightfield terrain or even an horizontal plane. In the case of a horizontal plane, the flow field will be the sole responsible for adding shape and detail to the final surface.

The surface will be represented by a two-dimensional parametric function $\mathbf{S}(u,v,t)$. Every point on the surface is determined uniquely by a pair (u,v) of parameters. The variable t accounts for the evolution of the surface along the flow or, stated equivalently, the amount of warping that the surface has received.

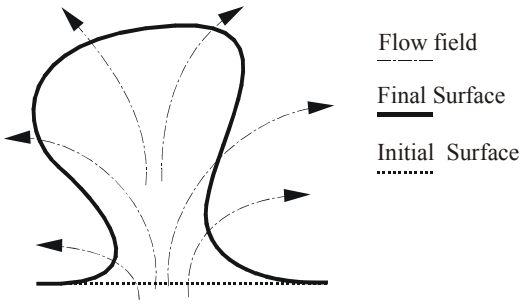


Figure 2. The motion of a material set of points under the action of a flow field.

Initially we will have $\mathbf{S}(u,v,0) = \mathbf{S}_0$, where $\mathbf{S}_0$ is the initial surface. Every point $\mathbf{x}(t) = \mathbf{S}(u,v,t)$ on the surface will then be advected along the flow, according to an ordinary differential equation of motion:

$$\frac{d\mathbf{x}}{dt} = \mathbf{v}(\mathbf{x},t) \quad (1a)$$

$$\mathbf{x}(0) = \mathbf{S}_0(u,v) \quad (1b)$$

This equation means that each particle will have a velocity, at any given instant, equal to the velocity of the flow at the point where the particle lies. The particle will, therefore, describe a trajectory along the field lines of the flow. For generality, we are considering here flow fields $\mathbf{v}(\mathbf{x},t)$ that may be time dependent. We will see in section 4.2 an example where this time dependency of the flow becomes useful.

The surface advection is a point-wise operation, acting on each surface point, independently from all the others. This can, therefore, be considered as a procedural modelling technique. We can formally define a warping operator, denoted by $Warp\{.\}$, such that:

$$\mathbf{S} = Warp\{\mathbf{S}_0; \mathbf{v}; t\} \quad (2)$$

A useful mechanism that all procedural textures and geometries share is the built-in ability to provide adaptive level of detail and anti-aliasing. This is usually accomplished by band-limiting the frequency content of the object as a function of distance to the observer. The same mechanism can be applied here in several ways: either by band-limiting the spectral content of the initial surface $\mathbf{S}_0$, the spectral content of the flow field $\mathbf{v}(\mathbf{x})$, or even both. Both entities can be made to disregard high frequency components for points that are progressively farther away from the viewpoint, in a seamless manner. This is similar to the principle that is used with procedural heightfields and textures [Musgrave98].

The operator (2) is invertible because any given point $\mathbf{S}_0(u,v)$ maps to one and only one point in $\mathbf{S}(u,v,t)$ and vice-versa. The inverse can easily be shown to be:

$$\begin{aligned}\mathbf{S}_0 &= Warp^{-1}\{\mathbf{S}; \mathbf{v}; t\} = \\ &= Warp\{\mathbf{S}; \mathbf{v}; -t\}\end{aligned} \quad (3)$$

Notice that we only need to invert the time variable, causing the particles to flow backwards towards their initial positions. The inverse warping operator will be of particular importance when we discuss the topic of rendering in the next section.

A numerical integrator is necessary to implement the warping operator, expressed in equation (1), in a general way [Press92]. For efficiency reasons, however, one should always give preference to flow fields such that equation (1) has an analytical solution. This will greatly reduce the computational cost of the algorithm and also increase the accuracy of the final non-heightfield terrain.

Typical heightfield terrains can also be generated with the present technique. One simply needs to specify a deformation $\mathbf{v}(x,y) = (0,0,f(x,y))$ with a vertical component only, where $f(x,y)$ is the desired heightfield function, and apply it to a horizontal plane. This feature provides the added flexibility of generating both heightfield and non-heightfield terrains, in an integrated fashion, with the specification of appropriate deformation fields.

3. RENDERING NON-HEIGHTFIELD TERRAINS

The authors have found that a ray-tracing algorithm is the most convenient for the rendering of terrains with overhangs. The other alternative would be to tessellate the terrain, in the parameter space (u,v), into a triangular mesh and render it using any conventional rendering algorithm. This is not convenient, however, because the surface may be subject to arbitrarily high deformations. These would give rise to extremely stretched triangles that would no longer represent the terrain with sufficient accuracy. One would have to adaptively refine the tessellation as function of the local rate of deformation, using a technique similar to that of [Snyder92]. A direct ray-tracing approach obviates all these algorithmic complexities and accommodates naturally terrains with an infinite area of support.

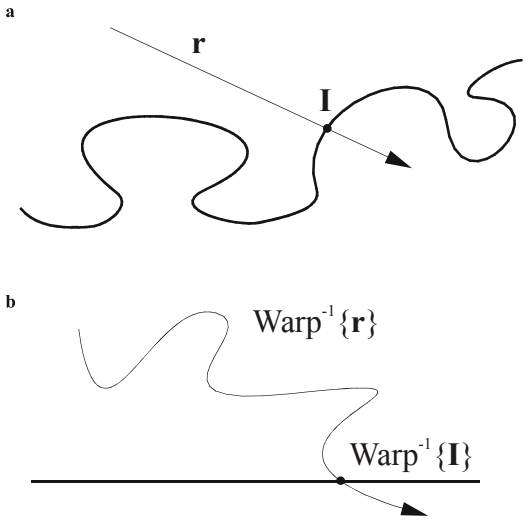


Figure 3. The intersection between a ray and the surface without inverse warping (a) and with inverse warping (b).

Two fundamental problems need to be solved in a ray-tracing context:

- Determination of the intersection point between a ray and the terrain.
- Determination of the terrain normal at the intersection point.

The first problem consists in determining, for any ray $\mathbf{r}(t)$ expressed in parametric form, the specific parameter $t = t_0$ for which the ray intersects the terrain. The intersection point is then given by $\mathbf{I} = \mathbf{r}(t_0)$. Instead of determining directly this ray-terrain intersection, we perform an inverse warping operation on the ray and determine its intersection with the initial surface. Figure 3 exemplifies this idea. The rationale is that the initial undeformed terrain has much less complexity than the non-heightfield terrain. It is better to find an intersection between a curve

and a simple shape than an intersection between a straight ray and a complex shape. The inverse warping operator will revert the terrain to its initial configuration, while transforming the ray into a parametric curve of arbitrary shape. In particular, if the initial terrain is the horizontal plane, the intersection test just needs to find the point where the vertical coordinate of the deformed ray changes sign. An intersection point, once found, must be warped back to its true position on the terrain. This type of ray domain deformation for the ray tracing of curved objects was initially proposed by Kajiya, although restricted to objects that were surfaces of revolution [Kajiya83].

3.1 Finding Terrain Intersections

The search for the intersection point between the deformed ray and the undeformed terrain is made with the help of an adaptive sampling of the ray, based on local curvature information. This idea is adapted from an algorithm described in [de Figueiredo95]. The algorithm recursively tests the smoothness of the ray between two points by sampling an intermediate point and finding the area of the enclosed triangle. In Figure 4, if point $\mathbf{P}_a$ is given by some parameter value t_a along the ray $\mathbf{r}(t)$, and point $\mathbf{P}_b$ by the parameter t_b, the intermediate point will be $\mathbf{P}_i = \mathbf{r}(0.5(t_a + t_b))$

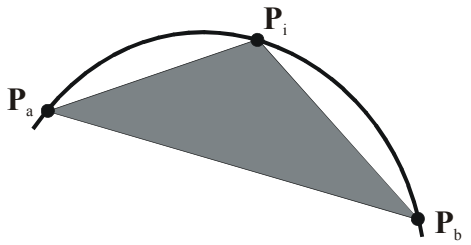


Figure 4. A subdivision of a parametric curve with local smoothness estimation based on the area of the triangle.

If the area is above a given threshold, the subdivision continues down to each of the two new intervals between the points, otherwise a straight line segment approximates that span of the ray. Every time such a straight segment is generated, it is also tested for intersection against the undeformed terrain. The following piece of pseudo-code clarifies the algorithm:

```
float CheckIntersection(Point Pa,
                        Point Pb) {
   float t;
   Point Pi = Midpoint(Pa,Pb);
   if (Area(Pa,Pi,Pb) < Epsilon)
       return Intersection(Pa,Pb);
   t = CheckIntersection(Pa,Pi);
   if (t > 0.0) return t;
   t = CheckIntersection(Pi,Pb);
   if (t > 0.0) return t;
   return -1.0;
}
```

In practice, one must also impose a maximum level of recursion to avoid locking the algorithm in sections where the ray might become discontinuous in the first derivative. Notice that, when descending to a lower level of subdivision, one always chooses first the interval that is closest to the eye point. In the case where the ray may intersect the terrain multiple times, this will guarantee that the first intersection found is the one which is closest to the viewer. We can also test for multiple intersections in the case where the terrain is transparent and light transport inside the medium must be considered. Again, the algorithm guarantees that the sequence of intersections will be sorted by increasing distance to the viewer, thereby facilitating the shading calculations.

The efficiency of the algorithm can be improved, at the cost of reducing somewhat its robustness, by noticing that parts of the ray far away from the surface need not be sampled with as much resolution. This effect is accomplished if we increase the tolerance factor for interval subdivision as a function of distance to the surface. The distance is taken to be the minimum distance of the two interval endpoints. In the simple case where the undeformed terrain is the horizontal plane, for instance, we can just replace the `Epsilon` factor in the previous pseudo-code by some linear function `Epsilon(z)` of the height z above the plane. The intersection routine will skim quickly through parts of the ray that are distant from the terrain and spend more time checking for intersections in parts that are at close proximity and where intersections are more likely to be found. This speedup, however, can lead to wrong intersections if the terrain exhibits very steep changes.

3.2 Finding Terrain Normal Vectors

Consider that an intersection was found at point $\mathbf{x}(u,v,T)$, where T is the amount of deformation and (u,v) are the parameter coordinates of the undeformed terrain. The normal at that point is calculated with the help of the two tangent vectors $\partial\mathbf{x}/\partial u$ and $\partial\mathbf{x}/\partial v$. These vectors can be computed by advecting them through the flow field, in exactly the same manner as the point $\mathbf{x}(u,v,T)$ itself was computed. For each of the two tangent vectors, we take the partial derivates at both sides of the advection equation (1). For instance, the trajectory of the tangent $\partial\mathbf{x}/\partial u$ will be given by:

$$\frac{d}{dt}\left(\frac{\partial \mathbf{x}}{\partial u}\right) = \nabla \mathbf{v} \cdot \frac{\partial \mathbf{x}}{\partial u} \qquad (4a)$$

$$\frac{\partial \mathbf{x}}{\partial u}(0) = \frac{\partial \mathbf{S}_0}{\partial u}(u,v) \qquad (4b)$$

where $\nabla\mathbf{v}$ is the gradient of the flow field. The tangent is initially equal to the tangent $\partial\mathbf{S}_0/\partial u$ of the undeformed terrain and evolves along with the flow up to the time T. The differential equation, giving the evolution of the tangent $\partial\mathbf{x}/\partial v$ is just like (4) if we replace the partials $\partial/\partial u$ by $\partial/\partial v$. If we consider again, the simplest case where the initial terrain is the horizontal plane, we will have:

$$\frac{\partial \mathbf{x}}{\partial u}(0) = (1,0,0) \quad (5a)$$

$$\frac{\partial \mathbf{x}}{\partial v}(0) = (0,1,0) \quad (5b)$$

Once an intersection point between a ray and the terrain is found, we warp it back to its position on the deformed terrain and, at the same time, obtain the two tangent vectors by solving the triple system of ODE's, consisting of the state vector $[\mathbf{x}\ \partial\mathbf{x}/\partial u\ \partial\mathbf{x}/\partial v]^T$, from $t = 0$ to $t = T$. With the two tangent vectors available, the normal is given by:

$$\mathbf{n} = \frac{\frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial \mathbf{x}}{\partial v}}{\left\| \frac{\partial \mathbf{x}}{\partial u} \times \frac{\partial \mathbf{x}}{\partial v} \right\|} \quad (5)$$

The external product of the tangents gives the vector that is perpendicular to the terrain at the point of intersection. The division by the modulus insures the normal is a proper unit sized vector, which can be used for shading calculations.

The above technique for the computation of the two tangent vectors relies on a prior knowledge of the gradient vector $\nabla\mathbf{v}(\mathbf{x})$ of the flow field. We have come across situations where obtaining the gradient of the flow is either difficult to compute or generates expressions that are too complex. In this situation, it is possible to use a less accurate method, that relies solely on knowledge of the flow field $\mathbf{v}(\mathbf{x})$ itself, to estimate the two tangents by considering the central finite difference approximations:

$$\frac{\partial \mathbf{x}}{\partial u} = \frac{\mathbf{x}(u-\varepsilon, v, T) - \mathbf{x}(u+\varepsilon, v, T)}{2\varepsilon} \quad (6a)$$

$$\frac{\partial \mathbf{x}}{\partial v} = \frac{\mathbf{x}(u, v-\varepsilon, T) - \mathbf{x}(u, v+\varepsilon, T)}{2\varepsilon} \quad (6b)$$

for some suitably small value of ε. In practice, one should try successively smaller ε values and monitor the converge of the tangent vectors. More details of this technique can be found in [Press92].

It is also interesting to note that the terrain, being a two-dimensional parametric surface, can be subject straightforwardly to image mapping techniques. It is only necessary to establish a one-to-one correspondence between the pixel coordinates of the image texture and the pair (u,v) of parametric coordinates of the surface. This could be used to texture the terrain with areas of vegetation or snow, based on a previously calculated image map. We will give an example in the results section where this mapping has been used to advantage.

4. RESULTS

In this section, we present two examples of how the current surface deformation technique can be used to generate terrains with overhangs. We must point out beforehand that all models derived here do not attempt to follow any accurate physical laws. This is in accordance with the philosophy of procedural modelling techniques, where one tries to find simple mathematical functions to create simulations of natural phenomena that are visually convincing, even if they are not related to the underlying physics of the problem. This approach to the modelling of natural phenomena has been termed *ontogenetic modelling* in [Musgrave98]. It is, very often, a necessity due to the high complexity of the physical laws and is used in situations where one only wishes to compute animations that "look right", without special regard for the physical accuracy of the achieved results.

4.1 A Canyon Terrain

The first example consists of a procedurally generated canyon. The initial shape of the canyon is obtained by saturating a Perlin based noise function $n(x,y)$ to the discrete altitudes 0 and 1. The saturation is based on the sign of the noise function, i.e., negative areas of the noise become the bottom plane $z = 0$, while positive areas become the top plane $z = 1$. Once this starting shape is defined, we deform the vertical walls of the canyon according to a horizontal flow field (Figure 5a). The intensity of deformation is changed as a function of height, according to an fBm function. This allows us to model horizontal layers or strata of rock that are slightly displaced relative to each other, thus creating overhangs. It also incorporates adaptive level of detail by limiting the number of octaves of the vertical fBm function as a function of distance. The flow field itself is given by the gradient of the Perlin noise function: $\mathbf{v}(x,y) = \nabla n(x,y)$. Given the properties of the gradient operator and the fact that the lateral walls are described by the isoline $n(x,y) = 0$, this flow field guarantees that the displacement of the lateral walls is always performed along a perpendicular direction (Figure 5b).

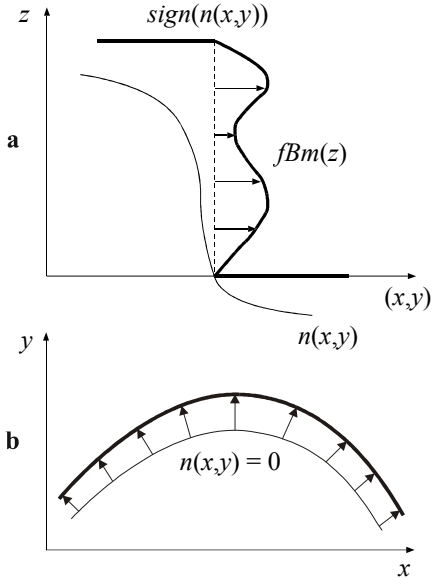


Figure 5. Construction of a canyon-like formation. Side view in (a) and top view in (b).

For this terrain configuration, the ray intersection test checks if each linear segment of the ray crosses from a negative to a positive area of the noise function, after the inverse warping operator has been applied. Consider

some portion of the ray $\mathbf{r}(t)$ from $t = t_a$ to t_b, which, according to our ray sampling algorithm, is small enough to be considered a straight segment. If it happens that $n(\mathbf{r}(t_a)) \times n(\mathbf{r}(t_b)) < 0$ then there must be an intersection for some parameter t_0 such that $t_a < t_0 < t_b$. The intersection parameter will be the solution of:

$$n(\mathbf{r}(t_0)) = 0 \qquad (7)$$

This is a non-linear equation, in terms of t_0, and since we know this unknown to be somewhere inside the interval $[t_a,t_b]$, we can apply a straightforward Newton-Raphson root-finder to converge to its true value. Once t_0 is known, we compute the intersection point $\mathbf{I}$ by warping back to the correct position in the deformed terrain:

$$\mathbf{I} = Warp\{\mathbf{r}(t_0); \nabla n; T\} \qquad (8)$$

Figure 6 shows the canyon with only one octave of an fBm function to deform the walls. The floor has not been rendered. The ability of procedural models, and of our non-heightfield procedural model in particular, to render terrains with infinite support becomes evident in this figure. Figure 7 shows the same canyon with eight octaves of fBm, deforming the walls at close range. In this case, the lower portion of the walls was sloped linearly with height to model a layer of weathering caused by the deposit of sediments from the upper strata. The floor of the canyon was also superimposed as a simple heightfield procedural terrain. This picture took 12 minutes of CPU time to render on a SGI O2 workstation at a resolution of 360x260 pixels.

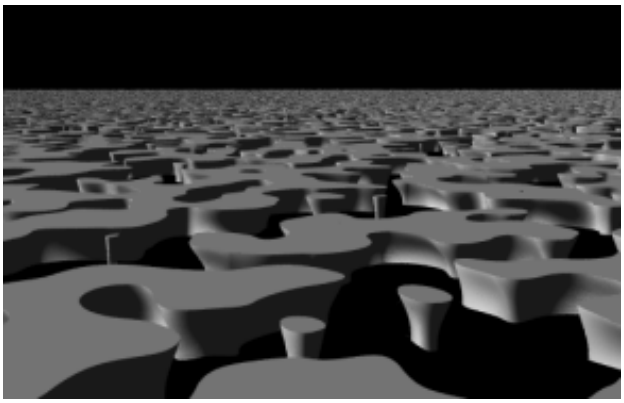


Figure 6. Saturation of a noise function into two discrete values with deformation of the lateral walls.

Figure 8 is another rendering of the same terrain from a different camera position. The rendering time for this picture was 1 hour of CPU time at a resolution of 360x360 pixels.

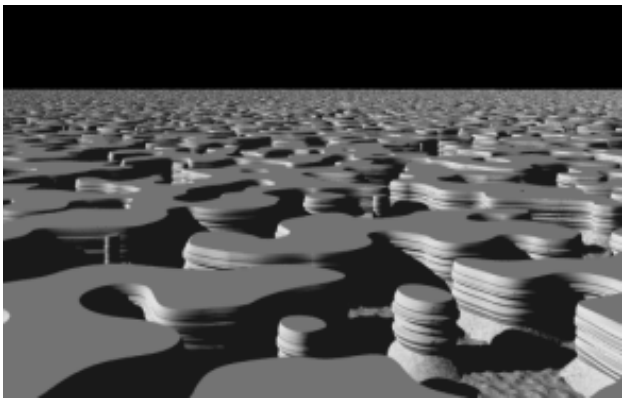


Figure 7. Same terrain as in Fig. 6 with increased deformation of the walls and a heightfield at the bottom.

The large disparity between these two computation times is related to the fact that the camera in Figure 8 is placed inside the canyon. For Figure 7, the camera is outside the canyon and each ray can be clipped against the range $0 < z < 1$ prior to any intersection checks because any intersection with the lateral walls must occur inside that range. This means, in practice, that only a relatively small portion of each ray must actually be considered for intersection. For Figure 8, on the other hand, this kind of optimisation cannot be applied and all of the ray must be checked for potential intersections.

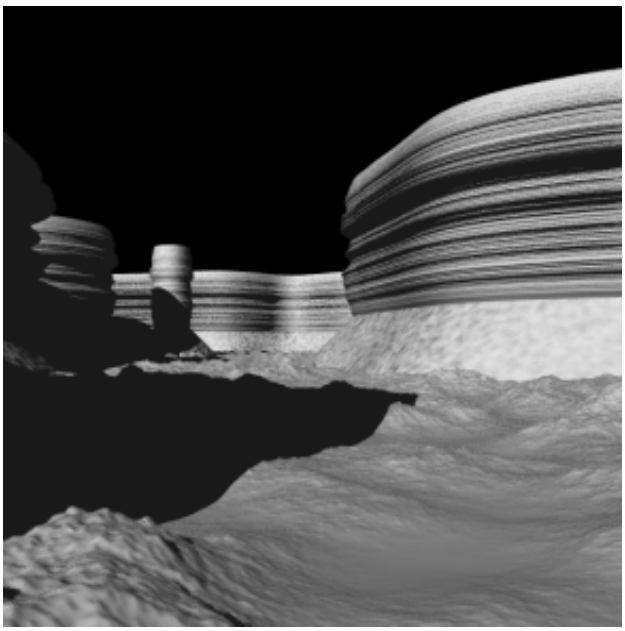


Figure 8. The terrain seen from a viewpoint inside the canyon.

4.2 Breaking Waves

In this example of a non-heightfield surface, we seek to reproduce and improve on some of the results given in [Fournier82]. Specifically, we seek to represent a breaking wave through all its stages of development. Other issues of the Fournier model, like the shallow water interaction with nearby shorelines, have been developed independently and are presented in [Gamito00]. We

intend to integrate the breaking wave model with our shallow water model in the near future.

The main restriction of the Fournier model, where breaking waves are concerned, is that, although the wave crest is deformed and an overhang is created, the wave crest never actually falls under gravity and hits the trough. To accomplish this effect, we take an horizontal plane to be the surface of the sea at rest and warp it along a deformation field with cylindrical shape. The expression $v(r)$ for the intensity of the deformation field is a function of distance r to the centre of rotation and is always tangential to the direction of rotation. Two radii of influence R_1 and R_2 and a decay factor α specify the field, according to the expression:

$$v(r) = \begin{cases} \left(1 - \dfrac{r - R_2}{R_1 - R_2}\right)^{\alpha} & \text{for } R_2 < r < R_1, \\ 0 & \text{otherwise} \end{cases} \qquad (9)$$

The deformation is more intense for $r = R_2$ and decays smoothly to zero at $r = R_1$. Figure 9 shows a sequence of deformations of the plane for increasing intensities of deformation. The parameters used were $R_1 = 0.75$, $R_2 = 20.0$ and $\alpha = 4.0$. To simulate the falling of the wave crest under gravity we use a secondary parabolic flow that implements correctly the law of gravity. The flow of equation (9) does not work for that purpose, as can be seen in Figure 10, because it would curl the wave into an ever increasing spiral.

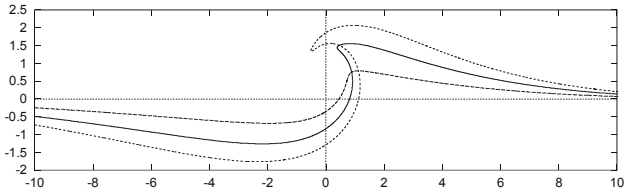


Figure 9. Several wave profiles for increasing intensities of deformation.

Any given particle switches from the circular to the parabolic flow once it crosses the half-line $\{x = 0; z > 0\}$, where coordinates are again relative to the centre of rotation of the flow. The parabolic flow is built such that every particle exhibits first and second order continuity of its trajectory at the transition point relative to the original circular flow. In this way, the deformed surface remains smooth everywhere, even though two different flow fields are being used to generate it. The expression for the parabolic flow is:

$$\mathbf{v}(x, z, t) = \left(-v, -\frac{v^2 t}{r}\right) \qquad (10)$$

for a wave that is travelling from the right to the left. In this equation, v is the horizontal velocity that each particle was carrying from the initial flow (9) at the transition point and $r^2 = (x^2 + y^2)$ is the distance to the centre of rotation at the same point.

4.2.1 Warping the surface of the wave

Say we want to warp some point $\mathbf{x}_0$ on the plane into some point $\mathbf{x}(T)$ on the wave for some amount T of deformation. This also means that we want to advect the point $\mathbf{x}_0$ along the flow for a time T. We start by applying the circular flow field (9) for the complete duration T:

$$\mathbf{x}(T) = Warp\{\mathbf{x}_0, \mathbf{v}_{circular}, T\} \qquad (11)$$

At the same time, we monitor the trajectory of the point and check if it starts a downward movement. This will happen, for some intermediate time $t < T$, when the point crosses the vertical axis above the centre of rotation. After that point, we begin a new deformation with the parabolic flow at the point where the previous deformation left off, i.e.:

$$\mathbf{x}(T) = Warp\{\mathbf{x}(t), \mathbf{v}_{parabolic}, T - t\} \qquad (12)$$

This advection is only applied for a time $T - t$, necessary to complete the total advection time of T from the initial point $\mathbf{x}_0$. Figure 10 shows the combined effect of the cylindrical and parabolic flows.

4.2.2 Inverse warping the surface of the wave

As we recall from section 3.1, an inverse warping operator must accompany every warping operator. This is so that we can check for ray-terrain intersections in undeformed space by applying the inverse deformation on the ray itself.

What we have been implementing in this particular case of a breaking wave is a form of state machine where each particle can undergo state changes when some predefined geometric conditions are verified. Different flow fields can then be applied based on the current state of each particle.

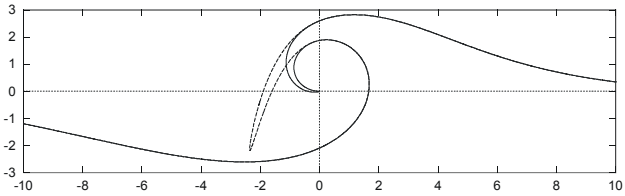


Figure 10. An excessive deformation can lead to incorrect results. A downward parabolic flow corrects the problem.

When we try to return from a point $\mathbf{x}(T)$, on the surface of the wave, to the original point $\mathbf{x}_0$ on the horizontal plane, two different hypotheses can happen:

1. The point is part of the plunging breaker and undergoes a state change from circular to parabolic flow.
2. The point is either in the rear or the trough of the wave and maintains a circular motion throughout the time T of the deformation.

If the point $\mathbf{x}(T)$ is on the rear of the wave (the region given by $x > 0$) then hypothesis 2 holds and we can simply apply:

$$\mathbf{x}_0 = Warp\{\mathbf{x}(T), \mathbf{v}_{circular}, -T\} \qquad (13)$$

If the point is in the front of the wave, the situation is a bit more ambiguous because it can either be part of the trough or the plunger (notice in Figure 10, in particular, that these two portions of the wave can easily intersect each other). We try hypothesis 1 first and apply:

$$\mathbf{x}_0 = Warp\{\mathbf{x}(T), \mathbf{v}_{parabolic}, -T\} \qquad (14)$$

always checking for the time $t < T$ when the particle switches to the circular flow. If we indeed find such a time, we return the correct point on the plane:

$$\mathbf{x}_0 = Warp\{\mathbf{x}(t), \mathbf{v}_{circular}, -(T-t)\} \qquad (15)$$

otherwise, the point is part of the trough and we can use equation (13) as it stands.

4.2.3 Animating the wave

It is quite straightforward to animate this wave model, if we just specify an intensity of deformation $T(y,t)$ that changes gradually along the axis y, around which the wave is curling, and also with time. Figure 11 shows several frames of the animation of a breaking wave that was produced with the above model. A short Mpeg clip of this animation can be found in [Gamito99].

There are several graphic elements in this animation that enhance the visual realism but are not directly related to the topic of this paper. In particular, the spray and the foam are modelled with procedural density functions. To render such elements, we march each ray forward in small increments and accumulate the opacity [Blasi93]. The backlighting on the surface of the wave is achieved by computing the distance travelled by each ray inside the wave and applying a simple exponential attenuation law. This means that both the entry point and the eventual exit point of each ray have to be computed. A procedural texture is also being applied on the top part of the plunging wave. As explained in section 3.2, this mapping can easily be accomplished since each point on the wave is identified by a unique (u,v) pair. This texture follows naturally the downward movement of the wave. The rest of the water surface is bump mapped with a three-dimensional procedural texture to simulate the small-scale ripples that are a characteristic of seawater.

Each frame of the animation took, on average, 40 minutes of CPU time for a resolution of 300x200 pixels. Despite the complexity of the scene present in these frames, the algorithm is quite efficient (compare with the computation time for Figure 8) because all the flow fields used in the breaking wave model have exact analytical solutions under the advection equation (1). This means that a numerical integrator for differential equations is not necessary and we can compute the $Warp\{\}$ operators with the simple evaluation of a mathematical expression. Some profiling tests, that we performed, have shown that only 40% of CPU time for the wave animation was spent in the ray-wave intersection tests. The remaining 60% of the computation time was spent integrating opacity values along the gas density functions and in computation of the atmospheric model.

5. CONCLUSIONS AND FUTURE WORK

The deformation of surfaces, driven by a vector field, provides a simple and extensible framework for the generation of overhanging structures. The operation is procedural, in the sense that it is performed point-wise and is controlled by a functionally defined vector field. It, therefore, inherits all the benefits of any procedural terrain modelling technique, like adaptive level of detail, built-in anti-aliasing and infinite support. The key to the success of this technique is to find appropriate deformation fields that will yield visually interesting results. This is in keeping with the philosophy behind procedural modelling techniques, where one tries to find simple mathematical functions that mimic natural phenomena to an acceptable degree. Two examples of surfaces with overhangs were already presented in this paper. We have seen, with these examples, how the basic technique can easily be adapted or extended to specific situations. More complex deformation fields should also be investigated, leading to more complex overhanging structures. The canyon terrain of section 4.1, for instance, could benefit if a small vertical deformation could also be added to the already existing horizontal deformation that generates the canyon walls. One other scenario, which is worth exploring in the future, is to apply the current deformation technique to surfaces that start out from an implicit representation, instead of the parametric representations that were used throughout this paper. Implicit surface representations can exhibit much greater topological richness that the relatively simple parametric representation. This added richness will certainly enlarge the range of non-heightfield surfaces that can be modelled with our technique.

The rendering of surfaces with overhangs is accomplished with a ray-domain inverse deformation. The deformed ray is sampled adaptively, based on its local curvature, and checked for intersection with the surface. The technique presented in this paper is not foolproof and can lead to wrong intersections whenever deformation fields with steep derivatives are present. The authors, however, have encountered no problems with the deformation fields used so far. We envisage that techniques like interval arithmetic or Lifchitz bounds could possibly be used to increase the robustness of the ray-surface intersection algorithm for any input deformation field [Snyder92,Hart96]. These robust techniques should also allow computational times to decrease significantly since larger steps along each ray can be taken with the confidence that no intersection will occur inside them.

6. REFERENCES

[Barr84] Barr, A.H., Global and local deformations of solid primitives, *in* H. Christiansen, ed., "Computer Graphics (SIGGRAPH '84 Proceedings)", vol. 18, pp. 21-30, 1984.

[Batchelor67] Batchelor, G.K., Fluid Dynamics, Cambridge University Press, ISBN 0-521-09817-3, 1967.

[Blasi93] Blasi, P., Le Saec, B., Schlick, C., A Rendering Algorithm for Discrete Volume Density Objects, *Computer Graphics Forum*, **12**(3), 201-210, 1993.

[de Figueiredo95] de Figueiredo, L.H., Adaptive sampling of parametric curves, *in* A. Paeth, ed., "Graphics Gems V", AP Professional, 1995.

[Fournier82] Fournier, A., Fussell, D., Carpenter, L., Computer rendering of stochastic models, *Communications of the ACM*, **25**(6), 371-384, 1982.

[Fournier86] Fournier, A., Reeves, W.T., A simple model of ocean waves, *in* D.C Evans & R.J. Athay, eds., "Computer Graphics (SIGGRAPH '86 Proceedings)", vol 20, pp371-384, 1986.

[Gamito99] Gamito, M.N., Musgrave, F.K., Non-heightfield rendering, <http://www.wizardnet.com/musgrave/sundry.html>

[Gamito00] Gamito, M.N., Musgrave, F.K., An accurate model of wave refraction over shallow water, *in* N. Magnenat-Thalmann, D. Thalmann, B. Arnaldi, eds., "Computer Animation and Simulation '2000", Eurographics, pp. 155-171, 2000.

[Hart96] Hart, J.C., Sphere Tracing: a geometric method for the antialiased ray tracing of implicit surfaces, *The Visual Computer*, **12**(9), 527-545, Springer-Verlag, 1996.

[Kajiya83] Kajiya, J.T., New techniques for ray tracing procedurally defined objects, *in* "Computer Graphics (SIGGRAPH '83 Proceedings)", vol. 17, pp. 91-102, 1983.

[Lewis89] Lewis, J.P., Algorithms for solid noise synthesis, *in* J. Lane, ed., "Computer Graphics (SIGRAPH '89 Proceedings)", vol 23, pp. 263-270, 1989.

[Mandelbrot83] Mandelbrot, B.B., The Fractal Geometry of Nature, W.H. Freeman and Co., New York, 1983.

[Musgrave89] Musgrave, F.K., Kolb, C.E., Mace, R.S., The synthesis and rendering of eroded fractal terrains, *in* J. Lane, ed., "Computer Graphics (SIGGRAPH '89 Proceedings)", vol. 23, pp. 41-50, 1989.

[Musgrave94] Musgrave, F.K., Texturing and Modelling: A Procedural Approach, 1st. ed., AP Professional, chapter 9, pp. 297-298, ISBN 0-12-228760-6, 1994.

[Musgrave98] Musgrave, F.K., Texturing and Modelling: A Procedural Approach, 2nd ed., AP Professional, ISBN 0-12-228730-4, 1998.

[Perlin89] Perlin, K., Hoffert, E.M., Hypertexture, *in* J. Lane, ed., "Computer Graphics (SIGGRAPH '89 Proceedings)", vol. 23, pp. 253-262, 1989.

[Press92] Press, W.H., Teukolsky, S.A, Vetterling, W.T., Flannery, B.P., Numerical Recipes in C: The Art of Scientific Computing, 2nd ed., Cambridge University Press, ISBN 0-521-43108-5, 1992.

[Saupe89] Saupe, D., Point evaluation of multi-variable random fractals, *in* J. Jurgens & D. Saupe, eds., "Visualisierung in Mathematik und Naturwissenschaft", Bremer Computergraphik Tage, Springer-Verlag, Berlin, 1989.

[Sederberg86] Sederberg, T.W., Parry, S.R., Free-form deformation of solid geometric models, *in* D.C. Evans & R.J., Athay, eds., "Computer Graphics (SIGGRAPH '86 Proceedings)", vol. 20, pp. 151-160, 1986.

[Snyder92] Snyder, J.M., Interval Analysis for Computer Graphics, *in* Catmull, E., ed., "Computer Graphics (SIGGRAPH '92 Proceedings)", vol. 26, pp. 121-130, 1992.

[Voss85] Voss, R.P., Fractal Forgeries, *in* R.A. Earnshaw, ed., "Fundamental Algorithms for Computer Graphics", Springer-Verlag, 1985.

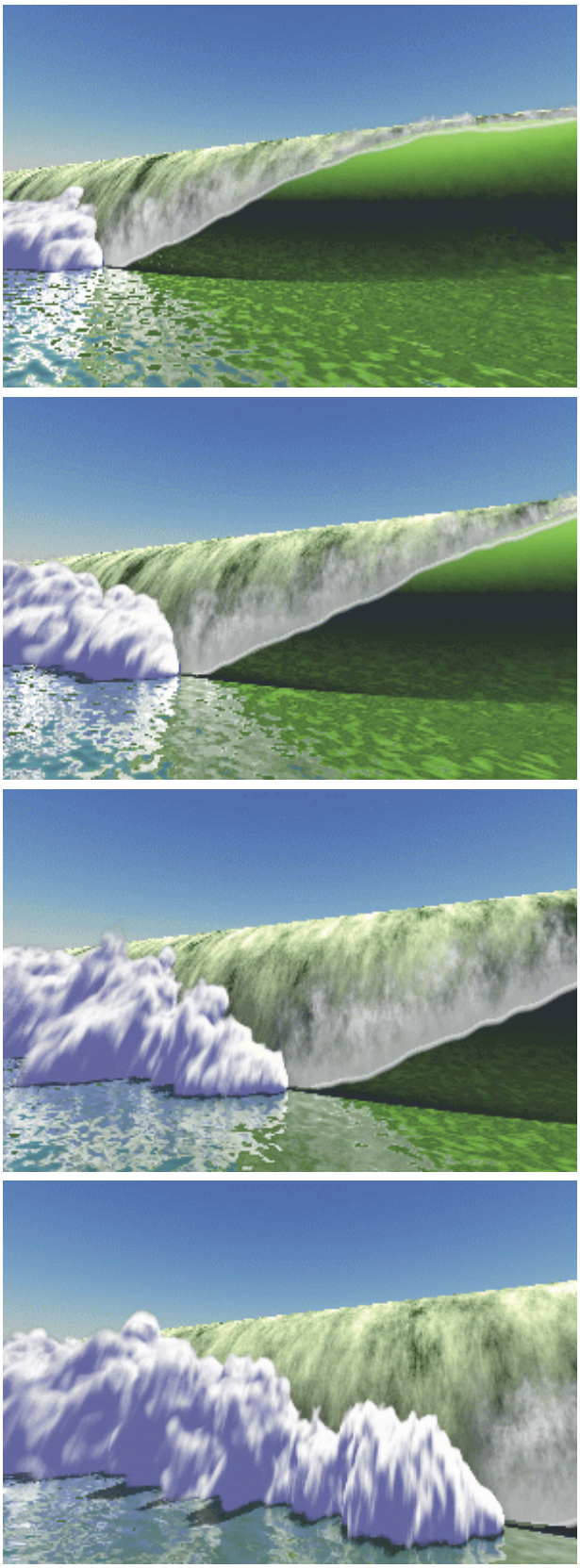


Figure 11. Several frames of the animation of a breaking wave. This is a complex scenario featuring non-heightfield waves, volumetric foam and spray, texture mapping, atmospheric light models, reflections and refractions on the surface of the water.

Sessão 2

INTERACÇÃO

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Interactive facilities for collaborative feature modeling on the Web

Rafael Bidarra Eelco van den Berg Willem F. Bronsvoort

Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4, NL-2628 CD Delft, The Netherlands
http://www.cg.its.tudelft.nl

(R.Bidarra/E.vdBerg/W.F.Bronsvoort)@its.tudelft.nl

Abstract

Collaborative modeling systems are distributed multiple-user systems that are both concurrent and synchronized, aimed at supporting engineering teams in coordinating their modeling activities. During a collaborative modeling session, several users are connected to each other in order to perform activities, such as design, manufacturing planning or evaluation, together, using some common product data. An interesting research challenge is to develop a collaborative modeling system that offers all facilities of advanced modeling systems to its users.

This paper focuses on the interactive modeling facilities required by such collaborative modeling systems. Previous work in the area of collaborative modeling is surveyed, and several techniques for interaction with feature models are presented, ranging from display of sophisticated feature model images to interactive selection facilities, which have been implemented in the web-based collaborative feature modeling system webSPIFF. It has a client-server architecture, with an advanced feature modeling system as a basis of the server, providing feature validation, multiple views and sophisticated visualization facilities.

The architecture of webSPIFF, the functionality of the server and the clients, their communication mechanisms, and the distribution of model data is described. In particular, maintenance and synchronization of model data at the clients, and techniques for their effective utilization for enhancing user interaction and collaboration are described. It is shown that a good compromise between interactivity and network load has been achieved, and that indeed advanced modeling with a collaborative system is feasible.

Keywords

Feature modeling, collaborative modeling, web-based modeling, graphical interaction

1. INTRODUCTION

In the last decade, research efforts in the areas of solid and feature modeling substantially contributed to the improvement of computer-aided design (CAD) systems. A broad range of advanced modeling facilities is now becoming available in high-end commercial systems, amplified by continuous enhancements in interactive and visualization capabilities, and profiting from the availability of faster and more powerful hardware. Still, these improvements have their counterpart in the increasing size and complexity of such systems. At the same time, a number of research prototypes are pushing the edge to even more advanced modeling facilities. For example, embodiment of richer semantics in feature models and validity maintenance of such models [Bidarra and Bronsvoort 2000], and physically-based modeling techniques [Kagan *et al.* 1999] are among the current research issues.

A common characteristic of most current CAD systems is that they run on powerful workstations or personal computers. Interaction with the system is usually only possible if the user is directly working at the CAD station, although remote interaction is sometimes possible through a high-bandwidth local area network. This situation is no longer satisfactory, as nowadays more and more engineers, often at different locations, are getting involved in the development of products. It would be preferable if a user could remotely browse and manipulate a model, via Internet, as if he were working directly at a powerful CAD station. A web-based system would be ideal for this, as it would facilitate access to all sorts of product information in a uniform, simple and familiar framework.

Even more attractive would be the support of collaborative modeling sessions, in which several geographically distributed members of a development team could work

together on the design of a product. Typically, in such collaborative sessions, different participants would be provided with their own, application-specific views on the product model according to the analyses or activities required, e.g. detailed design, manufacturing planning or assembly planning [de Kraker *et al.* 1997; Hoffmann and Joan-Arinyo 1998]. In addition, each session participant, as in normal development teams, should be given his own competence and specific session privileges by the system.

So far, only a small number of tools have been developed that somehow support collaborative design activities. For example, tools for collaborative model annotation and visualization via Internet are now becoming available, providing concepts such as shared cameras and telepointers [Autodesk 2000; Parametric 2000; Kaon 2001]. However, such tools are primarily focused on inspection, e.g. using simple polygon mesh models, and do not support real modeling activities. In other words, they are valuable assistants for teamwork, but no real CAD systems. Some more recent research is focusing on the possibility of enhancing existing CAD systems with collaborative facilities; see Section 2. To the best of our knowledge, the only commercial system currently offering some collaborative modeling facilities is OneSpace [CoCreate 2000]. However, its modeling capabilities are severely constrained by the modeler at the server, SolidDesigner, and by the model format into which it converts all shared models.

The idea of collaborative modeling combines very well with the increasingly popular concept of Application Service Providers (ASP), in which clients remotely access, via Internet, specialized applications running on a server, being billed exclusively for the service time they spend logged on at the ASP server. Such an approach has been identified as a very promising and affordable alternative for distributed CAD teams [Comerford 2000]. The first rudimentary commercial CAD ASP has recently been launched by [CollabWare 2000].

In order to satisfy all requirements outlined above, it is an interesting research challenge to develop a modeling system that offers all facilities of advanced feature modeling systems to its users, while at the same time providing them with the necessary coordination mechanisms that guarantee an effective collaboration. With this goal in mind, a new web-based, collaborative feature modeling system has been developed at Delft University of Technology. A complete description of its client-server architecture and functionality can be found in [Bidarra *et al.* 2001], including solutions to the critical concurrency and synchronization problems that characterize collaborative design environments.

This paper concentrates on the facilities for interaction with feature models of this prototype system. In particular, it describes the models that are maintained by the clients, and how these models can be effectively used for visualization and user interaction. The paper is organized as follows: first, the main research issues of collaborative modeling are briefly surveyed (Section 2); second, an overview of the web-based, collaborative modeling system is given (Section 3); third, several facilities for interactive visualization of the product model, in particular of its features, are described (Section 4); fourth, techniques are presented for interactively selecting and using feature entities in the specification of modeling operations (Section 5); finally, some results and conclusions are presented (Section 6).

2. PREVIOUS WORK

Collaborative systems can be defined as distributed multiple-user systems that are both concurrent and synchronized. *Concurrency* involves management of different processes trying to simultaneously access and manipulate the same data. *Synchronization* involves propagating evolving data among users of a distributed application, in order to keep their data consistent.

These concepts being in general already rather demanding, their difficulty becomes particularly apparent within a collaborative modeling framework, where the amount of model data that has to be synchronized is typically very large, the concurrent modeling actions taking place may be very complex, and the requirements for real time display and user interaction are so acute. This section briefly surveys collaborative modeling, highlighting the key aspects put forward by recent research, and summarizing the lessons learned from a few prototype systems proposed so far.

2.1. Client-server architecture

The requirements for concurrency and synchronization in a collaborative modeling context lead almost inevitably to the adoption of a *client-server* architecture, in which the server provides the participants in a collaborative modeling session with the indispensable communication, coordination and data consistency tools, in addition to the necessary basic modeling facilities. For a recent survey on client-server architectures, see [Lewandowski 1998].

A recurrent problem in client-server systems lies in the conflict between limiting the complexity of the client application and minimizing the network load. In a collaborative modeling context, client complexity is mainly determined by the modeling and interactive facilities implemented at the client, whereas network load is mainly a function of the kind and size of the model data being transferred to/from the clients.

A whole range of compromise solutions can be devised between the two extremes, so-called *thin clients* and *fat clients*. A pure thin-client architecture typically keeps all modeling functionality at the server, which sends an image of its user interface to be displayed at the client. Clicking on the image generates an event, containing the screen coordinates of the interface location the user clicked on. This event is sent to the server, which associates it with an action on a particular widget. Eventually, this action is processed, and an updated image of the resulting user interface is sent back to the client, where it is

displayed. This approach requires a continuous information stream between server and clients, and is therefore very expensive in terms of network traffic. The response time would be intolerably high for many model specification actions, thus making it very ineffective to remotely participate in a modeling session.

On the other extreme, a pure fat client offers full local modeling and interaction facilities, maintaining its own local model. Communication with the server is then often required in order to synchronize locally modified model data with the other clients. In a collaborative environment where clients can concurrently modify local model data, preventing data inconsistencies between different clients becomes a crucial problem. In addition, fat clients, to be effective, place on the platform running them the heavy computing power requirements of typical CAD stations.

2.2. Current research prototype systems

Several collaborative modeling prototype systems have recently been described in literature. Some of these systems will be shortly surveyed here, and their shortcomings identified.

CollIDE [Nam and Wright 1998] is a plug-in for the Alias modeling system, enhancing it with some collaborative functionality. Users of CollIDE have private workspaces, where model data can be adjusted independently from other users. In addition, a shared workspace exists containing a global model, which is synchronized between all users participating in a collaborative modeling session. Users can simply copy model data between the private and shared workspaces, in order to create and adjust certain model data locally, and add it to the model in the shared workspace. The architecture of CollIDE poses severe restrictions to crucial collaborative modeling issues. In particular, no special measures have been taken to reduce the amount of data sent between the participants of a collaborative modeling session, resulting in delayed synchronization of the shared workspace and of the users' displays. Also, since each user operates on a separate instance of the modeling system, able to perform modeling operations by itself, concurrency has to be handled by the users themselves in order to keep shared model data consistent. A positive point in CollIDE is, however, that it provides its users with most interactive and visualization facilities of the native system Alias, although this comes at the cost of a very fat client.

The ARCADE system [Stork and Jasnoch 1997] defines a *refine-while-discussing* method, where geographically distributed users can work together on a design, interacting with each other in real-time. Every participant uses a separate instance of the ARCADE modeling system, and all ARCADE instances are connected to a session manager via Internet. A message-based approach was chosen, where every change of the product model is converted into a short textual message, which is sent to all other instances of ARCADE through the session manager. ARCADE provides a collaborative environment in which the network load is kept low. This was done by including all modeling functionality in the distributed ARCADE instances, which exchange only textual messages, rather than large sets of polygons. A drawback of this approach, however, is that the user application becomes rather complex, thereby requiring much computational power. In addition, ARCADE provides a primitive concurrency control mechanism, where only one user can edit a particular part at a time.

CSM, the Collaborative Solid Modeling system proposed by [Chan *et al.* 1999] is a web-based collaborative modeling system. Within its client-server architecture, the server contains a global model, while every client owns a local copy of this model. When a user has locally modified the model, it is propagated to all other users through the server. Concurrency is managed in two ways: (i) the model can be locked, using token passing, restricting it from being accessed by other users as long as some user is performing a modeling operation; and (ii) functionality can be locked, preventing certain functions from being used by particular users. Clearly, such methods provide a very strict concurrency handling policy. In fact, they turn the clients into several independent modeling systems, just using the same product model alternately. In a truly collaborative modeling system, one expects a higher level of coordination support.

NetFEATURE [Lee *et al.* 1999] claims to be a collaborative, web-based, feature modeling system. A server provides basic functions on a central product model, including creation and deletion of features. On the clients, a local model is available, containing a boundary representation of the product, derived from the server-side central model. The local model is reportedly used for real-time display, navigation and interaction, although very little has been described by the authors on how this operations take place. For more advanced operations, the server must be accessed. Updating the local model is done incrementally, which required a rather heavy naming scheme. This scheme severely reduces the modeling functionality of the system, degrading it to a history-based geometric modeling system, instead of a genuine feature modeling system. Furthermore, NetFEATURE uses, just like CSM, very strict concurrency handling methods, thus seriously limiting genuine collaborative modeling.

2.3. Conclusions

Collaborative modeling systems can support engineering teams in coordinating their modeling activities. Instead of an *iterative* process, sending product data back and forth among several team members, designing becomes an *interactive* process, in which several engineers are simultaneously involved to agree on design issues. Collaborative modeling systems typically have a client-server architecture, differing in the distribution of functionality and data between clients and server.

Concurrency control is still a crucial issue in current collaborative environments. If a user is allowed to change a model entity, while another user is already changing the same entity, problems can easily arise concerning consis-

tency. To avoid this situation, a strict concurrency control mechanism can limit access for other users. It depends on the application, whether all entities of the design should be locked or just some of them. If possible, users should be allowed to simultaneously modify different parts of the design, but this could lead to much more complicated concurrency control mechanisms. Also, one should always bear in mind that designing is a constructive activity. Users can therefore be given some responsibility for establishing a good collaboration.

Current systems also often fall short in adequately handling synchronization of model data among distributed clients. Timely updating data over a network is difficult, since there is a certain delay between the moment data is sent and the moment it is received at another node of the network; during this time interval, the latter might try to manipulate data that is not up-to-date. Mechanisms to detect such conflicts should be available, and recovery mechanisms provided. Good locking can also help to avoid such situations, but sometimes it may hinder users' flexibility.

For a collaborative CAD system to be successful, it should combine a good level of interactivity with the sort of powerful visualization typically provided by conventional CAD systems. Users will not be able to design properly if they have to wait a considerable time after every operation. But increasing interactivity by just porting more and more data and functionality to the clients is not a good solution either, as synchronization problems would then turn critical. Furthermore, fat clients are typically platform-dependent applications that require more complex installation and maintenance procedures, and are therefore less practical in a web-based context.

In conclusion, a good compromise solution to the difficulties summarized above can better be achieved with a web-based, client-server architecture. The next section introduces webSPIFF, a prototype system that follows this approach.

3. ARCHITECTURE OF webSPIFF

webSPIFF has a client-server architecture, consisting of several components; see Figure 1. On the server side, two main components can be identified: the SPIFF modeling system, providing all feature modeling functionality; and

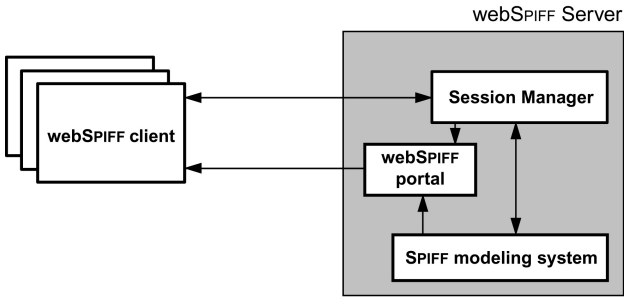


Figure 1. Architecture of webSPIFF

the Session Manager, providing functionality to start, join, leave and close a modeling session, and to manage all communication between SPIFF and the clients. The webSPIFF portal component provides the initial access to a webSPIFF session for new clients, and includes a web server where model data is made available for download by the clients.

The clients perform operations locally as much as possible, e.g. regarding visualization of, and interaction with, the feature model, and only high-level semantic messages, e.g. specifying modeling operations, as well as a limited amount of model information necessary for updating the client data, are sent over the network.

The server coordinates the collaborative session, maintains a central product model, and provides all functionality that cannot, or should not, be implemented on the client. In particular, as soon as real feature modeling computations are required, such as required by modeling operations, by conversion between feature views and by feature validity maintenance, they are executed at the webSPIFF server, on the central product model, and their results are eventually exported back to the clients.

An important advantage of this architecture is that there is only one central product model in the system, thus avoiding inconsistency between multiple versions of the same model.

3.1. The server

As a basis for the server, the SPIFF system developed at Delft University of Technology was chosen, which offers several advanced modeling facilities. First, it offers *multiple views* on a product model, each view consisting of a feature model with features specific for the application corresponding to the view. The current version of webSPIFF provides two such views: one for design and another for manufacturing planning of parts. In the design view, the feature model consists of both additive (e.g. protrusions) and subtractive (e.g. slots and holes) features. In the manufacturing planning view, the feature model consists of only subtractive features. All views on a product model are kept consistent by feature conversion [de Kraker *et al.* 1997]. Second, it offers *feature validity maintenance* functionality. This can guarantee that only valid feature models, i.e. models that satisfy all specified requirements, are created by a user [Bidarra and Bronsvoort 2000]. Third, it offers *sophisticated feature model visualization* techniques, which visualize much more specific feature information than most other systems do. For example, feature faces that are not on the boundary of the resulting object, such as closure faces of a through slot, can be visualized too [Bronsvoort *et al.* 2001]. All these facilities are computationally expensive, and require an advanced product model, including a cellular model with information on all features in all views [Bidarra *et al.* 1998].

The Session Manager stores information about an ongoing session and its participants. It manages all information streams between webSPIFF clients and the SPIFF modeling

system. Since several session participants can send modeling operations and queries to the webSPIFF server at the same time, concurrency must be handled at the Session Manager. It is also the task of the Session Manager to synchronize session participants, by sending them the updated data structures, after a modeling or camera operation has been processed. The Session Manager has been implemented using the Java programming language [Sun Microsystems 2000].

3.2. The clients

The clients of webSPIFF make use of standard web browsers. When a new client connects to the webSPIFF portal, a Java applet is loaded, implementing a simple graphical user interface (GUI), from which a connection with the Session Manager is set up. Different clients can connect from various locations, local through a network or remote via Internet, in order to start or join a modeling session.

Once connected to the server, the user can join an ongoing collaborative session, or start a new one, by specifying the product model he wants to work on. Also, the desired view on the model has to be specified. Information on the feature model of that view is retrieved from the server, and used to build the client's GUI, through which the user can start active participation in the modeling session.

The bottom line is obviously that clients should be able to specify modeling operations in terms of features and their entities; for example, a feature, to be added to a model, should be attachable to entities of features already in the model (e.g. faces and datums). After a feature modeling operation, with all its operands, has been fully specified, the user can confirm the operation. The operation is then sent to the server, where it is checked for validity and scheduled for execution. Notice that this can result in an update of the product model on the server, and thus also of the feature model in the view of each session participant.

In addition to the above functionality, several visualization and interactive facilities of the SPIFF system have also been ported to the clients. All these facilities of webSPIFF clients make use of so-called *camera* windows, i.e. separate windows in which a graphical representation of the product model is shown. Visualization facilities include displaying of sophisticated feature model images, rendering of a 3D model that supports interactive modification of camera viewing parameters (e.g. rotation and zoom operations) and a collaborative, shared camera. These will be dealt with in Section 4. webSPIFF cameras also provide facilities for interactive specification of modeling operations, e.g. assisting the user in selecting features or feature entities by having them picked on a sophisticated image of the model. The interactive functionality of webSPIFF cameras is described in detail in Section 5.

3.3. Distribution of model data

As explained above, only one central product model is maintained at the server. This feature model includes all canonical shapes, representing individual features in a specific view, and the cellular model. To support visualization and interaction facilities, however, the webSPIFF clients need to locally dispose of some model data. This data is derived by the server from its central model, but it does not make up a real feature model. webSPIFF clients need just enough model information in order to be able to autonomously interact with the feature model, i.e. without continuously requesting feedback from the server.

Model data at the clients can be classified into the following three categories:

Textual data

This data is used for specific sets of model information, mostly in list form. The most important are:

- *List of feature classes:* contains the names of all feature classes available in a given view. It is used to fill a GUI list widget when adding a new feature instance, and is requested from the server at client initialization time. This list does not need to be refreshed during a modeling session.
- *List of feature instances:* contains the names of all feature instances in a given view of the model. It is used to fill a GUI list widget when editing or removing an existing feature instance. This list is set upon initialization of the client, and is refreshed after each modeling operation.
- *List of parameter values:* contains the values of all parameters of a given feature instance, in a predefined order. It is used to fill various GUI entry widgets when editing the selected feature instance, and is always queried before a feature editing operation.

Graphical data

This comprises the *sophisticated feature model images*, rendered at the SPIFF server, and displayed in camera windows at the clients. These images provide very powerful visualizations of a feature model, as explained in Subsection 4.1. Many visualization options can be specified. A separate image is needed for each camera, and it must be updated every time the model or the camera settings are changed. The images are stored in the web server to be downloaded by the respective clients.

Geometric data

This comprises two kinds of models: the visualization model and the selection model.

- *visualization model:* represents the global shape of the product model. The visualization model is generated by SPIFF, and it is used at the clients for interactively changing the camera viewing parameters, as explained in Subsection 4.2. All cameras on a particular client use the same local visualization model,

but each camera displays it with its own viewing parameters.

- *selection model:* is a collection of three-dimensional objects representing the canonical shapes of all features in a given view of the model. It is also generated by SPIFF, with the purpose of supporting the interactive selection of feature faces on a client's camera, during the specification of a modeling operation, as explained in Section 5. Again, the selection model is identical for all cameras on a client, each applying its own viewing parameters.

Client model data is never modified directly by the clients themselves. Instead, after a modeling operation has been sent to the server and executed, updated model data is sent back to the client. In addition, if the central product model has been changed, appropriate updated model data is sent to all other session participants as well. This consists of, possibly several, new model images, a new visualization model, and an incremental update of the selection model (containing only new and/or modified feature canonical shapes).

Similarly, when a client modifies any camera settings, the corresponding camera operation is sent to the webSPIFF server, which generates a new sophisticated feature model image. Since the feature model remains unaffected by camera operations, the server only needs to send the new sophisticated feature model image back to the client that requested the camera update.

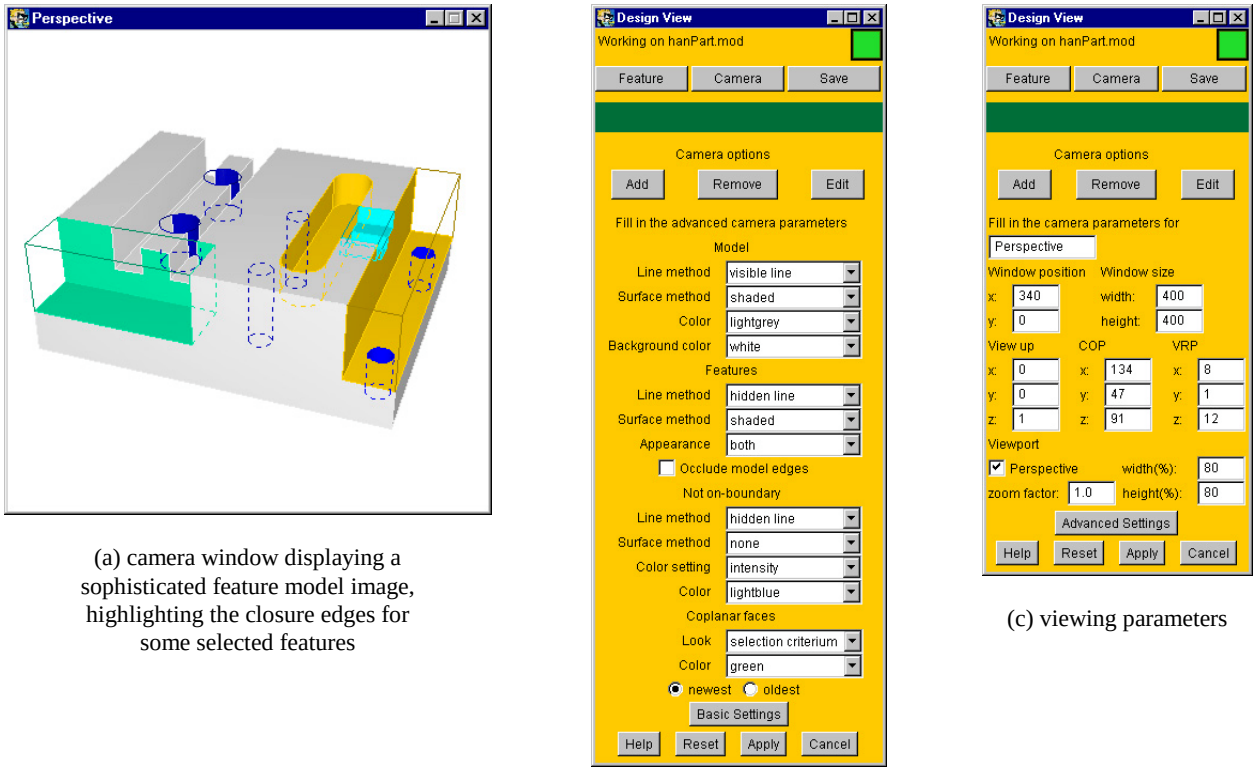
Temporary local inconsistencies can still occur at a client, for example after the execution of a modeling operation. Since sending information from the server to all clients takes some time, for a short period model information on the clients is not up-to-date. Avoiding conflicts arising from this transitory mismatch is one of the main tasks of the Session Manager.

4. PRODUCT MODEL VISUALIZATION

As anticipated in Subsection 3.2, webSPIFF provides clients with three ways of visualising a product model, all making use of *camera windows*. Each client may create as many cameras as desired. First, a sophisticated feature model image can be displayed. Second, a model can be rendered that supports interactive modification of camera viewing parameters, e.g. rotation and zoom operations. Third, shared cameras may be created that support collaborative, synchronized visualization of the model among several users.

4.1. Sophisticated feature model images

The most powerful visualisation technique generates *sophisticated feature model images*, which can very effectively support the user during the modeling process. These camera images provide not only a plain visualisation of the resulting final shape of the product model. Several advanced visualisation techniques are available that allow the user to customise the images to a variety of



(a) camera window displaying a sophisticated feature model image, highlighting the closure edges for some selected features

(b) model visualization parameters

(c) viewing parameters

Figure 2. webSPIFF camera panels

needs [Bronsvoort *et al.* 2001]. Sometimes, a user wants to have a closer look at a particular feature in a model, e.g. because he wants to fine-tune its parameters. Using different visualisation techniques for a selected feature and the rest of the model, extra insight into the selected feature is offered, e.g. on its shape and location in the model. For example, the selected feature may be visualised with shaded faces, and the rest of the model as a wire frame or with visible lines only. As already mentioned in Subsection 3.1, also additional feature information, such as closure faces of subtractive features, can be visualised. The facilities for rendering such images make extensive use of the ACIS Modeling Kernel [Spatial 2000], and are therefore not available on the clients. Instead, the images are rendered by the SPIFF modeling system, and sent by the Session Manager to the clients, where they are displayed in a camera; see Figure 2(a).

In addition to the settings for the above-mentioned techniques, several viewing parameters (such as center of projection, view reference point, projection type, etc.) can be set per camera. The camera panels of the webSPIFF GUI offer the clients functionality to specify and modify any camera parameter by means of menus, control boxes and checkboxes; see Figure 2(b) and (c). Interactive specification of the viewing parameters will be elaborated in Subsection 4.2.

The sophisticated image displayed in a camera has to be updated whenever the client modifies any of its camera parameters. Similarly, the image no longer reflects the current state of the model when any session participant has modified the model. In both cases, the image is regenerated on the server and resent to the client(s). Both GIF and JPG image formats provide satisfactory results; see Figure 2(a) for a sophisticated image example. Sending an image from the server to a client is therefore very cheap, both in terms of network load (approximately 10 Kbytes) and display time at the client.

4.2. Interactive visualization model

As described in the previous subsection, changing the viewing parameters of a camera can be done by specifying values for them using the camera panels. This is convenient, for example, when the user wishes to position the viewing camera at an exact location. Often, however, it is much more practical to be able to position and orient the viewing camera in 3D space in an interactive way, using the mouse, as usual in most CAD systems. As the mouse moves, the viewing parameters are modified continuously according to the mouse events generated, creating a smooth animation. Rendering a sophisticated feature model image at the server, and sending it back to a client, takes quite some time, which makes it impossible to update the sophisticated image in real time: the time elapsed between arrival of two successive images at the client would simply be too long, hindering smooth interaction.

The requirement for graphical interaction led to the introduction of a *visualization model*, which is a polygon mesh, generated at the server in VRML format [Ames *et al.* 1997], and sent to the client, where it is loaded into a Java3D scene. Unlike the cellular model on the server, it contains no information about features, except possibly different color attributes for faces originating from different features. Figure 3 shows a camera window with an image of a visualization model.

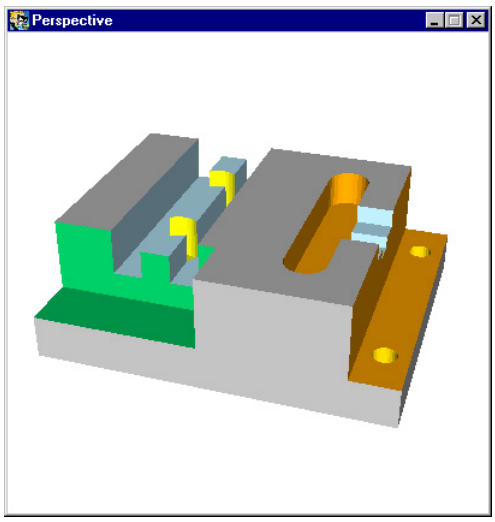
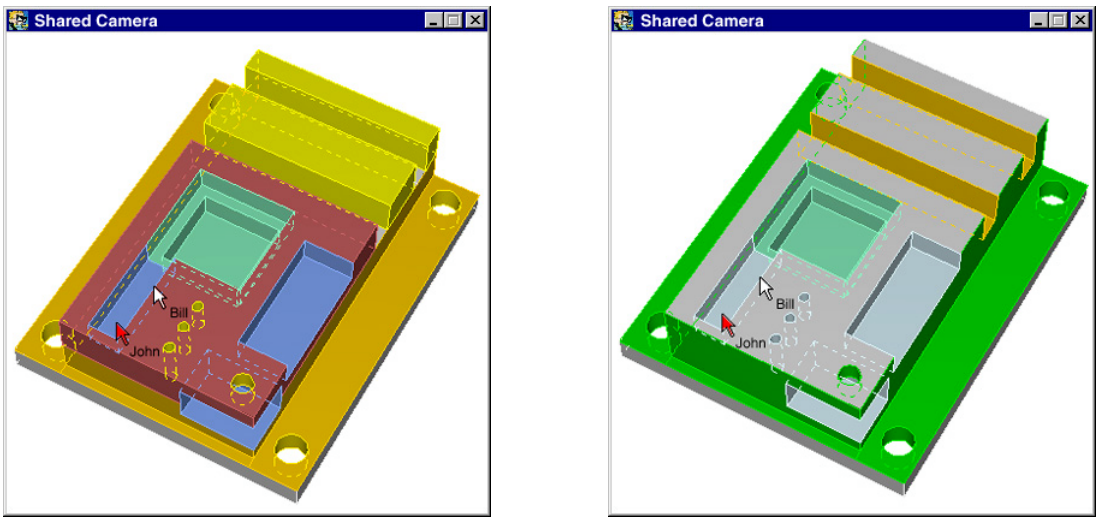


Figure 3. Camera window displaying the visualization model of the part used in Figure 2

By maintaining the visualization model at the clients, viewing cameras can be interactively oriented and positioned in virtual space as follows. As default, a sophisticated feature model image is shown in a camera window, while the visualization model is hidden. When a mouse button is pressed on the camera, the sophisticated feature model image disappears, and the visualization model is displayed instead. The user can then interactively adjust the viewing parameters, until the camera has the desired position and orientation. After being confirmed by the user, the new camera parameters are sent to the server, which in turn generates a new sophisticated feature model image according to the new parameters. This is then delivered back to the client, where it is displayed in the camera window, hiding the visualization model again.

The visualization model only needs to be regenerated by the server and updated at the clients whenever any user modifies the product model. Sending the VRML file to the clients is reasonably cheap in terms of network load (in the order of 100 Kbytes for a moderately complex feature model). Taking into account these file sizes, it can be questioned whether compressing them before transmission to the clients would further improve system throughput, due to the overhead introduced by the compression and decompression algorithms. It would probably be more effective to use techniques for incremental or progressive transmission of the VRML data; see, for example, [Gueziec *et al.* 1999].



(a) design view (b) manufacturing planning view

Figure 4. Shared cameras at two users with different views on the same product model

4.3. Shared cameras

In a collaborative modeling environment, synchronous communication channels play an important role. Among the various techniques available for supporting collaborative modeling work, one of the most effective is the use of *shared cameras*. In a shared camera, several distributed users share the same viewing parameters on the visualized product model. These parameters are permanently synchronized, so that everytime one user modifies any of them, the shared cameras of all other users are automatically updated.

In webSPIFF, shared cameras were very easily implemented, because every client already disposes locally of its own visualization model, introduced in the previous subsection. The only requirement here is the propagation to all users of the changing viewing parameters, so that these can be adjusted for the local shared camera. This is very effectively handled using the Remote Method Invocation (RMI) facilities of Java, via the Session Manager, which receives from any client the modified parameters, and forwards them to the remaining users.

In addition to the above, webSPIFF provides each user of a shared camera with a personalized *telepointer*. The telepointers of all participants in a shared camera session are also constantly updated in all shared cameras. In this way, e.g. when discussing on some local geometric detail (typically using some phone conferencing facility), participants in a shared camera can always precisely trace back where each interlocutor is pointing at.

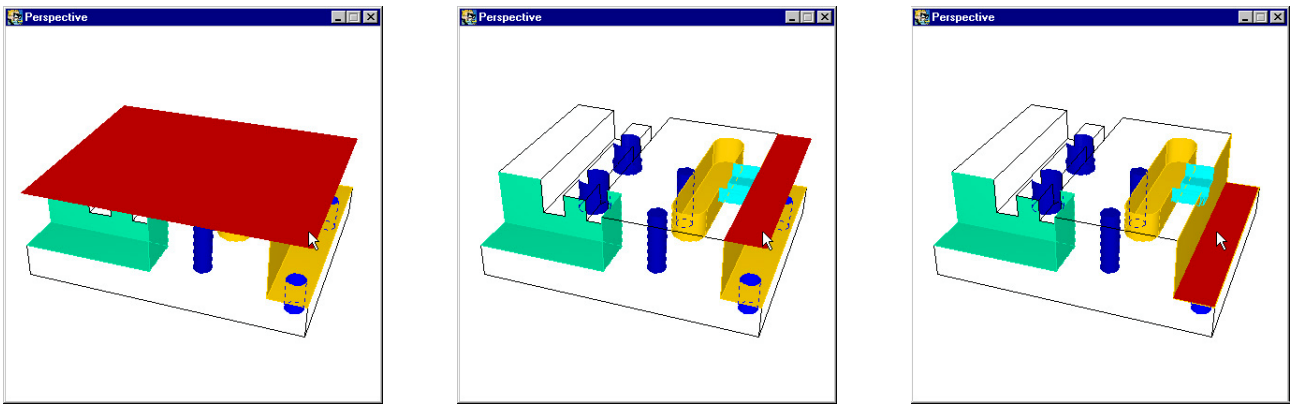
Another advantage of this use of shared cameras within the multiple-view feature modeling framework of webSPIFF is that what each participant sees rendered in his shared camera is (the visualization model of) his own view specific feature model. Figure 4 illustrated this, showing two shared cameras in the same session. The user in (a) is working on the design view, and hence his shared camera visualizes the design view of the product model; the user in (b), however, is working on the manufacturing planning view, and sees therefore the feature model of that view in his shared camera. Although each one 'sees' only feature instances that are meaningful within his own view on the product, the presence of telepointers facilitates focusing their dialogue on a mutual region of interest for their discussion.

5. INTERACTIVE SELECTION OF FEATURE ENTITIES

As explained in Subsection 3.2, an essential characteristic of the SPIFF system is that its modeling operations are specified in terms of features and feature faces. The interface of webSPIFF clients provides a panel for the specification of feature modeling operations, presenting the required menus filled with appropriate names (e.g. of all features, or of all faces of a particular feature). The user can then browse through these names to specify the operands of modeling operations, as he might do when working directly at a CAD station running the SPIFF system.

However, graphical interaction is very useful, not only for visualization of the product model, as described in Section 4, but also for assisting the user in selecting model entities, specifically for modeling operations. In fact, it is often much more convenient to graphically select those entities directly on an image of the visualized product model than from a menu.

For this, the *selection model* was introduced at the web clients. It consists of a set of feature canonical shapes, each of which comprises a number of uniquely named entities, in particular the feature faces. Each canonical shape is generated at the server into a separated file in VRML format, and loaded into the Java3D scene at the client.



(a) (b) (c)

Figure 5. Selection of the step bottom face using the selection model

The canonical shapes in the selection model are never fully displayed simultaneously. Instead, they are kept invisible in the camera, until the user selects a point with the mouse. At that moment, a conceptual ray is determined from the position of the selected point and the viewing parameters used to generate the image. The feature faces intersected by the ray are subsequently highlighted for possible selection, until the user confirms the selection of one of them; see Figure 5 for an example. Notice that in this way also feature faces can be selected that are (partly or totally) not on the boundary of the resulting object, as shown in Figure 5.(b).

When the camera viewing parameters have been modified, the canonical shapes of the selection model do not need to be updated at the client: the only thing needed is to visualize them according to the new viewing transformation, similarly to what is done with the visualization model. A canonical shape only needs to be regenerated by the server, and reloaded by the client, when the parameters or the position of the corresponding feature are modified, as a result of some modeling operation. Sending VRML files of the canonical shapes to the clients is again cheap in terms of network load (in the order of 5 Kbytes per canonical shape).

6. CONCLUSIONS

This paper discussed a number of user interaction facilities suitable for web-based, collaborative feature modeling. These have been implemented in the new collaborative modeling system webSPIFF, which has a client-server architecture. The webSPIFF server runs on a HP B180L Visualise workstation. So far, webSPIFF clients running on Unix, Windows and Linux platforms have successfully participated in collaborative sessions. The only requirement at the client side is a Java/Java3D-enabled web browser. The webSPIFF portal has a demo version available on Internet for users to experiment with, at www.webSPIFF.org.

webSPIFF provides a powerful framework for investigating many issues involved in collaborative feature modeling systems, including synchronization, concurrency and user interaction aspects. The proposed distribution of functionality between the server and the clients has resulted in a well-balanced system. On the one hand, the full functionality of an advanced feature modeling system is offered by the server. On the other hand, all desirable interactive modeling functionality is offered by the clients, ranging from display of sophisticated images of feature models to interactive selection facilities. The Java-based client application is quite simple, and a good compromise between interactivity on the clients and network load has been achieved.

As Internet technology rapidly improves, faster and better collaboration becomes possible. It can therefore be expected that, although the development of collaborative modeling systems is still at its early stages, such systems will soon play an important role in the product development process.

7. REFERENCES

Ames, A., Nadeau, D. and Moreland, J. (1997) The VRML 2.0 Sourcebook. Second Edition, John Wiley & Sons, New York

Autodesk (2000) AutoCAD 2000 Online. Autodesk Inc., San Rafael, CA, USA. *http://www.autodesk.com*

Bidarra, R., van den Berg, E. and Bronsvoort, W.F. (2001) Collaborative modeling with features. TBP in: CD-ROM Proceedings of the ASME 2001 Design Engineering Technical Conferences, Pittsburgh, PA, USA, ASME, New York

Bidarra, R. and Bronsvoort, W.F. (1999) Validity maintenance of semantic feature models. Proceedings of Solid Modeling '99 – Fifth Symposium on Solid Modeling and Applications, Bronsvoort, W.F. and Anderson, D.C (Eds.), ACM Press, NY, pp. 85–96

Bidarra, R. and Bronsvoort, W.F. (2000) Semantic feature modelling. *Computer-Aided Design*, **32**(3): 201–225

Bidarra, R., de Kraker, K.J. and Bronsvoort, W.F. (1998) Representation and management of feature information in a cellular model. *Computer-Aided Design*, **30**(4): 301–313

Bronsvoort, W.F., Bidarra, R. and Noort, A. (2001) Feature model visualization. *Submitted for publication*

Chan, S., Wong, M. and Ng, V. (1999) Collaborative solid modeling on the WWW. Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, CA, pp. 598–602

CoCreate (2000) Shared engineering. http://www.cocreate.com/onespace/documentation/whitepapers/shared_eng.pdf

CollabWare (2000) An introduction to GS–Design Beta. https://www.prodeveloper.net/downloads/whitepaper.pdf

Comerford, R. (2000) Software, piecewise. *IEEE Spectrum*, **37**(2): 60–61

Gueziec A., Taubin, G., Horn, B. and Lazarus, F. (1999) A framework for streaming geometry in VRML. *IEEE Computer Graphics and Applications*, **19**(2): 68–78

Hoffmann, C.M. and Joan-Arinyo, R. (1998) CAD and the product master model. *Computer-Aided Design* **30**(11): 905–918

Kagan, P., Fischer, A. and Bar-Yoseph, P.Z. (1999) Integrated mechanically-based CAE System. Proceedings of Solid Modeling '99 – Fifth Symposium on Solid Modeling and Applications, Bronsvoort, W.F. and Anderson, D.C (Eds.), ACM Press, NY, pp. 23–30. Also in: *Computer-Aided Design*, **32**(8/9): 539–552

Kaon (2001) HyperSpace-3DForum. Kaon Interactive Inc., Cambridge, MA, USA. *http://www.kaon.com*

de Kraker, K.J., Dohmen, M. and Bronsvoort, W.F. (1997) Maintaining multiple views in feature modeling. Proceedings of Solid Modeling '97 – Fourth Symposium on Solid Modeling and Applications, Hoffmann, C.M. and Bronsvoort, W.F. (Eds.), ACM Press, NY, pp. 123– 130

Lee J.Y., Kim, H., Han, S.B. and Park, S.B. (1999) Network-centric feature-based modeling. Proceedings of Pacific Graphics '99, Kim, M.-S. and Seidel, H.-P. (Eds.), IEEE Computer Society, CA, pp. 280–289

Lewandowski, S. (1998) Frameworks for component-based client/server computing. *ACM Computing Surveys*, **30**(1): 3–27

Nam, T.J. and Wright, D.K. (1998) CollIDE: A shared 3D workspace for CAD. Proceedings of the 1998 Conference on Network Entities, Leeds. *http://interaction.brunel.ac.uk/~dtpgtjn/neties98/nam.pdf*

Parametric (2000) Pro/ENGINEER 2001*i*. Parametric Technologies Corporation, Waltham, MA, USA. *http://www. ptc.com*

Stork, A. and Jasnoch, U. (1997) A collaborative engineering environment. Proceedings of TeamCAD '97 Workshop on Collaborative Design, Atlanta, GA, pp. 25–33

Spatial (2000) ACIS 3D Modeling Kernel, Version 6.2. Spatial Technology Inc., Boulder, CO, USA. *http://www.spatial. com*

Sun Microsystems (2000) The Sun Java™ Technology Homepage. *http://java.sun.com*

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Interfaces Caligráficas RISC

João Paulo Pereira
Dep. de Eng.ª Informática
ISEP/INESC
R. de São Tomé, Porto
jpp@dei.isep.ipp.pt

Joaquim A. Jorge
Dep. de Engenharia Informática
IST/UTL
Av. Rovisco Pais, Lisboa
jorgej@acm.org

Vasco Branco
Dep. de Comunicação e Arte
Univ. de Aveiro
Aveiro
vab@ca.ua.pt

F. Nunes Ferreira
Dep. de Eng.ª Electrotécnica e de Computadores
FEUP
R. dos Bragas, Porto
fnf@fe.up.pt

Sumário

Ao aumento do poder de cálculo das plataformas computacionais na última década não têm correspondido ganhos de usabilidade dos programas de desenho assistido por computador, pese embora o sucesso das chamadas interfaces de tampo de secretária (desktop). Tal deve-se ao facto de os modelos de construção de objectos e cenas tridimensionais reflectirem mais as características e virtudes da representação computacional desses objectos e não o ponto de vista do utilizador. No presente artigo abordamos o problema através da simplificação da interface e número de comandos, construindo incrementalmente imagens complexas recorrendo a uma interface caligráfica de repertório reduzido de comandos. Através do sistema GIDeS procuramos demonstrar que um conjunto simples de formas base e comandos de manipulação permite construir cenas surpreendentemente complexas graças a um paradigma de desenho incremental, satisfação inteligente de restrições e reconhecimento de esboços.

Palavras-chave

Técnicas de Interacção, Modelação 3D, Desenho de Esboços, Interfaces Gestuais, Interfaces Caligráficas.

1. INTRODUÇÃO

A evolução dos sistemas de CAD nas últimas décadas caracteriza-se, entre outros aspectos, pelo facto de o notável incremento no poder dos mesmos como ferramentas de apoio ao design ter sido obtido à custa de um indesejável aumento na complexidade da sua utilização e de um consequente distanciamento dos mesmos em relação aos tradicionais papel e lápis [Blinn90]. A profusão de comandos existentes e a rigidez da interacção tendem a intrometer-se na mente do designer, perturbando os processos criativos do mesmo, pelo que não surpreende que este opte pela utilização do papel e do lápis nas fases iniciais do design, recorrendo ao computador apenas nas derradeiras etapas, quando a forma do objecto em concepção já está estabelecida e importa converter o esboço num desenho rigoroso.

Existe por conseguinte um fosso aparentemente intransponível entre o papel e o lápis por um lado, de utilização simples e passiva, e os sistemas de CAD profissionais, complexos e pouco naturais, por outro.

Em 1994 foi apresentado o protótipo IDeS [Branco94], um sistema computacional de apoio ao design que constituiu uma primeira tentativa de substituir com vantagem o papel e o lápis nas fases iniciais e criativas daquele processo. A concepção de IDeS teve como paradigma subjacente o desenho esboçado pelo utilizador, a partir do qual, e em conjugação com operações de construção, era concebido um modelo aproximado do objecto 3D pretendido pelo designer.

Em 2000 apresentámos a primeira versão do sistema GIDeS [Pereira00a] [Pereira00b], uma evolução do sistema de Branco caracterizada, entre outros aspectos, pela substituição da interface manobrada por menus por uma interface caligráfica [Jorge94] [Jorge00] baseada no desenho de esboços, em que é adoptada uma forma inovadora e não intrusiva[1] de lidar com a ambiguidade e a imprecisão inerentes aos processos humanos de transmissão de informação, convertendo-as em mais valias que levam a uma maior aproximação dos sistemas de CAD aos tradicionais papel e lápis. As listas de expectativas

[1] Este modo não intrusivo de apresentar ao utilizador informação dependente do contexto, sugerindo mas não impondo um leque de acções alternativas, e gerindo com eficácia eventuais ambiguidades, foi denominado de listas de expectativas [Pereira00a].

reduzem também a carga cognitiva imposta ao utilizador, na medida em que permitem reduzir o conjunto de instruções – gestos de comando – do sistema, abrindo o caminho para uma nova classe de interfaces a que demos o nome de RISC – *Reduced Instruction Set Calligraphic Interfaces* [Pereira01].

A concepção da nova versão do sistema GIDeS tem como paradigma o que nós designamos por desenho incremental, um processo construtivo que tem como objectivo a transposição do já referido hiato entre o desenho puro de esboços, por um lado, e a abordagem tradicional dos sistemas computacionais de apoio ao design actualmente existentes, por outro.

Tendo como base o desenho, o sistema GIDeS continua a ser adequado para as fases iniciais do design, na medida em que se aproxima, na sua facilidade de utilização, do conjunto papel/lápis tão do agrado dos designers. No entanto esta nova versão inclui algumas inovações que permitem a concepção rigorosa de modelos de objectos 3D, sem pôr em causa a simplicidade e a naturalidade dos processos de interacção intuitiva.

A precisão requerida nas fases finais do design é obtida com o recurso a uma permanente assistência ao utilizador por parte do sistema, assistência essa que decorre a todos os níveis do processo de edição[2] e que consiste em sugerir restrições que, a serem aceites, convertem os esboços imprecisos do utilizador em desenhos técnicos rigorosos. Para além disso, as restrições implementadas especificamente no âmbito da edição 3D podem ser complementadas com as restrições inerentes à edição 2D, mediante a utilização de linhas de construção.

Neste artigo procedemos à descrição das funcionalidades e aperfeiçoamentos introduzidos na nova versão do sistema GIDeS, comparando-o com outros trabalhos que têm vindo a ser desenvolvidos nesta área e apresentando a nossa abordagem ao processo de construção de desenhos precisos a partir de esboços ambíguos, sem pôr em causa as reconhecidas vantagens que são inerentes à utilização do papel e do lápis. São também apresentadas ilustrações de alguns objectos rigorosos construídos aquando da avaliação preliminar da usabilidade do nosso sistema. Por último, fazemos uma breve descrição da investigação em curso e do trabalho a realizar futuramente.

2. TRABALHO RELACIONADO

São em número significativo os trabalhos que têm vindo a ser realizados nos últimos anos no âmbito dos sistemas gestuais aplicados à modelação 3D e a outros domínios. O protótipo GIDeS foi já objecto de um estudo comparativo com alguns destes trabalhos, o qual pode ser encontrado em [Pereira00b]. De entre os trabalhos analisados constam o sistema SKETCH de Zeleznik *et al.* [Zeleznik96], o protótipo Jot de Forsberg *et al.* [Forsberg97], o *bloco-notas* translúcido desenvolvido por Encarnação *et al.* [Encarnação99], o sistema Teddy de Igarashi *et al.* [Igarashi99] e o protótipo Pegasus de Igarashi *et al.* [Igarashi97].

Mais recentemente Fonseca *et al.* desenvolveram o sistema $C_{AL}I$, uma biblioteca de componentes para a construção de interfaces caligráficas que tem como ponto de partida um núcleo de reconhecimento de formas geométricas e comandos gestuais [Fonseca00]. O algoritmo de reconhecimento funciona com base na determinação de características geométricas simples e recorre à utilização de lógica difusa para lidar com ambiguidades.

Mankoff *et al.* passam em revista algumas das interfaces existentes cujo funcionamento se baseia nalgum tipo de reconhecimento[3], e dão um especial destaque para o problema da ambiguidade e para as técnicas utilizadas na sua resolução [Mankoff00b]. Com base neste estudo desenvolveram um *toolkit* de interfaces com o utilizador designado por OOPS – *Organized Option Pruning System* – que consiste essencialmente numa biblioteca de técnicas reutilizáveis de correcção de erros – denominadas técnicas de mediação – complementadas com a infra-estrutura necessária ao tratamento da ambiguidade [Mankoff00a].

Gross *et al.* desenvolveram um projecto designado por BoE – *Back of an Envelope* – que recorre à utilização de interfaces caligráficas nos mais variados domínios de aplicação como sejam as bases de dados, os programas de simulação, a modelação 3D, etc. [Gross00]. A abordagem utilizada pretende combinar as virtudes das interfaces altamente estruturadas utilizadas nos programas de desenho e modelação rigorosos, por um lado, com a liberdade e flexibilidade das interfaces que possibilitam o desenho à mão livre, por outro.

Turner *et al.* desenvolveram um sistema de modelação designado por Stilton que permite a construção de modelos de objectos tridimensionais numa projecção em perspectiva ou em fotografias panorâmicas mapeadas na cena como sendo o "chão" e as "paredes" [Turner00]. A informação geométrica necessária ao processo de reconstrução é obtida a partir dos esboços feitos pelo utilizador num plano imaginário situado à sua frente.

3. GIDES: SITUAÇÃO ACTUAL

A nova versão do sistema GIDeS sofreu melhoramentos e inclui agora novas funcionalidades no sentido de tornar o sistema adequado a todas as etapas do design, desde as fases criativas iniciais, à edição rigorosa dos objectos resultantes dos esboços imprecisos do utilizador.

[2] Desde o desenho de esboços – edição 2D – à construção e transformação de modelos de objectos tridimensionais complexos – edição 3D.

[3] Não só gestual mas também da fala e da escrita, entre outros.

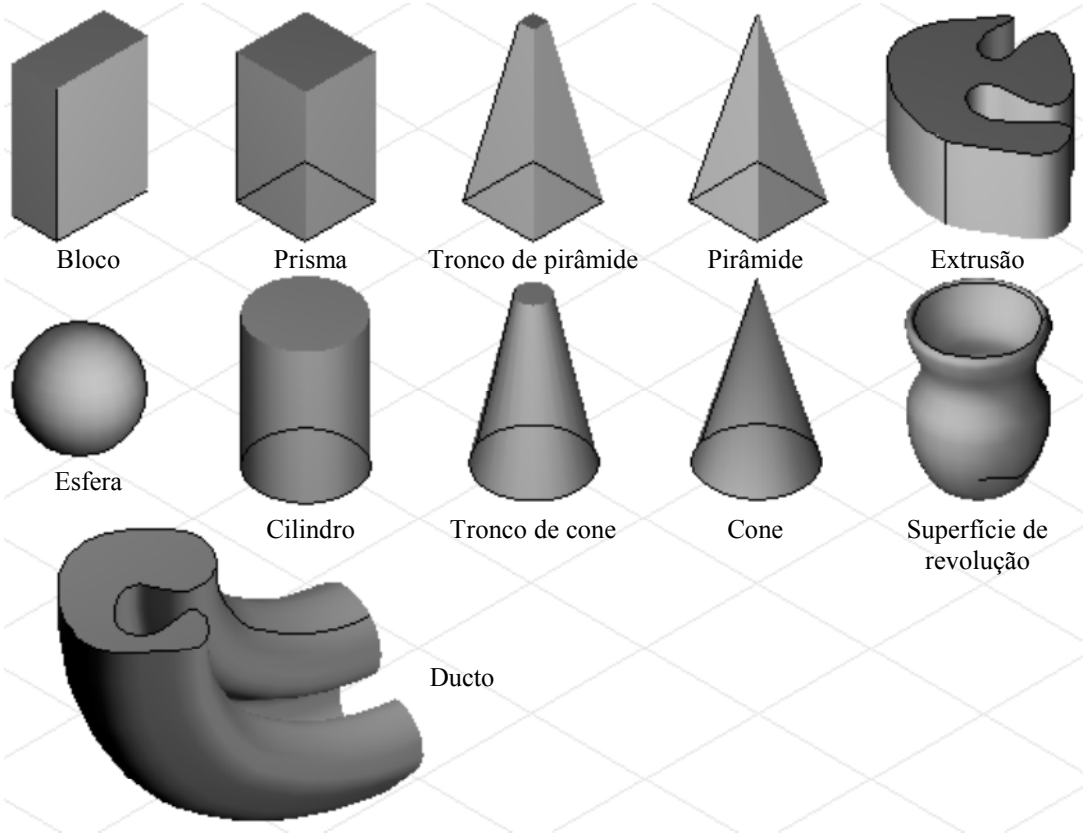


Figura 1 – Primitivas 3D e gestos correspondentes

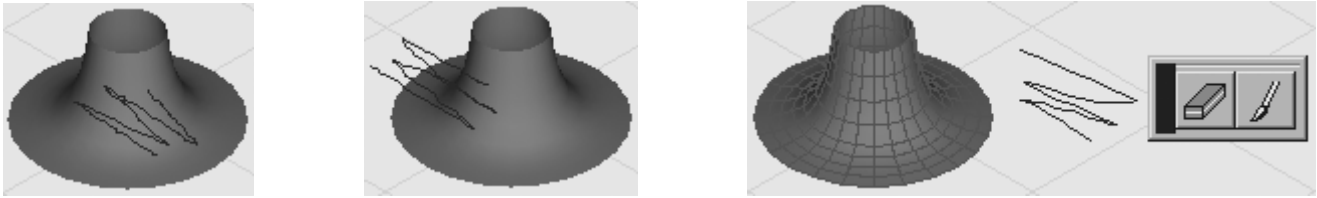
3.1 Reconhecimento e listas de expectativas

A arquitectura básica da interface caligráfica, descrita em [Pereira00b], foi objecto de vários aperfeiçoamentos, nomeadamente ao nível dos módulos de reconhecimento de gestos e das listas de expectativas.

Foi acrescentada uma nova primitiva – o ducto – ao conjunto de primitivas 3D já existente (Figura 1).

Aos dois sistemas de reconhecimento de gestos de comando e ao processo topológico e geométrico de classificação dos gestos correspondentes às primitivas 3D, já existentes na primeira versão, foram acrescentados mais dois módulos, o primeiro responsável pelo reconhecimento de primitivas de desenho – circunferências, elipses, linhas curvas genéricas, segmentos de recta e linha poligonais – e o segundo incumbido de identificar as situações em que pode ser desejável a execução de uma operação booleana entre sólidos e inferir, na maioria dos casos, qual das operações disponíveis é a mais adequada ao contexto. Os novos módulos funcionam em conjugação com o mecanismo de geração de listas de expectativas, o qual foi objecto de uma profunda remodelação ao nível da apresentação gráfica: com excepção das listas de comandos, a utilização de ícones dependentes do contexto foi preterida em favor de representações em escala reduzida dos elementos de desenho e das primitivas 3D que vão sendo sugeridos no decurso das acções do utilizador.

A Figura 2 ilustra um exemplo de como as listas de expectativas permitem reduzir significativamente o número de gestos de comando reconhecidos pelo sistema – interfaces RISC – e limitar, por conseguinte, a carga cognitiva imposta ao utilizador. Neste caso temos dois comandos – o de apagar e o de aplicar uma textura a um sólido – que partilham o mesmo gesto de "riscar".



a) apagar **b) aplicar/remover uma textura** **c) ambiguidade de comandos**

Figura 2 – Lista de expectativas de comandos

A diferença reside no facto de o traço de apagar ter de intersectar a fronteira correspondente à projecção do objecto (Figura 2a), por oposição ao traço de textura que tem de ser inteiramente desenhado sobre a superfície da referida projecção (Figura 2b). O designer pode também optar por apagar ou aplicar uma textura a um conjunto previamente seleccionado de sólidos. Nestas circunstâncias o sistema GIDeS não dispõe de informação contextual suficiente para identificar o comando pretendido, pelo que é gerada uma lista de expectativas de comandos (Figura 2c).

A Figura 3 mostra como uma lista de expectativas pode fazer uso da ambiguidade intrínseca aos processos de reconhecimento gestual e transformá-la numa mais valia para o utilizador. O sistema de reconhecimento de primitivas de desenho começa por classificar o traço como sendo suave ou anguloso, após o que a lista de expectativas adequada ao contexto é gerada com as diversas opções ordenadas em função da classificação que foi feita. No exemplo o designer esboçou um traço suave muito semelhante a uma elipse, e a primeira sugestão da lista consiste efectivamente naquela primitiva, mas a verdade é que existe a possibilidade de o utilizador pretender uma linha curva genérica a qual, sendo parecida, é não obstante diferente de uma elipse. A segunda sugestão da lista, uma *spline* fechada, vem de encontro às expectativas do designer. A terceira opção – uma linha poligonal fechada – prevê a hipótese de o sistema ter interpretado incorrectamente o desejo do utilizador esboçar uma linha suave, pelo que também é sugerida a correspondente linha angulosa.

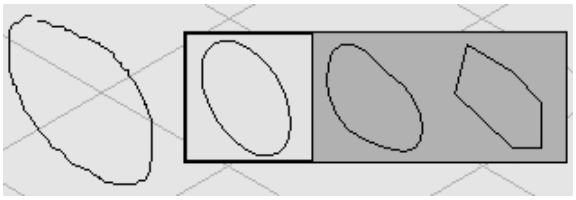


Figura 3 – Lista de expectativas de primitivas de desenho

Este exemplo ilustra também como o emprego de listas de expectativas veio resolver um dos problemas mais delicados dos sistemas de reconhecimento gestual. Estes processos de classificação não devem, por um lado, ser demasiado rigorosos, sob pena de a taxa de ocorrência de falsos negativos ser elevada. Por outro lado o sistema não deve, de igual modo, ser demasiado tolerante, caso contrário a taxa de incidência de falsos positivos aumentará significativamente. A utilização de listas de expectativas permite que o módulo de reconhecimento seja ajustado no sentido de ser amplamente tolerante à imprecisão dos traços esboçados, sendo que a ocorrência de eventuais falsos positivos não vai prejudicar o utilizador, uma vez que este pode pura e simplesmente ignorar as sugestões que não lhe convêm e prosseguir com o seu trabalho.

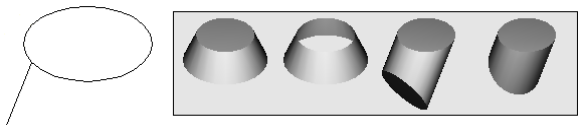


Figura 4 – Lista de expectativas de primitivas 3D

Na Figura 4 está representada uma lista de expectativas de primitivas 3D. Atente-se uma vez mais como as interfaces RISC permitem que o mesmo gesto seja interpretado de diferentes maneiras, no exemplo da figura como um tronco de cone, uma superfície de revolução e dois ductos com orientações distintas. Observe-se também o aspecto gráfico das diversas opções da lista, em que os ícones existentes na primeira versão do sistema GIDeS foram substituídos por modelos em escala reduzida das primitivas em causa.

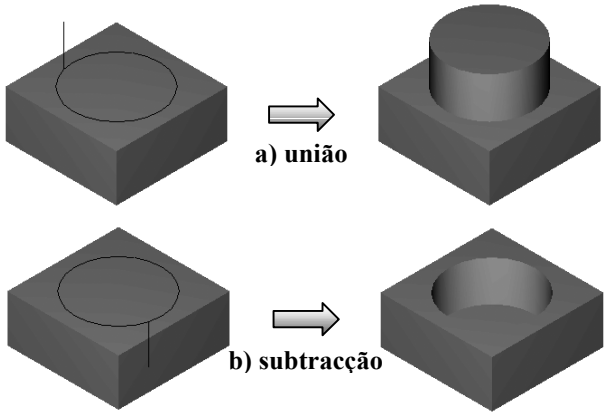


Figura 5 – Identificação automática de operações booleanas

Sempre que o designer esboça uma primitiva sobre um sólido já existente na cena, o novo objecto é devidamente colocado e ligado ao primeiro. O sistema tenta nestas circunstâncias identificar a operação booleana – união ou subtracção – adequada ao contexto, com base na orientação do traço desenhado pelo utilizador (Figura 5). Certas primitivas, como é o caso da esfera, não são orientadas, o que impede a identificação da operação booleana pretendida. Neste caso é gerada uma lista de expectativas que permite ao utilizador resolver a indeterminação (Figura 6).

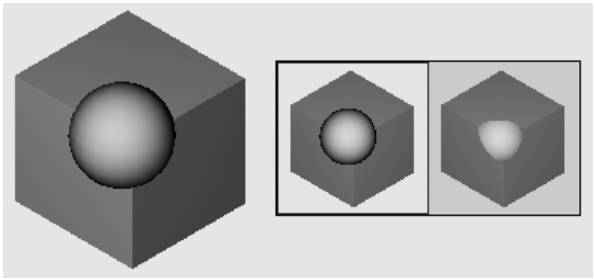


Figura 6 – Lista de expectativas de operações booleanas

3.2 Edição 2D: Correcção de esboços

O sistema GIDeS permite ao utilizador corrigir a forma dos traços esboçados. Este, à semelhança do que os designers costumam fazer com o papel e lápis, limita-se a desenhar directamente sobre a secção da curva que

pretende alterar, encarregando-se o módulo de edição 2D de identificar e apagar a porção indesejada (Figura 7).

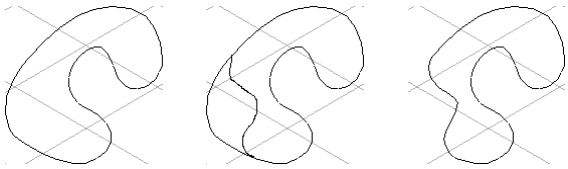


Figura 7 – Correcção de esboços

3.3 Edição 3D

O sistema GIDes oferece ao utilizador um conjunto de funcionalidades que lhe permitem editar objectos tridimensionais de uma forma simples e eficiente. Para além de um *clipboard*, existem quatro modos de interacção, seleccionáveis por meio de uma lista de expectativas (Figura 8), dos quais três estão associados a transformações geométricas de corpo rígido e um ao corte de objectos.

Figura 8 – Operações de edição 3D

3.3.1 Clipboard

O *clipboard* permite efectuar as tradicionais operações de corte, cópia e colagem de objectos tridimensionais[4].

3.3.2 Transformações geométricas

A nossa abordagem à tarefa de efectuar transformações geométricas de corpo rígido – translações e rotações – é diferente da que é habitual encontrar nos sistemas de CAD existentes comercialmente. Em vez de deduzir as transformações necessárias à realização da tarefa que tem em mente, o utilizador dispõe de três modos de interacção distintos os quais, mediante o desenho de simples traços, habilitam o sistema a inferir, com base em determinadas restrições[5], as transformações que devem ser levadas a cabo.

3.3.2.1 Colagem

Neste modo de interacção, o utilizador desenha uma linha a ligar dois sólidos e o sistema efectua o conjunto de transformações necessário para ligar – colar – o primeiro objecto ao segundo pelas faces especificadas (Figura 9). Sólidos eventualmente colados ao primeiro sofrem o mesmo conjunto de transformações, de forma a manterem-se colados.

[4] Não obstante as designações em português serem as mesmas, convém não confundir as operações do *clipboard* com os modos de corte e colagem de objectos descritos nas secções que se seguem.

[5] As restrições envolvidas são de três tipos: coincidência dos planos de faces, alinhamento de arestas e coincidência de vértices.

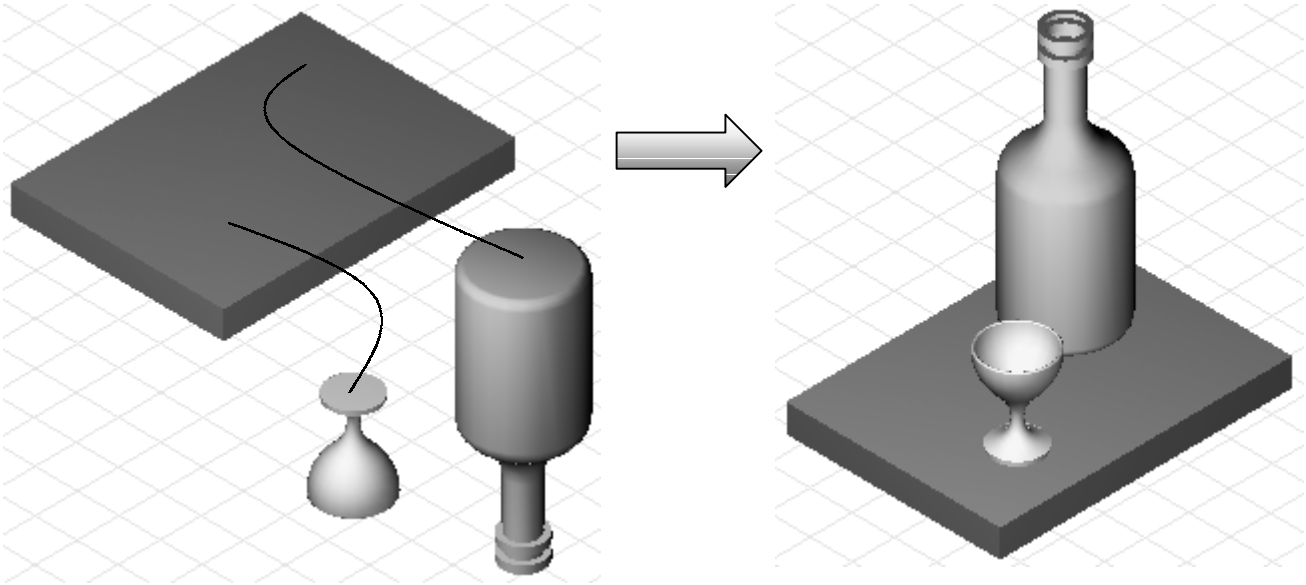
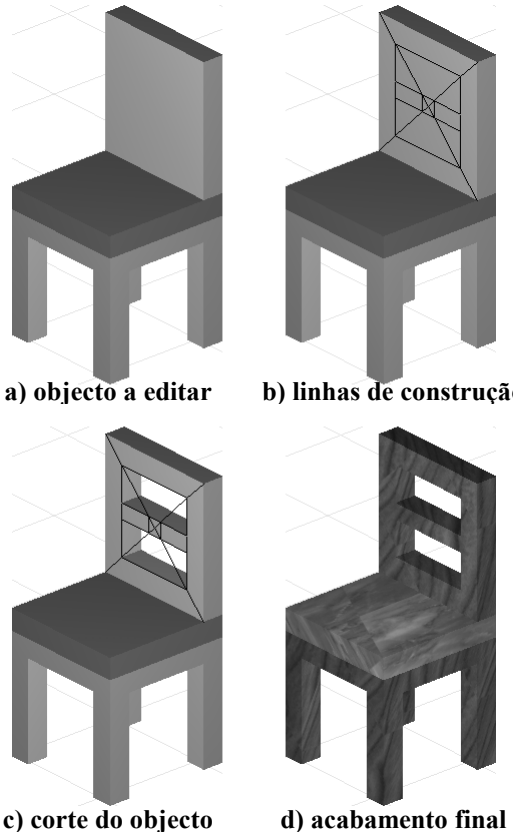


Figura 9 – Colagem de objectos



a) objecto a editar **b) linhas de construção**

c) corte do objecto **d) acabamento final**

Figura 10 – Utilização de linhas de construção no corte rigoroso de objectos

3.3.2.2 Ajustamento

Quando um sólido está colado a outro, este modo permite ao utilizador ajustar a posição do primeiro em relação ao segundo. O sistema identifica as restrições a aplicar ao processo de translação, de modo a que o objecto possa apenas deslizar ao longo do plano da face à qual está ligado. Outros sólidos eventualmente colados ao objecto são alvo da mesma transformação.

3.3.2.3 Deslocamento

Este modo de interacção permite ao utilizador colocar sem restrições um objecto na cena. O processo de translação é também aplicado a outros sólidos eventualmente colados ao primeiro. Um mecanismo de detecção permite o posicionamento do objecto em cima de outros sólidos já existentes.

3.3.3 Cortes

O sistema GIDeS põe à disposição do designer um modo de interacção que lhe permite efectuar com simplicidade e eficácia cortes de objectos tridimensionais. Este limita-se a desenhar o perfil do corte pretendido, encarregando-se o sistema de determinar, construir e subtrair um sólido de extrusão ao objecto em causa, de modo a obter o efeito desejado. Os perfis de corte podem ser abertos ou fechados. A Figura 10 ilustra as fases finais do processo de construção rigorosa de uma cadeira, sendo que na terceira (Figura 10c) se procede ao corte do bloco que irá constituir as costas do objecto. A figura exemplifica também a utilização de linhas de construção (Figura 10b) que vão servir de guia ao utilizador no delineamento preciso dos perfis de corte.

3.4 Operações de câmara

Não obstante estarem ainda disponíveis os gestos de comando que permitem efectuar operações de câmara, existe agora um processo alternativo de manipulação directa que é activado sempre que o utilizador pressiona o botão lateral do estilete. O sistema gera uma lista de expectativas a qual permite escolher a operação – *pan*, *zoom* e *viewpoint* – pretendida, bem como restaurar a isometria (Figura 11).

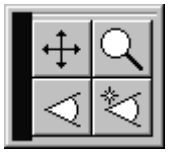


Figura 11 – Operações de câmara

4. CONCLUSÕES E TRABALHO FUTURO

A avaliação formal de usabilidade da nova versão do sistema GIDeS está ainda em curso. No entanto foram já realizados testes preliminares junto de uma pequena amostra de designers, e os comentários e resultados obtidos são deveras encorajadores. As Figuras 12 e 13 ilustram alguns exemplos de objectos rigorosos razoavelmente complexos realizados aquando dos referidos testes. Com base nas sugestões que nos foram feitas estamos a considerar o desenvolvimento do protótipo em duas orientações distintas. A primeira tem a ver com a implementação de uma camada suplementar de desenho – *layer* – reservada para as linhas de construção. Actualmente o sistema comporta apenas duas camadas, uma de desenho 2D e outra de objectos 3D, pelo que não lhe é possível discernir as linhas de construção dos restantes elementos de desenho. Isto tem o inconveniente de obrigar o utilizador a apagar manualmente as referidas linhas assim que estas deixam de lhe ser úteis. A existência de uma terceira camada resolve eficazmente este problema.

A segunda linha de investigação que está a ser seguida tem a ver com o alargamento do leque de restrições disponibilizado pelo sistema. Actualmente apenas algumas restrições relativamente simples – gravidades linear, angular e radial, coincidência de faces, arestas e vértices – foram implementadas. Restrições mais complexas como a simetria, a reflexão e outras estão também a ser analisadas.

Figura 12 – Candeeiros

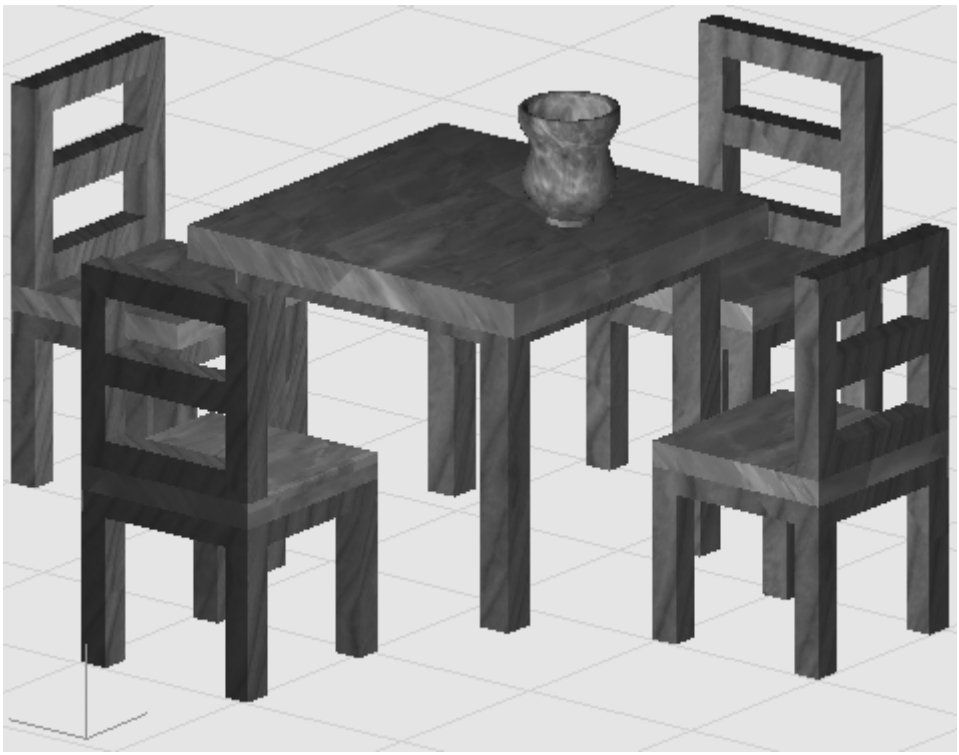


Figura 13 – Mesa, cadeiras e vaso

5. REFERÊNCIAS

[Blinn90] Blinn J F: Jim Blinn's Corner - The Ultimate Design Tool, *Computer Graphics & Applications*, *IEEE*, Vol. 10, No. 11, pp. 90 – 92, 1990.

[Branco94] Branco V, Ferreira F N, Costa A: Sketching 3D models with 2D interaction devices, *EUROGRAPHICS '94 Conference Proceedings*, Daehlen M, Kjelldahl L (editors), Oslo, Blackwell Pub., pp. 489 – 502, 1994.

[Encarnação99] Encarnação L M, Bimber O, Schmalstieg D, Chandler S D: A Translucent Sketchpad for the Virtual Table Exploring Motion-based Gesture Recognition, *Computer Graphics Forum*, Vol. 18, No. 3, pp. C-277 – C-285, 1999.

[Fonseca00] Fonseca M J, Jorge J A: $C_{AL}I$: Uma Biblioteca de Componentes para Interfaces Caligráficas, *Actas do 9.º Encontro Português de Computação Gráfica*, pp. 93 – 100, Fev. 2000.

[Forsberg97] Forsberg A S, LaViola Jr. J J, Markosian L, Zeleznik R C: Seamless Interaction in Virtual Reality, *Computer Graphics & Applications*, *IEEE*, Vol. 17, No. 6, pp. 6 – 9, 1997.

[Gross00] Gross M D, Do E Y-L: Drawing on the Back of an Envelope: a framework for interacting with application programs by freehand drawing, *Computers & Graphics*, Vol. 24, No. 6, pp. 835 – 849, Elsevier, Dec. 2000.

[Igarashi97] Igarashi T, Matsuoka S, Kawachiya S, Tanaka H: Interactive Beautification: A Technique for Rapid Geometric Design, *Proceedings of the ACM Symposium on User Interface Software Technology (UIST)*, 1997.

[Igarashi99] Igarashi T, Matsuoka S, Tanaka H: Teddy: A Sketching Interface for 3D Freeform Design, *SIGGRAPH '99 Conference Proceedings*, *ACM*, 1999.

[Jorge94] Jorge J A: Parsing Adjacency Grammars for Calligraphic Interfaces, PhD Thesis, Rensselaer Polytechnic Institute, Troy, New York, 1994.

[Jorge00] Jorge J A, Glinert E P: Calligraphic Interfaces: towards a new generation of interactive systems, Jorge J A, Glinert E P (guest editors)*, Computers & Graphics*, Vol. 24, No. 6, pp. 817, Elsevier, Dec. 2000.

[Mankoff00a] Mankoff J, Hudson S E, Abowd G D: Providing integrated toolkit-level support for ambiguity in recognition-based interfaces, *Proceedings of ACM CHI'00 Conference on Human Factors in Computing Systems*, pp. 368 – 375, 2000.

[Mankoff00b] Mankoff J, Abowd G D, Hudson S E: OOPS: a toolkit supporting mediation techniques for resolving ambiguity in recognition-based interfaces, *Computers & Graphics*, Vol. 24, No. 6, pp. 819 – 834, Elsevier, Dec. 2000.

[Pereira00a] Pereira J P, Jorge J A, Branco V, Ferreira F N: Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons, *The 8-th International Conference in Central Europe on Computer Graphics, Visualization and Interactive Digital Media 2000*, Plzen, Czech Republic, Feb. 2000.

[Pereira00b] Pereira J P, Jorge J A, Branco V, Ferreira F N: GIDeS: Uma Abordagem Caligráfica à Edição 3D, *Actas do 9.º Encontro Português de Computação Gráfica*, pp. 101 – 108, Fev. 2000.

[Pereira01] Pereira J P, Jorge J A, Branco V, Ferreira F N: Reduced Instruction Set Calligraphic Interfaces: Sketching Complex 3D Objects with (Fewer) Gestures, d^3 desire designum design, 4th European Academy of Design Conference Proceedings, pp. 194 – 196, Aveiro, Portugal, April 2001.

[Turner00] Turner A, Chapman D, Penn A: Sketching space, *Computers & Graphics*, Vol. 24, No. 6, pp. 869 – 879, Elsevier, Dec. 2000.

[Zeleznik96] Zeleznik R C, Herndon K P, Hughes J F: SKETCH: An Interface for Sketching 3D Scenes, *SIGGRAPH '96 Conference Proceedings*, *ACM*, Vol. 30, No. 4, pp. 163 – 170, 1996.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Vision-Based Interaction within a Multimodal Framework

Vítor Sá [1,2] Cornelius Malerczyk [1] Michael Schnaider [1]

[1] Computer Graphics Center (ZGDV)
Rundeturmstraße 6
D-64283 Darmstadt

[2] University of Minho (UM)
Campus de Azurém
P-4800-058 Guimarães

{vitor.sa,cornelius.malerczyk,michael.schnaider}@zgdv.de

Abstract
Our contribution is to the field of video-based interaction techniques and is integrated in the home environment of the EMBASSI project. This project addresses innovative methods of man-machine interaction achieved through the development of intelligent assistance and anthropomorphic user interfaces. Within this project, multimodal techniques represent a basic requirement, especially considering those related to the integration of modalities. We are using a stereoscopic approach to allow the natural selection of devices via pointing gestures. The pointing hand is segmented from the video images and the 3D position and orientation of the forefinger is calculated. This modality has a subsequent integration with that of speech, in the context of a multimodal interaction infrastructure. In a first phase, we use semantic fusion with amodal input, considering the modalities in a so-called late fusion state.

Keywords
EMBASSI project, 3D deictic gestures, multimodal man-machine interaction, agent-based systems.

1. INTRODUCTION

EMBASSI is the name of a joint project sponsored by the German government (BMBF), which began in the summer of 1999, addressing innovative methods of *man-machine interaction* (MMI). In the broad area of information interfaces, the term MMI reminds us that computers are gradually infiltrating more and more of the machinery and equipment commonly used in our daily life.

In today's man-machine interaction, computer input and output are quite asymmetric. The amount of information or bandwidth that is communicated from computer to user is typically far larger than the bandwidth from user to computer [Jacob96]. Since this unbalance often influences both the intuitiveness and performance of user interaction, one of the EMBASSI benefits will be the enhancement of bandwidth from the user to the system.

While a computer *output* presentation over multiple channels has become familiar to us under the designation of *multimedia*, the input channels or sources, also called *input modes* or *modalities*, are the basis of those kinds of applications said to support *multimodal human-computer interaction*.

Our contribution is to the field of video-based interaction techniques and is integrated in the home environment of EMBASSI. In this paper, we are only considering the gesture modality, more concretely the gestural typology of deictic (pointing) gestures. A related study can be founded in [Kohler96].

The following Section 2 describes the EMBASSI project as the multimodal framework where we are applying our vision-based interaction techniques, specifically hand pointing gestures. Section 3 presents in some detail the necessary calibration, as well as the recognition and tracking system. Section 4 outlines the integration of the gesture modality with the others in the context of a multimodal interaction. In section 5, we present some conclusions.

2. FRAMEWORK

Our work is integrated in the EMBASSI framework - described in more depth in [Hildebrand00]. In EMBASSI, innovative interaction technology will be achieved through the development of intelligent assistance and anthropomorphic interfaces. Telecommunication and network infrastructure is used in order to pro-

vide an added value opponent to stand-alone systems with regard to usability and functionality.

2.1 Goals

The overall objective of EMBASSI is the support of humans during interaction with different kinds of technical infrastructures in everyday life. Although the foreseen achievements of EMBASSI can be applied to devices in industrial and office environments, the scope of EMBASSI is directed towards the private sector, including home, car and public terminal applications.


Figure 1 – EMBASSI application areas

The aim of the EMBASSI specification should lead to a new user paradigm in private application areas:

- Transition of the paradigm „device" to the paradigm „system", where user expectation in terms of environmental knowledge of the interaction is incorporated. The basic groundwork is provided by network technology, which allows the inquiry of certain system and device states (e.g. TCP/IP/IEEE802.3(11), IEEE1394/ HAVi).

- Transfer from unimodal to polymodal input and output. In addition to the use of speech and pointing gestures in private home applications, an anthropomorphic graphical output will be addressed.

Figure 2 – Anthropomorphic user interface

The development of appropriate assistance technology is a primary objective of EMBASSI. This includes the elaboration of a uniform approach for the systematic development of user interfaces and assistance systems by the development of:

- *Modular building blocks* of interaction basic technology for a natural and intuitive man-machine interaction;

- A generic architectural framework, including an adequate semantic protocol for the realization of assistance systems based on interoperable components.

2.2 Architecture and protocol

The aim of the intended generic architecture is to provide the backbone of the different EMBASSI derivatives. Fundamental objectives of the architecture are:

- Homogenize the different application scenarios, especially with respect to the protocol and interfaces;

- Provide a common understanding of the interfaces and modules.

The complex interaction process of intelligent assistance in a multimodal manner, where the system consists of diverse technology components (from the recognizers and multimodal integrators, to the context and dialogue managers), results in very complex information processing.

One way to realize this information processing flow as an architecture is to pipeline the components via procedure calls -- or remote procedure calls -- in the case of a distributed but homogeneous system (in programming language). For distributed and heterogeneous software, this may prove difficult and the solution goes through *agent-based software engineering*. In essence, the several system components are "wrapped" by a layer of software that enables them to communicate via a standard language over TCP/IP. The communication is then processed directly based on some concepts of distributed systems, like asynchronous delivery, triggered responses and multi-casting, or, alternatively, by using a *facilitated* form. In EMBASSI, a facilitated ("hub-spoken") multi-agent architecture is being used. Only when unavoidable, due the possible bottleneck derived from high-volume multimedia data transfer, can this approach be "by-passed".

To integrate all the independent and heterogeneous system components, the widely used KQML (Knowledge Query and Management Language) was chosen as the agent communication language (ACL). This decision was based on the effectiveness of KQML regarding the communication between agent-based programs. It provides high-level access to information and can be used for low-level communication tasks, such as automatic error checking.

KQML is complementary to distributed computing approaches (e.g. OMG CORBA/IIOP), whose focus is on the transport level (how agents send and receive messages). The focus of KQML is the "language level" – the meaning of the individual messages. Furthermore, in order to successfully interoperate, the agent-based system must also agree at the "policy level" (how agents structure conversations) and at the "architecture level" (how to connect systems in accordance with constituent protocols) [Finin93].

KQML is a language for representing *communicative acts* - for programs to communicate attitudes about information. KQML has a base definition with the possibility of being extended, which allows agents to use so-called *performatives* (the name of KQML messages) that do not appear in the standard specification.

One of the most positive characteristics of this ACL is the separation from the language used to code its *contents*. This has been demonstrated to be a good practice in the context of agent communication. The EMBASSI project, not being restricted to KQML, follows this advised distinction and, therefore, is using XML (Extensi-

ble Markup Language) [W3C00] for syntax definition of the message content, and Description Logics (DL) [Domini97] for knowledge representation. The DL language is used to build sets of concepts and roles called ontology [Franconi99], the way the lower level modules of the architecture have to provide knowledge to the more universal ones.

To this end, we found several software tools to build agent-based systems. Some examples are OAA [Martin99], Jackal [Cost99] and JATLite [Jeon00]. The last one was made open-source software, available under the GNU general public license since December 1998.

2.3 Modalities

EMBASSI has an open and modular architecture with an amodal treatment of the different modalities at the level of the dialogue manager. This permits the inclusion of as many modalities as needed to perform the intended advanced interaction.

The speech modality itself is treated in two separate modules: one is called the *speech recognizer* and the other the *speech analyzer*. The first detects the different morphemes of the sentences (based on the possible allophones and the basic phonemes) and sends a *word hypotheses graph* to the next module to be analyzed. Consequently, the analyzer module, based on the received structure, determines the semantic information concerning the utterance in cause, always considering the EMBASSI-defined ontologies.

The video-based components are integrated in home, car, and public terminal scenarios. Different requirements concerning illumination, camera hardware, accuracy, and reliability are considered in various scenarios. Video-based input encompasses a large spectrum of modalities from *gestures*, *facial expressions* and *emotions*, to *lip-reading*, *eye tracking*, and *stick pointing*.

3. VISUAL INPUT

Milota and Blatner [Milota95] have defined "a taxonomy of gesture with accompanying voice". As mentioned at the beginning, we are only considering the deictic gestures typology.

The video-based interaction we are describing uses a stereoscopic approach and allows a natural selection of devices via a pointing gesture. This can be combined with a speech recognition component to enhance the independent unimodal inputs by an integrated multimodal approach. The pointing hand is segmented from the video images and the 3D position and orientation of the forefinger is calculated. Afterwards, the selected device is identified by using the known camera parameters and device positions.

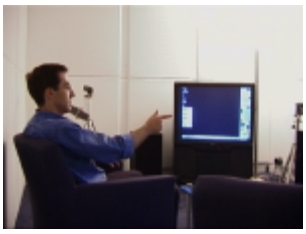


Figure 3 – Gesture-based interaction

The recognition of the pointing gesture is based on a template matching method. The gesture is described by predefined landmark points on the boundary of the object (see figure 4). A statistical description of the gesture in nature is calculated in combination with its modes of variation. This *Point Distribution Model* [Cootes00] of the hand is used for detecting and recognizing the gesture in gray level images. The system set up for gesture recognition contains modules for camera calibration of a stereoscopic camera system, data acquisition, image pre-processing, object recognition, object fitting, 3D pose estimation and communication. The modules and their relationship are described below.

Figure 4 – Model of the pointing gesture

3.1 Image pre-processing

After the acquisition of two corresponding gray level images, image pre-processing is necessary. To suppress disturbing noise, averaging with a Gaussian convolution mask smoothes the images [Haberäcker95]. In the case of bad lighting conditions, performing histogram equalization enhances the image contrast. Furthermore, the edge information is extracted by applying various edge detection algorithms like the Sobel operator or the Canny edge detector [Sonka98] (see figure 5).

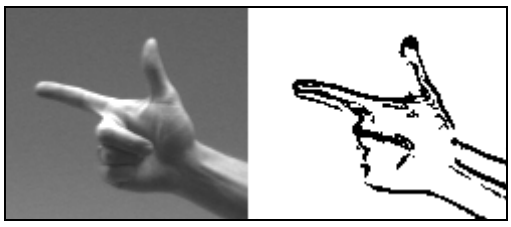


Figure 5 – Pointing gesture and binarized edge image

3.2 Object recognition

The pre-processed images are now used to detect and recognize eventually existing pointing gestures. The outcomes of this process are rough approximations of the contours of the gesture in the images. To detect these initial contours for a later fitting phase, the *Simulated Annealing* algorithm [Metropolis53, Pirlot96] was

chosen. *Simulated Annealing* is a stochastic optimization algorithm. Its purpose is the minimization of an objective function *E(X)* where *X* is a multidimensional state vector of the objective function. An initial value x_0 is randomly changed over many iterations by updating the current solution by a solution randomly chosen in its neighborhood. The change of the variable from x_{i-1} to x_i may result in an increase or decrease of the function value. To avoid getting stuck in a local minimum, it is necessary not only to allow ameliorations but also suitable deteriorations. This is done by introducing a temperature parameter *T*, which is decreased every *n* iterations. $E(x_i)$ is accepted as the next current solution if $E(x_i) < E(x_{i-1})$. Otherwise, there are two possibilities for the state of *X*: Either x_i will be accepted with the probability $P(x_i)$ or rejected with $1- P(x_i)$. *P* is calculated according to the Boltzmann distribution

$$P(\Delta E) = \exp\left[\frac{-\Delta E}{k_B T}\right]$$

with $\Delta E = E(x_i) - E(x_{i-1})$ and k_B the Boltzmann's constant. At the end of the algorithm, when *T* is small enough, deteriorations will hardly be accepted and, most of the time, only downhill steps are accepted.

To use *Simulated Annealing* for object recognition, it is necessary to adapt the algorithm by specifying the objective function *E(X)* and the change of the variable *X* from one state to another. Since the gesture is described by its boundary, the value of the objective function can be calculated as the sum of edge information values over all landmark points of the current shape. After placing an initial contour into the image, it is transformed from step to step by choosing random values for translations in x- and y- direction, a scaling and a rotation of the shape. Furthermore, suitable deformations of the current shape are allowed using the *Point Distribution Model* [Cootes00] that was calculated in an offline training phase.

For each step, the cost function is calculated as the sum of edge information values *g* over all landmark points p_i of the current shape:

$$E(X) = -\sum_{i=1}^{n} g(p_i)$$

Assuming that high values in the edge map indicate strong edge information, it is intuitively clear that a contour with no information at any landmarks generates a high function value and that a perfect matching contour produces the smallest possible function value of *E*.

3.3 Object fitting

The outcome of *Simulated Annealing* is a rough approximation of the true gesture. This approximation now has to be fitted to the real contour. Using an *Active Shape Model* [Cootes00], this fitting is done as the next step in the recognition process. Here, we iterate toward the best fit by examining an approximate fit, locating improved positions for all landmark points of the gesture, then recalculating a valid contour by using the underlying *Point Distribution Model*. The outcome of this process is the true boundary of the hand as seen in figure 6. We are now able to derive typical features of the pointing gesture like the position of the forefinger tip or the center of gravity of the hand to calculate a position and orientation of the gesture in three dimensions.

Figure 6 – Initial contour and fitted contour

3.4 3D Pose estimation

With a calibrated stereoscopic camera system, it is possible to reconstruct 3D coordinates of corresponding image points (e.g. finger tips or the center of the hand) in order to estimate a position and direction of the pointing gesture (see figure 7). For example, the center of the hand P_1 in the left camera image and the corresponding point P_2 in the right camera image are known and the optical axes are not parallel. Ideally, the rays through the camera center point $C_{M,1}$ and P_1 and through the camera center point $C_{M,2}$ and P_2 should intersect. Due to calibration errors and discretization, this does not normally occur. A good approximation is the midpoint of the shortest connection line between the two rays. Let s_1 be the direction of the ray through $C_{M,1}$ and P_1 and s_2 the direction of the ray through $C_{M,2}$ and P_2. Ray 1 is then given by:

$$C_{M,1} + t \cdot s_1 , \; t > 0$$

where $s_1 = P_1 - C_{M,1}$. Calculating the normal vector n_1 of the plane containing ray 1, the intersection of this plane with ray 2 is given by

$$n_1 \cdot (C_{M,2} + t \cdot s_2 - C_{M,1}) = 0.$$

A second point can be evaluated by intersecting the plane containing ray 2 with ray 1. The midpoint P_M of both intersection points then obtains the estimation of the reconstructed 3D coordinates of the center of the hand (see figure 7).

By reconstructing several landmark points lying on the upper boundary of the pointing forefinger, the direction of pointing is reconstructed applying a linear regression on these 3D points. The center of gravity of the gesture is used as the reconstruction of the 3D position. Position and orientation of the hand are used to start the tracking of the pointing hand.

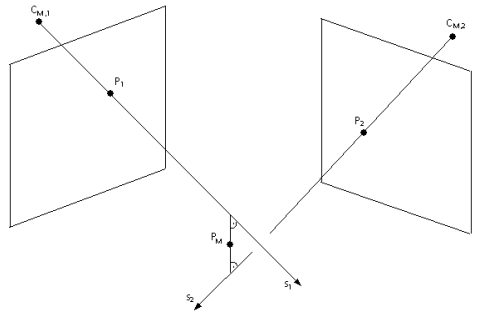


Figure 7 – 3D pose estimation

After being recognized and tracked, the pointing gesture must be integrated with the other modalities. This is what we describe in the next section, starting with some general consideration of the fusion of modalities.

4. MODALITY INTEGRATION

Following a design space in respect to the interaction process, parameters about temporal availability and the fusion possibility of the different modalities must be inferred. These values can have meaning or not, depending on the level of abstraction in which the data is being processed (the representation of speech input as a signal, as a sequence of phonemes or as a meaningful parsed sentence are examples of different abstraction levels). The next figure, taken from [Nigay93], classifies the different situations that should be considered. The shadowed zones are where a multimodal system would figure in.

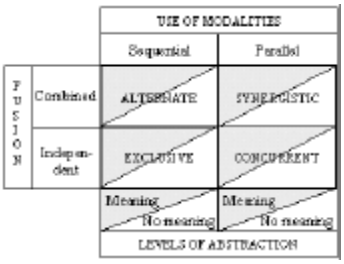


Figure 8 – Multi-feature system design space

There are two distinct classes of multimodal systems - one integrates signals at the *feature level* and the other at the semantic level. The first one is based in multiple hidden Markov models or temporal neural networks and is adequate for closely coupled and synchronized modalities (e.g. speech and lip movements). The other one is based on an *amodal input* and is appropriated when the modes differ substantially in the time scale characteristics of their features (e.g. speech and gesture input) [Wu99].

In the first phase of EMBASSI, we use semantic fusion with amodal input, considering the visual modalities in a so-called *late fusion* state. (The case of lip-reading, for instance, is considered visual, but is related to the perception of speech and requires an *early fusion* process).

Since the speech modality is out of the scope of this paper, we are giving emphasis to how the gesture modality will influence the global interaction. We will also include the gaze mode of interaction, considering that it has a similar, even if more restricted, purpose – the selection of devices.

4.1 Device selection

Similar to the speech modality, just after being recognized, the visual modalities must be analyzed in order to send valid information to the integration component. Due to the restrictiveness of this analysis component, it was called the device selection module.

In our approach, the modality integration is done in a two-step process. The first step consists of a common analyzer for two recognized modes, gesture and gaze, with the advantage of a *mutual disambiguation* possibility. This procedure constitutes a kind of pre-fusion mechanism occurring just before the global fusion with the speech modality. After being processed, the results are references to devices, in a whole scenario of several devices ("living room scenario"). Together with the others, these modalities play an important role in the overall multimodal interaction process, yielding partial solutions to difficult problems of natural language anaphora.

The goal is to obtain a selected device, or a set of them in case of ambiguity, based on the vector received from the recognizer components. This is done through cooperation with other modules on several levels, one of them being the establishment of a *world coordinates system*.

The components involved are the gesture and gaze *recognizers*, an *embassi sensor* responsible for setting-up the system, and a *context manager*. The following picture depicts the flux of information. We are considering the use of a context manager to store the spatial characteristics of the devices.

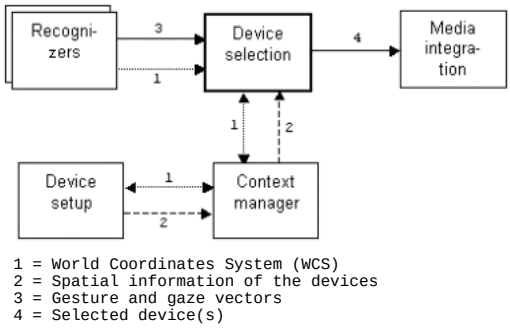


Figure 9 – Cooperation for device selection

After being "agentified" by the EMBASSI KQML-based infrastructure, the components communicate with each other by using appropriated performatives. The following illustration represents an example of communication between the recognition and analyzer modules: - with a KQML message and after calculating the position and direction, the gesture (or gaze) recognizer informs the gesture analyzer with spatial and temporal information.

```
(tell :sender GestureRec
      :receiver DeviceSelection
      :reply_with Ge-Rec_Msg1.0
      :ontology spatialOntology
      :language XML
      :content (<event time="23:59:59:321">
                  <vector x="1" y="2" z="2"
                          dx="0.4" dy="0.75"
                          dz="0.3"
                    actor="hand" />
            </event> ) )
```

Figure 10 – Agents communication example

The core operation of the device selection component is based on the ray interception step, detecting the intersections between the line that represents the hand pointing gesture and the devices' surfaces. The same procedure could be applied to the gaze direction.

Due to precision restrictions, we are considering several lines within a probability space. That consists of a *space of ambiguity*, which can be visually represented by a cone in space, formed from the user position (P_1) to the *device zone*.

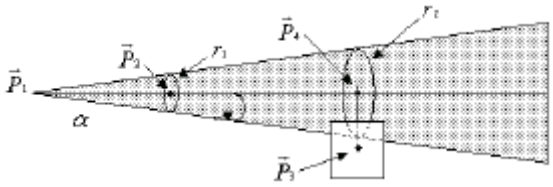


Figure 11 – Space of ambiguity

Here, an angle a is introduced to form a cone, within which the intersections are calculated. The further the object is from the user's position, the greater the imprecision that can occur. In the picture, we can see two positions, (P_2 or P_4) with different coefficients of ambiguities represented by the rays r_1 and r_2, respectively. The objects in P_3 can be detected due to the great ambiguity in position P_4.

Therefore, it is often possible to have more than one selected device. The first way to solve this is through fusion with the gaze. The fusion mechanism is done by "time-proximity", which is feasible due the basic system characteristic of event time-stamping: supposing that P_t represents a pointing act in an instant t, and that it occurs during some ?t. The relevant points in time for this act are the beginning (t) and the end (t+?t) where $P_t = P_{t+?t}$. It is in this interval of time (?t) that the fusion is performed.

In case of ambiguity persistence, it will be solved with the help of speech. The input of the media integrator in Fig. 9 is a graph of device probabilities.

4.2 Actions

Currently, we are using only the deictic gesture. It can be seen like a pen in the 3D space, pointing and performing linear movements, particularly used to interact with elements in a display like a big TV set.

Due to tracking and segmentation difficulties in natural environments, and in order to build a robust system, we have restricted the vocabulary, namely in that which concerns the rapidity of movements. Optimistically speaking, this is a good characteristic due the fact that the vocabulary set must be small to be accepted and learnable by the user. The next table summarizes the possible actions we are implementing.

Motion	Meaning	Action examples
Fix	- select	- with "turn this on"
Fast change	- origin/ destination	- with "put that there"
slowly left slowly right	- horizontal scrolling	- video forward - TV menu items
slowly up slowly down	- vertical scrolling	- volume - TV menu items
forward	- activate	- turn on
backward	- deactivate	- turn off

Table 1 – Gesture vocabulary

Our vocabulary set contemplates the "on" and "off" actions, the left, right, up, down movements (to select a set of options vertically or horizontally distributed), and another more aleatory "from-to" pointing.

5. CONCLUSIONS

We have briefly presented an ambitious project addressing innovative methods of man-machine interaction in non-professional environments of everyday life, such as at home and in the car. Most of the prototypes already developed will undergo improvements in the second phase of the project that is now underway.

Our main focus in this paper was the home environment, concerning the input with gestures in order to complement, e.g. infrared or speech remote controls. We are demonstrating that by using only one feature, the deictic gesture, we can tremendously reduce the recognition task and still have a functional dialogue system.

This natural method of interaction, without the need for markers attached to the user's body, for example, remains a very difficult task due to precision problems. Two ways to minimize these problems are to use (visual or auditory) feedback and the benefits of multimodality, which enjoys a high level of preference among users, as many research studies (e.g. [Chu97]) report.

ACKNOWLEDGEMENTS

The work of the first author is partially funded by the Portuguese Ministry of Science and Technology (MCT), through the Information Society Operational Program (reference PRAXIS XXI/BD/20095/99).

The EMBASSI project is supported by the German Ministry of Science and Education (BMBF) as a focus project on Man-Machine Interaction (signature 01IL904).

REFERENCES

[Cootes00] Cootes T., Taylor C.: "Statistical Models of Appearance for Computer Vision", University of Manchester, http://www.wiau.man.ac.uk, 2000.

[Cost99] Cost R. et al.: *"An Agent-Based Infrastructure for Enterprise Integration"*, 1st International Symposium on Agent Systems and Applications / 3rd Inter-

national Symposium on Mobile Agents, IEEE Computing Society, 1999.

[Chu97] Chu C., Dani T., Gadh R.: *"Multimodal Interface for a Virtual Reality Based Computer Aided Design System"*, Proceedings of IEEE International Conference on Robotics and Automation, 2:1329-1334, April 1997.

[Domini97] Domini F. M. et al.: "Reasoning in Description Logics", CSLI Publications, 1997.

[Finin93] Finin T. et al.: "Specification of the KQML Agent-Communication Language", University of Maryland, 1993.

[Franconi99] Franconi E.: *"Ontology!"*, http://www.cs.man.ac.uk/~franconi/ontology.html, 1999.

[Haberäcker95] Haberäcker P.: "Praxis der Digitalen Bildverarbeitung und Mustererkennung", Carl Hansen Verlag, 1995.

[Hildebrand00] Hildebrand A., Sá V.: *"EMBASSI: Electronic Multimedia and Service Assistance"*, Intelligent Interactive Assistance and Mobile Multimedia Computing – IMC2000, Rostock-Warnemünde, Germany, November 9-10, 2000.

[Jacob96] Jacob, R. J.: "The Future of Input Devices", ACM Computing Surveys 28A(4), December 1996.

[Jeon00] Jeon H., Petrie C., Cutkosky M.: *"JATLite: A Java Agent Infrastructure with Message Routing"*, IEEE Internet Computing, March-April 2000.

[Kohler96] Kohler M.: "Vision Based Remote Control in Intelligent Home Environments", 3D Image Analysis and Synthesis '96, Erlangen, 18-19 November, 1996.

[Martin99] Martin D., Cheyer A., Moran D.: *"The Open Agent Architecture: A Framework for Building Distributed Software Systems"*, Applied Artificial Intelligence: An International Journal", 13(1-2): 91-128, January-March 1999.

[Metropolis53] Metropolis N. et al.: *"Equation of state calculation by fast computing machines"*, Journal of Chemical Physics, 21:187-1092, 1953.

[Milota95] Milota A., Blattner M.: *"Multimodal interfaces with voice and gesture input"*, IEEE International Conference on Systems, Man and Cybernetics, 3:2760-2765, 1995.

[Nigay93] Nigay L., Coutaz J.: *"A Design Space for Multimodal Systems: Concurrent Processing and Data Fusion"*, Proceedings of InterCHI'93, Conference on Human Factors in Computing Systems, ACM, 1993.

[Oviatt97] Oviatt S., DeAngeli A., Kuhn K.: *"Integration and Synchronization of Input Modes during Multimodal Human-Computer Interaction"*, Proceedings of Conference on Human Factors in Computing Systems: CHI '97, 415-422, ACM Press, New York, 1997.

[Pirlot96] Pirlot M.: "General local search methods", Elsevier Science B.V, 1996.

[Sonka98] Sonka M., Hlavac V., Boyle R.: "Image Processing, Analysis and Machine Vision", PWS Publishing, 1998.

[Wu99] Wu L., Oviatt S., Cohen P.: *"Multimodal Integration – A Statistical View"*, IEEE Transactions on Multimedia, 1(4):334-341, 1999.

[W3C00] W3C – World Wide Web Consortium: *"Extensible Markup Language"*, http://www.w3.org/XML, 1997-2000.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Jogo do Galo
Agente de Apoio à Utilização de Jogos por Deficientes Profundos

António Pereira Ricardo Amaro
Licenciatura em Engª. Informática, Instituto Superior Técnico
Av. Rovisco Pais, 1000-049 Lisboa
{ajabsp, rjmpa}@mega.ist.utl.pt

Ana Paiva João Brisson Lopes
Departamento de Engª. Informática, Instituto Superior Técnico
Av. Rovisco Pais, 1000-049 Lisboa
Ana.Paiva@inesc.pt brisson@ist.utl.pt

Sumário
O projecto apresentado nesta comunicação teve por objectivo realizar e testar um agente autónomo para apoio à utilização de jogos por deficientes profundos. O agente desenvolvido tem como primeiro objectivo a parcial substituição do terapeuta que apoia o utilizador. A sua função é ensinar e manter o utilizador interessado no jogo. Apresenta-se a arquitectura adoptada e as opções tomadas durante a concepção e desenvolvimento realizados, incluindo aspectos como a comunicação com o jogo e os comportamentos com que o agente tenta manter o utilizador motivado e interessado. Os testes de campo realizados permitiram verificar que o emprego de um agente é extremamente benéfico e constatar problemas de que se apresentam soluções para o trabalho que dará continuidade ao agora realizado.

Palavras-chave
Agentes autónomos, interacção, dispositivos de interacção, realimentação, deficiência, terapia, jogos.

1. INTRODUÇÃO

A utilização das tecnologias da informação por pessoas portadoras de deficiência coloca problemas muito específicos, principalmente ao nível da interacção pessoa-máquina. O grau de variabilidade de deficiência implica o emprego de soluções específicas, pois "cada caso é um caso" diferente de todos os outros [Eduards99]. Consequentemente, a interacção entre o utilizador e a máquina deve ser adaptada a cada utilizador específico, tanto no que se refere aos dispositivos de entrada e saída, como às características do diálogo, sua apresentação e incentivo ao diálogo.

No caso de portadores de deficiência profunda, existem casos em que a interacção com dispositivos de entrada pode ser realizada com base em dispositivos como ratos ou joysticks. Mas estes casos são raros. Para a generalidade dos portadores de deficiência, com origem em paralisia cerebral ou paraplegia em elevado grau, as limitações psicomotoras impedem o emprego de dispositivos daqueles tipos.

A maioria das soluções adoptadas para estes casos baseia-se em técnicas de varrimento dos ícones seleccionáveis presentes no ecrã com selecção por meio de dispositivos do tipo manípulo (ou interruptor de mão). Um dispositivo deste tipo, de que a figura 1 apresenta um exemplo, embora lento, permite níveis de interacção bastante elevados como, por exemplo, a composição de textos por combinação de um manípulo com um teclado virtual.

As aplicações têm assim que suportar soluções como a anterior para proporcionar o aumento da acessibilidade. No entanto, este aspecto cobre apenas a entrada de comandos e dados.

Figura 1 – Manípulo.

Entre os factores mais importantes na interacção pessoa-máquina encontram-se o reconhecimento do estado do sistema ou da aplicação, que se pretende imediato, e a

retro alimentação, que assinala os resultados dos comandos introduzidos pelo utilizador. Este deve poder facilmente aperceber-se do estado em que a aplicação se encontra, que comandos podem ser executados e da alteração do estado da aplicação após a execução de um comando.

Estes requisitos, essenciais para uma boa interface, são particularmente críticos no caso de interfaces destinadas a serem usadas por portadores de deficiência, podendo-se estabelecer um paralelo entre aquelas características das interfaces e a solicitação dos portadores de deficiência à comunicação. Com efeito, na sua vida comum, estas pessoas requerem atenção continuada por parte e terapeutas, educadores e familiares cuja missão consiste em, entre outras coisas, solicitar a comunicação e premiar a sua realização. O reforço ao sucesso e o incentivo face ao insucesso desempenham um papel muito importante neste contexto.

Quando a máquina assume, ainda que parcialmente, o papel de terapeuta ou educador, cabe à interface sublinhar o estado da sua aplicação, solicitar a interacção e reforçar o prémio face aos resultados da interacção, sempre numa óptica de incentivo.

Existem técnicas variadas para realizar estes objectivos. De entre elas, destaca-se o emprego de agentes autónomos. Este trabalho teve por objectivo investigar o emprego de agentes autónomos como incentivador à comunicação por parte de pessoas portadoras de deficiência profunda.

1.1 Projecto Intercomunicando

O presente trabalho integra-se num projecto de investigação e desenvolvimento em curso, o projecto Intercomunicando, financiado pelo programa CITE 2000 do Secretariado Nacional para a Reabilitação e Integração das Pessoas com Deficiência.

O projecto tem por objectivo estabelecer um ambiente de comunicação mediada pela máquina de pessoas portadoras de deficiência entre si e com terapeutas, educadores e familiares.

Para além das ferramentas e aplicações destinadas à comunicação, empregando as linguagens pictográficas PIC (Pictogram Ideogram Communication [Maharaj]), PCS (Pictographic Communication System [Johnson97]) e Makaton ([Makaton]), o projecto Intercomunicando desenvolverá também outras ferramentas destinadas ao treino e ambientação dos utilizadores e à gestão dos perfis de utilizador e suas fichas clínicas[1].

1.2 Jogo do Galo

Na sua fase inicial, o projecto Intercomunicando desenvolveu alguns protótipos com o fim de avaliar a qualidade e adequação de possíveis soluções.

Uma das soluções estudadas consistiu no emprego de agentes autónomos como forma de realizar o apoio e incentivo à comunicação por parte de portadores de deficiência.

Os testes com agentes autónomos no papel de entidades incentivadoras da interacção entre o portador de deficiência e a máquina deveriam ser realizados em condições atractivas e de pequena complexidade. De entre as várias hipóteses consideradas, os jogos preenchiam todos estes requisitos, principalmente se fossem simples.

Estas razões levaram então à realização dos testes dos agentes autónomos com o jogo do galo.

O trabalho aqui apresentado foi realizado como um projecto da disciplina de Introdução aos Agentes Autónomos da Licenciatura em Engenharia Informática e de Computadores do Instituto Superior Técnico.

1.3 Requisitos dos Utilizadores

Os utilizadores alvo deste projecto são indivíduos com elevada deficiência, tanto a nível motor como a nível mental, e necessitam, a todo o tempo, de apoio por parte de um terapeuta. Esse apoio tem duas vertentes. A primeira vertente consiste na motivação e no incentivo à participação do deficiente na tarefa a realizar. A outra vertente consiste no ensino da utilização da interface e no ensino da realização da própria tarefa.

Estes utilizadores apresentam geralmente um controlo motor espástico, em grau muito variado, mas com a característica fundamental de não poderem realizar operações de selecção fina. Isto significa que os ícones presentes numa interface deverão ser de grande tamanho de forma a permitir operações de selecção sem grande precisão. Este requisito aplica-se tanto ao emprego de dispositivos apontadores comandados pela mão como pelo pé, existindo casos em que o emprego de dispositivos baseados em pedais constitui a única solução possível.

Os utilizadores apresentam também características muito variáveis quanto aos aspectos de rapidez de compreensão e accionamento dos dispositivos apontadores e seleccionadores. Isto exige temporizações muito específicas adaptadas a cada utilizador.

1.4 Metodologia Adoptada

Dada a continuidade que vai ser dada a este projecto através do projecto Intercomunicando, optou-se, numa primeira fase, pela familiarização com as ferramentas de desenvolvimento e as técnicas de criação de agentes [Português00].

Desenvolveu-se assim, um agente básico de apoio ao jogo do galo. Este agente ensina a utilizar a interface e mantém o utilizador atento, não tendo capacidade de ensino das estratégias de jogo [Microsoft98a, Microsoft98b, Microsoft98c].

O agente adoptado foi o Microsoft Agent que, embora apresente algumas limitações, permite:

- Várias personagens
- Expressões corporais variadas e ampliáveis
- Geração de fala a partir de texto

[1] Um dos objectivos consiste em avaliar do progresso da comunicação realizada ao longo do tempo.

A natureza dos utilizadores e as limitações do Microsoft Agent, levantaram alguns problemas. Para a motivação dos utilizadores, é importante que a personagem associado ao agente apresente uma boa expressividade na fala e nas animações. Isso não é de todo conseguido porque o motor de fala do MS Agent não está muito desenvolvido para Português e as animações disponibilizadas não são as mais desejadas. Por outro lado a quantidade de animações e falas têm que ser limitadas para que os utilizadores consigam facilmente perceber os incentivos e obtenham um bom desempenho

2. ARQUITECTURA DO AGENTE

Um agente para realizar um jogo é um agente específico, isto é, o seu estado e funcionamento dependem do jogo utilizado. No entanto, para qualquer jogo o agente terá de manter um estado interno que o permita avaliar as condições do "mundo" e decidir as acções que deve realizar.

No caso concreto deste trabalho, este estado é simples. O agente deve manter o conjunto de jogadas que cada utilizador realiza e informação sobre quais as combinações de jogadas ganhadoras. Isto é suficiente para que o agente saiba qual o estado do estado do tabuleiro de jogo e perceba quando cada jogo termina e qual o respectivo resultado. A implementação do agente deve também manter valores estatísticos sobre cada sessão com o objectivo de definir o seu estado emocional.

A figura 2 apresenta a arquitectura geral do agente autónomo concebido.

As frases que o agente deve proferir podem ser divididas em 4 tipos de frases que dependem do estado do jogo e dos resultados obtidos:

- Vitória
- Derrota
- Empate
- Continuação do jogo

As frases correspondendo a cada um destes tipos estão organizadas em tabelas. Cada tabela está dividida em 3 partes, sendo a cada uma dessas partes associada um valor do estado emocional. É assim possível seleccionar uma frase apropriada à emoção do agente.

Por cada frase seleccionada é realizada uma rotação da tabela de modo a que selecções consecutivas de frases da mesma tabela não resultem na repetição de frases. Este mecanismo garante a manutenção implícita de um historial de frases.

A detecção de um "longo" período de inactividade por parte do jogador é realizada através de tempos limite (timeouts) que são reinicializados quando é realizada uma jogada.

O estado emocional do agente é actualizado no final de cada jogo, de acordo com o resultado do mesmo e com o estado emocional anterior. Esta actualização é realizada pelo "módulo reactivo".

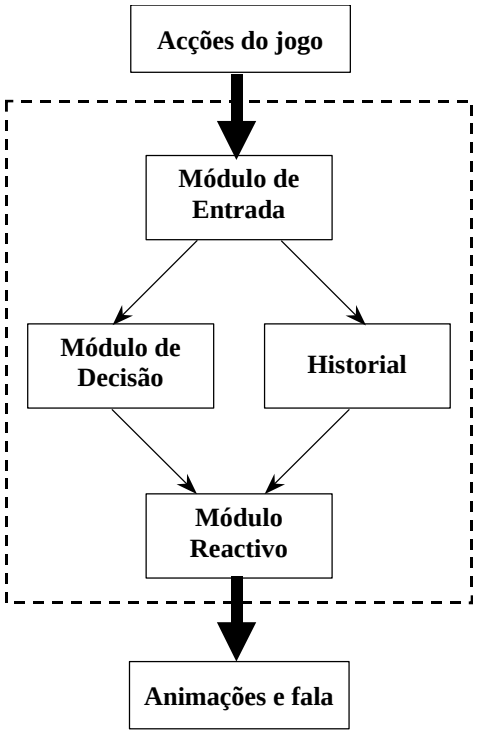


Figura 2 – Arquitectura de um agente autónomo apresentando os respectivos módulos

O estado do tabuleiro de jogo e das estatísticas é actualizado pelo "módulo de entrada" sempre que recebe uma jogada.

Cabe ao "módulo de decisão" analisar o seu estado e escolher a acção a realizar, comandando directamente a personagem do MS Agent. O historial de frases é mantido e actualizado por este módulo.

3. DESENVOLVIMENTO

O desenvolvimento deste projecto foi realizado em várias fases que compreenderam

- O jogo
- A comunicação Agente - Jogo
- A comunicação Agente - Utilizador.

3.1 Jogo

Numa primeira fase foi necessário desenvolver um jogo para poder testar e demonstrar as capacidades do agente. O jogo do galo foi o escolhido, por ser simples e de fácil aprendizagem.

O jogo tem implícito um agente jogador. Dadas as limitações que os utilizadores a que o jogo se destinava apresentam, este agente foi desenhado para apresentar uma proficiência mínima. Deste modo, inseriu-se um factor aleatório nas suas jogadas, permitindo assim a tomada de decisões erradas.

3.2 Comunicação Agente - Jogo

Existiam à partida várias hipóteses para desenvolver o jogo. Estas hipóteses tinham implicações na comunicação entre o agente e o jogo.

Uma destas formas consistia na implementação do agente e do jogo através de Applets independentes que comunicariam entre si por meio de sockets. Esta solução permitiria a portabilidade do módulo do agente para outro jogo. A informação a transmitir por meio de sockets seria informação sobre as jogadas efectuadas. Todas as outras percepções, nomeadamente no que toca aos movimentos do rato e mesmo o tempo que este esta inactivo, teriam de ser captadas utilizando ferramentas disponibilizadas pela Microsoft.

Uma outra hipótese consistia na integração do agente e do jogo num mesmo módulo, sendo assim a comunicação e as restantes percepções realizadas mais facilmente. Por este motivo, foi esta a solução adoptada.

Como o jogo do galo é bastante simples, as únicas percepções que o agente tem que absorver para manter o estado do mundo actualizado são as jogadas. Como o agente se encontra integrado na própria aplicação, as percepções são realizadas por chamadas a funções disponibilizadas pelo agente.

3.3 Comunicação Agente - Utilizador

A comunicação entre o agente e o utilizador é feita através de uma personagem do MS Agent e de uma barra que representa o estado emocional do agente. Este estado resulta da evolução de resultados dos vários jogos.

Com a introdução da barra do estado emocional (satisfação) do agente na interface pretende-se encontrar uma motivação extra para que o utilizador vença uma série de jogos. A figura 3 apresenta o aspecto desta barra de satisfação (ou da felicidade).

Figura 3- Barra do estado emocional

Criou-se assim, uma série de jogos que termina quando é atingido um dos extremos da barra. Por cada jogo ganho pelo utilizador, o agente fica mais "feliz" e aproximamo-nos do extremo positivo da barra. Quando o utilizador perde um jogo, o estado do agente fica menos “feliz” e o comprimento da barra diminui. O estado emocional do agente mantém-se em caso de empate.

Durante todo o tempo de jogo, o agente vai apoiando e motivando o utilizador, através da fala e de animações. Estas estão directamente relacionadas com as incidências do jogo. No final de cada jogo, conforme tenha ou não ganho, o agente manifesta-se, felicitando ou animando o utilizador.

Caso este se mantenha inactivo durante um período de tempo, o agente tenta captar a sua atenção, incentivando-o a fazer uma jogada.

3.4 Ambiente da Aplicação

A solução encontrada para a comunicação permitiu ainda desenvolver o projecto através de ferramentas simples. O emprego do MS Agent pode ser feito a partir de aplicações em C++ e outras linguagens de alto nível, mas pode também ser feito a partir de scripts que correm debaixo de navegadores da WWW.

Assim, dado que se pretendia uma aplicação experimental em que fosse possível e simples introduzir alterações e aumento de funcionalidade, foi decidido que a aplicação a desenvolver deveria ser feita em JavaScript e HTML e que o ambiente de apresentação seria o proporcionado pelo navegador Internet Explorer.

Esta solução permitiria ainda a fácil alteração da aplicação utilizando ferramentas simples que estariam disponíveis durante a realização de testes de campo.

4. INTERFACE

Como vimos atrás, a interface da aplicação desenvolvida é apresentada ao utilizador através do navegador Internet Explorer. A primeira página apresentada é a página de apresentação do jogo (Figura 4) onde, com o auxílio de um terapeuta, educador ou outra pessoa, o utilizador pode escolher a personagem do Microsoft Agent que irá dar corpo ao agente e acompanhar o utilizador durante todo o jogo.

Depois da escolha da personagem para o agente, o assistente do jogo, é apresentada uma interface simples mas fácil de entender e utilizar. Nesta página, que a figura 5 apresenta, as dimensões dos objectos foram propositadamente exageradas devido às características especiais dos utilizadores.

Com efeito, a atenção dos utilizadores tem que ser atraída por objectos de grandes dimensões, não só para garantir a atracção, mas também porque a maioria dos utilizadores apresenta graves problemas de acuidade visual. Este problema, que era conhecido à partida, foi objecto de alguns testes sobre a atenção despertada por ponteiros. Verificou-se que os utilizadores preferiam ponteiros de grandes dimensões e que as eventuais animações que pudessem existir nos ponteiros não prejudicavam o seu reconhecimento, antes o facilitavam.

Tal como a figura 5 mostra, todas as barras de ferramentas foram retiradas da janela de jogo com o objectivo de:

- maximizar a área de jogo
- não confundir/atrapalhar a atenção do utilizador
- evitar que o utilizador pudesse inadvertidamente destruir ou danificar a janela de jogo

Existem no agente algumas funcionalidades extra, que só são acessíveis através de um clique com o botão direito do rato, em cima da personagem. Essas funcionalidades são apenas de interesse dos terapeutas, como por exemplos as estatísticas de jogo, as propriedades do personagem ou mesmo a opção de ocultar a personagem do agente.

Figura 4 - Página de entrada e escolha do assistente

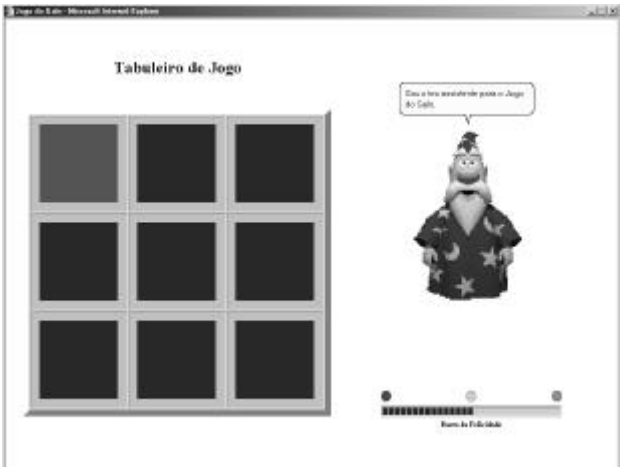


Figura 5 – Início da aplicação. A personagem do agente explica quem é e apresenta o jogo.

As estatísticas recolhem o número de jogos realizados, ganhos perdidos ou empatados, nas várias séries e no total, tal como a figura 6 apresenta.

O jogador utiliza o rato para clicar sobre o tabuleiro, escolhendo assim a posição onde quer jogar. Todas as acções do utilizador são realizadas pela selecção de quadrados do tabuleiro de jogo. Tanto para jogar, como para iniciar novo jogo, é necessário movimentar o rato no tabuleiro e clicar uma vez por cada acção. O agente comenta cada jogada com uma frase que é simultaneamente de incentivo a continuar.

Se o jogador pára de jogar, o agente espera pacientemente, como se vê na figura 7 . Se o utilizador demorar mais do que um dado tempo, que é configurável, o agente toma então uma atitude activa, incentivando o jogador a prosseguir (figura 8).

Ao fim de um jogo ganho pelo utilizador, o agente felicita-o efusivamente (figura 9) e, se o sucesso neste jogo leva a barra do seu estado (de felicidade) ao máximo, o agente, além de felicitar o utilizador, presenteia-o com um troféu (figura 10).

Figura 6 – Janela apresentando as estatísticas dos jogos.

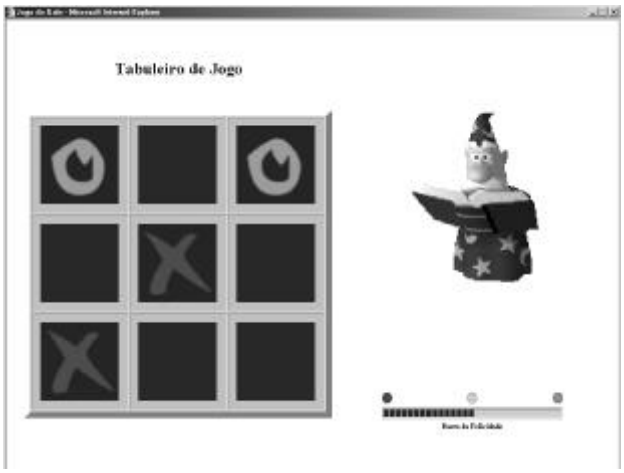


Figura 7 –Agente esperando a próxima jogada do utilizador.

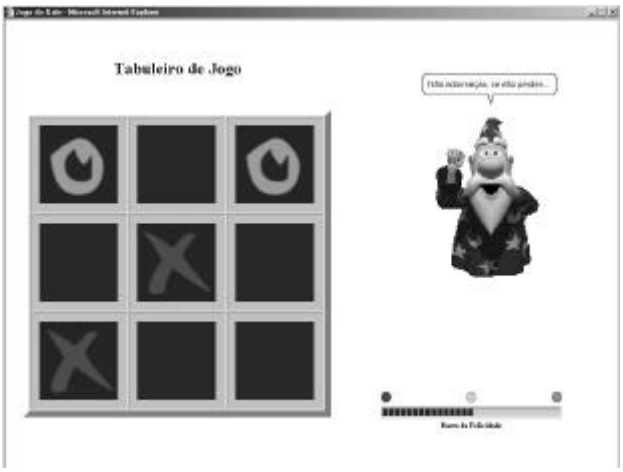


Figura 8 – Agente incentivando o jogador quando este deixou de jogar por um largo período de tempo.

Naturalmente, quando um jogador perde ou empata, o agente não deixa de assinalar o facto, diminuindo o seu estado de felicidade quando se tratar de uma derrota. Mas, em qualquer dos casos, o agente não deixa de incentivar o jogador.

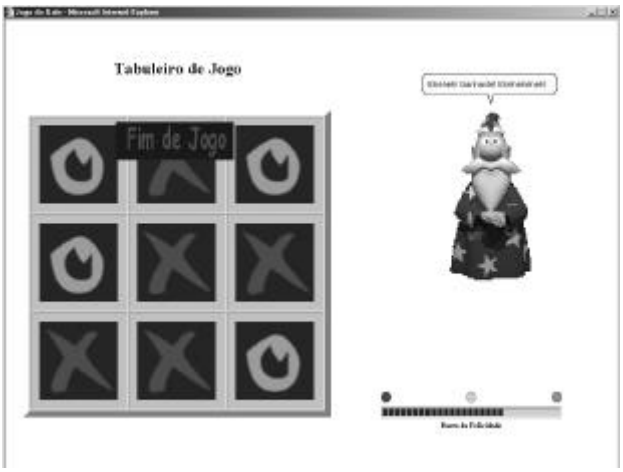


Figura 9 – Agente felicitando o jogador depois de uma vitória.

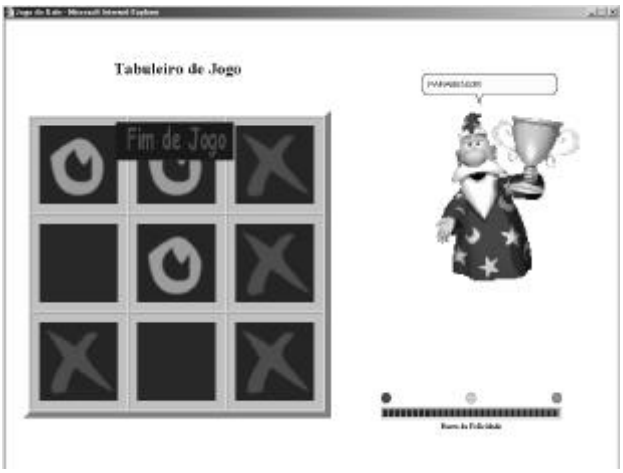


Figura 10 – Agente felicitando o jogador após uma série de vitórias.

5. TESTES DE CAMPO

Os testes de campo da aplicação desenvolvida foram realizados na AFID (Associação Nacional de Famílias para a Integração da Pessoa Deficiente), principal promotor do projecto Intercomunicando.

Os testes foram realizados com os objectivos de:

- Determinar os parâmetros mais significativos do modelo do utilizador, bem como os respectivos intervalos de variação.
- Avaliar do grau de aceitação e eficácia do agente na realimentação e no estímulo à prossecução de tarefas.
- Detectar necessidades e problemas da aplicação desenvolvida, com vista à continuação do projecto Intercomunicando.

Nos testes de campo participaram seis utentes da AFID que apresentavam deficiências motoras e mentais em graus variados. O grupo de teste era bastante heterogéneo, tendo elementos com dificuldades motoras notórias e outros com maior facilidade de movimentação e coordenação. O grupo incluiu pessoas com paralisia cerebral e pessoas com paraplegia em elevado grau.

Embora o tempo de treino dos utilizadores tenha sido nulo, o primeiro contacto com a aplicação foi interessante. Todos revelaram grande ansiedade e vontade de jogar e a atenção prestada ao agente foi aceitável.

6. CONCLUSÕES

Os testes de campo permitiram retirar bastantes conclusões que serão agora integradas na continuação do projecto.

Uma primeira conclusão diz respeito ao emprego do MS Agent. Com efeito os testes de campo permitiram constatar que o MS Agent não está suficientemente desenvolvido para poder exprimir com um certo grau de realidade as emoções e a expressividade que os humanos têm. As causas mais consensuais para este resultado têm origem na qualidade das animações disponíveis e no facto de o motor de fala a partir de texto empregue (Lernout & Hauspie) só se encontrar disponível para o dialecto brasileiro da língua portuguesa[2]. Provavelmente, poder-se-ão obter resultados muito melhores com um motor de fala em português que seja mais desenvolvido do que o motor empregue.

A realização de testes de campo, tornou clara a necessidade de estudar e desenvolver aplicações deste tipo, assim como alguns dos pontos menos positivos da aplicação desenvolvida revelados pelos testes. Embora o universo de utilizadores alvo tivesse sido bastante heterogéneo, podemos, de acordo com os resultados observados nos testes, dividi-lo em dois grupos: os que conseguiram compreender e jogar sem ajuda dos terapeutas e os que necessitaram ajuda.

Os primeiros, revelaram um maior grau de desenvolvimento e menores limitações, o que lhes permitiu compreender as regras e funcionamento do jogo. Mostraram grande motivação e concentração, não só no jogo em si mas, também, nas intervenções do agente. O objectivo fundamental deste projecto, motivar e cativar os jogadores, foi totalmente alcançado neste grupo.

Para tal, muito contribuiu a "barra da felicidade". Este grupo de utilizadores percebeu que a felicidade do agente dependia do seu desempenho no jogo. Os utilizadores deste grupo jogaram repetidas vezes até conseguirem alcançar o nível máximo da barra. Dada a evidente facilidade de adaptação à aplicação, ficou claro que, para este grupo, o desenvolvimento do agente e o do próprio jogo poderia ter sido bastante mais ambicioso.

Dos resultados experimentais obtidos junto deste primeiro grupo ressalta que as capacidades e inteligência do agente podem, e devem, ser melhoradas, aumentando e diversificando a sua intervenção no decorrer da utiliza-

[2] Note-se que este motor foi empregue tal como é distribuído, não tendo sido programadas quaisquer alterações ao timbre ou velocidade da fala gerada.

ção. Por sua vez, o jogo, poderá ser substituído por aplicações com um carácter terapêutico bem mais vincado.

Nos utilizadores do segundo grupo eram evidentes as limitações motoras e mentais. Embora a sua motivação e vontade de jogar fossem claras, a falta de concentração, a não compreensão da aplicação e, principalmente, a enorme dificuldade de adaptação aos periféricos de interacção do computador não permitiram alcançar os objectivos. Foi, no entanto, possível detectar alguns pontos a rever, dada a sua inadequação ao perfil dos utilizadores.

As diversas reacções observadas nos dois grupos de utilizadores tornaram também evidente a necessidade de introduzir funcionalidade para adaptar os níveis de jogo e de actuação do agente ao utilizador. O processo de selecção de jogadas deve também ser configurável.

Neste capítulo, a conclusão principal é a necessidade de simplificar as tarefas a realizar. Sem excluir o emprego de ratos ou joysticks, é necessário aumentar o número de formas de realizar a selecção através do varrimento visível das posições do tabuleiro, permitindo assim que a interacção com a aplicação seja feita por um simples toque num manípulo.

Igualmente, verificou-se que as dimensões originais da barra exprimindo o estado do agente ("barra da felicidade") eram demasiado reduzidas. Esta deficiência foi corrigida numa versão posterior do protótipo, estando em curso o estudo e avaliação de concepções alternativas para exprimir o estado emocional do agente.

Outra conclusão é a necessidade de dar maior ênfase à jogada realizada pelo computador, pois muitos dos utilizadores não se apercebiam de que este tinha acabado de realizar uma jogada. Esta ênfase pode mesmo ser reforçada por comentários do agente.

Os testes realizados evidenciaram ainda problemas com origem em dificuldades motoras. Os acontecimentos e as acções de selecção devem ser convenientemente espaçadas no tempo e as acções de selecção sujeitas a uma filtragem apropriada. Com isto evitar-se-ão problemas registados nos testes, como a sobreposição de acontecimentos provocada pelos toques duplos ou prolongados no rato.

Fundamentalmente, é muito importante desenvolver uma aplicação que permita um elevado grau de configuração, de modo a conseguir a adaptação ao maior número possível de utilizadores, e dotar o agente de apoio à aplicação de uma maior capacidade.

Uma possível solução consistirá no emprego do framework APE para agentes autónomos [Cabral01]. Esta solução permitiria que aplicações do mesmo tipo apresentassem os seus objectos de uma forma normalizada e seria possível empregar um determinado agente com um grupo de aplicações e não apenas com uma.

Dos resultados obtidos, parecem óbvias as inúmeras possibilidades e potencialidades da utilização de agentes neste campo específico das aplicações informáticas para terapia de deficientes.

7. TRABALHO FUTURO

A experiência adquirida com este projecto permite-nos afirmar que o emprego de agentes com o fim de incentivar e premiar utilizadores portadores de deficiências profundas apresenta boas perspectivas.

Assim, para além dos melhoramentos que atrás referimos, os próximos passos consistirão no desenvolvimento de um framework genérico no contexto da comunicação mediada pela máquina e na aprendizagem das linguagens pictográficas empregues nesta comunicação. Este framework deverá apresentar funcionalidade que permita a construção de perfis do utilizador consoante o seu desempenho e características da sua deficiência.

Este trabalho será realizado no âmbito de um Trabalho Final de Curso intitulado "Comunicação e Aprendizagem para Deficientes Profundos", da Licenciatura em Engenharia Informática e de Computadores do Instituto Superior Técnico. Nesse trabalho irá ser melhorada a interacção do agente com o utilizador, as suas percepções e o seu nível de inteligência e adaptação ao utilizador.

8. AGRADECIMENTOS

Os autores agradecem todo o apoio prestado pela AFID durante a realização deste trabalho, em especial o apoio prestado pela Dra. Fátima Cabral e pela Dra. Edite Antunes.

9. REFERÊNCIAS

[Bradshaw97] Bradshaw, J., Software Agents, MIT Press, 1997.

[Cabral01] Cabral, N., Astronomia Juvenil 2 – Introdução de Emoções nos Agentes Pedagógicos Tiara e Titan, Trabalho Final de Curso, Licenciatura em Eng. Informática e de Computadores, IST, 2001.

[Eduards99] Eduards, A. D. N., Extra-Ordinary Human-Computer Interaction, Cambridge University Press, Cambridge, 1999.

[Johnson97] Johnson, R.M., The Picture Communication Symbol Guide, Mayer-Johnson, 1997.

[Maharaj] Maharaj, S.C., Pictogram Symbol Reference Book (Internationally Used Symbols), ZYGO Industries Inc.

[Makaton] The Makaton Vocabulary Development Project (MVDP), http://www.makaton.org.

[Microsoft98a] The Microsoft Agent User Interface, Microsoft Corporation, 1998.

[Microsoft98b] Programming the Microsoft Agent Control, Microsoft Corporation, 1998.

[Microsoft98c] Programming the Microsoft Agent Server Interface, Microsoft Corporation, 1998.

[Português00] Português, R. (Ed.), Seminário de Agentes de Interface, Instituto Superior Técnico, 2000.

Sessão 3

MODELAÇÃO E VISUALIZAÇÃO

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

IDL: A Geometric Interference Detection Language

Pedro Santos
Pedro.Santos@iscte.pt

Manuel Gamito
mag@iscte.pt

José Miguel Salles Dias
Miguel.Dias@iscte.pt

ADETTI/ISCTE, Associação para o Desenvolvimento das Telecomunicações e Técnicas de Informática, Edifício ISCTE, 1600-082 Lisboa, Portugal, www.adetti.iscte.pt

Abstract

This paper describes a novel programming language approach to the problem of automatically verifying the design in an architectural and civil engineering project, fully described in 3D. The language is referred to as IDL, Interference Detection Language, and is used to write geometric-based design verification tests, which are then interpreted and applied to an architectural 3D virtual scene, resulting in geometric interferences if the geometric objects of the scene fail to verify those tests. IDL operators are algorithmically based on simple Geometric Boolean Set and Constructive Solid Geometry operations, which are applied on a Binary Spatial Partitioning organisation of the 3D scene being analysed. This paper also presents Visual IDL, an intuitive graphical editor, which allows a common user to write design verification tests, according to the IDL syntax notation, and still be relatively independent of the language constructs. With Visual IDL, the user is not required to learn the language itself, but is still able to write typical tests quickly and efficiently. Visual IDL can be thought of as a higher-level abstraction of the language, allowing the user to concentrate on exactly what is required, rather than on how to execute it.

Keywords

Computing in AEC-Architecture Engineering and Construction, automatic design verification, geometric interference detection, visual scripting language.

1. INTRODUCTION[1]

Automatic Design Verification (ADV) technology is becoming an important topic in 3D Virtual Prototyping and Digital Mockup (DMU) applications. Some players are operating in this area, with their solutions biased towards the automotive and aerospace markets. This type of technology uses a spatial organisation of the 3D virtual scene based on voxels [Tecoplan]. After a voxel conversion pre-processing stage, end-users can quickly verify any test environments for collision conditions and minimum distance checks between parts. Collisions can also be checked on-line for assembly and disassembly of parts. Static and dynamic swept volumes can be generated for packaging purposes. The technology can be either integrated into existing high-end CAD packages, such as CATIA, or offered as a standalone product, using the VRML data format and a multi-user virtual environment [Blaxxun]. This approach has inspired the authors in developing a new Automatic Design Verification technique for the AEC-Architecture Engineering and Construction sectors, (but that can well be applied in other industrial sectors), with the following characteristics:

- Flexible and upgradeable, through the development of the core of the technology: a simple yet extensible design verification language, referred to as IDL, the Interference Detection Language, the focus of this paper.
- Data format independence by the adoption of the VRML as its native format.
- Support of Object Classification following the recommendations of an ISO standard [ISO 13567].

IDL operators are algorithmically based on basic Geometric Boolean Set and Constructive Solid Geometry operations, which are applied on a Binary Spatial Partitioning (BSP) organisation of the 3D scene being analysed [Requicha80,Foley90]. BSPs provide a much more efficient and general spatial partitioning scheme, for the implementation of Geometric Boolean Set operators, than the voxel organisation scheme [Thibault87] (such as the one used in [Tecoplan]). The IDL scripting language supports different types of geometrical interference tests: minimum

[1] This work is sponsored by the European project IST 26287 M3D "Multi-site cooperative 3D design system for architecture".

IDL layer	Element Class Name	IDL category	R Colour	G Colour	B Colour
AR: Architect	Interior wall	WALL-I	134	134	134
AR: Architect	Exterior wall	WALL-E	187	187	187
SR: Structural Engineer	Concrete Column	C-COL-	0	255	255
SR: Structural Engineer	Concrete Beam	C-BEAM	0	0	255
AC: Air-conditioning Engineer	Air ducts	DUCTS-	255	255	0

Table 1: Example of ISO Codes

distance checks, collisions, intersections between different types of AEC objects and in fact, any user-defined design verification rule that can be expressed by means of a geometric Boolean expression. Our surveys, have shown that architects, designers and engineers require the availability of an Automatic Design Verification tool, such as the one presented in this text, in their regular design process tasks [Dias99]. This ADV tool, in turn, requires programming scripts to be written in a specific technical-oriented language (IDL). This may be a difficult task to the above-mentioned users. To overcome this problem, we have developed Visual IDL, an intuitive graphical editor that allows a common user to write design verification tests, according to the IDL syntax notation, whilst maintaining a relative independency from the language constructs. IDL was conceived, taking in mind its extensibility potential: new objects produced by parts in motion of the assembled design can be used in the future in the context of IDL tests. The IDL language and ADV tool were developed in the framework of a European Union funded project, referred to as M3D (Multi-Site Cooperative 3D Design System for Architecture [M3D]). This project has recently been finalised by all of its developers and partners and approved by a European committee of expert reviewers. The developed M3D system aids architects and engineers, from several specialities, to work concurrently over the Internet, in the design of a full architectural and building construction project, allowing them to be geographically dispersed. M3D provides also the design team, with a multi-user 3D shared virtual environment, especially applicable to the AEC sectors [Luo00]. Each member of such a team can access a joint M3D session from different locations. M3D supports directly the Design Management process, providing a service that enables each specialist to insert 3D design work into a common web database. M3D also allows the aggregation of the several specialised projects, into a final integrated building construction project. In M3D, users are able to navigate in cooperative mode through the 3D virtual scene, pinpointing potential problems, exchanging ideas, comments and suggestions. Users can also communicate by attaching "post-its", linking several documents of different MIME types to 3D scene objects or by using more traditional communication means, such as text chat, audioconference and videoconference. Users are able to activate the ADV tool, whenever needed, that will check for geometric interferences among the different elements of the 3D scene, of different specialities (architectural, structural, water and sewage, etc) that comprise the building construction project. This tool, starts by analysing the whole geometric and topological description of the scene and organises it spatially using a BSP tree scheme. Afterwards, it interprets the IDL script previously selected by the user, which defines the design rules that must be enforced in the integrated project, and executes it over the scene graph, possibly computing the geometric interferences that violate those rules. If geometric interferences are indeed found, ADV adds them to the main 3D scene graph, for viewing purposes.

The paper is structured as follows: the syntax of IDL is synthetically explained in section 2. Visual IDL is then introduced in section 3, as a means to accelerate the development and execution of IDL scripts. This section emphasises in the description of the Visual IDL wizard service and in user interface issues. Section 4 briefly presents some results of interference detection tests, obtained by the execution of the ADV tool over a 3D scene. Section 5, provides a synthesis of the presented architecture and extracts some conclusions.

2. IDL SYNTAX

An IDL test is divided in two parts: the declarative part and the tests specification part.

2.1 The declarative part

In the declarative part, we first define the object we want to test (this is called the target object) and then which objects (the sources) will be tested against it. The objects are all named according to a specific ISO standard [ISO 13567]. The geometry is subdivided into a hierarchy of layers and categories. A layer is a portion of the architectural project that is done independently by one specific specialisation of the design team. As examples, we may have an architectural layer, a structural engineering layer or a water and sewage layer, among others. Each layer is further subdivided into a series of categories, that is, specific specialised project elements (also known as object classes). An architect could define categories like walls, stairs or window openings.

By definition, each object in M3D belongs to a specific layer and, within that layer, to a specific category. This classification of objects is necessary to properly specify the design verification tests. Along with the layer and category definition, an object is also defined by its colour code (in RGB format), which aids in the data exchange process with third party AEC CAD applications.

In Table 1, we show a sample of common ISO 13567 codes (or ISO codes for short) used in M3D. As we can observe in this table, for each layer there are several categories. Taking, as an example, an architectural interior wall: it belongs to the layer of architecture (AR) and category WALL-I, so its full ISO name is ARWALL-I. After declaring the target and the source objects classes in an ordered list, with the target category first, followed by the source categories, all separated by commas and using their corresponding ISO codes, a short string follows it, describing the test itself. This text is merely an aid for the user, since the ADV program will not act on it, as we can see in Table 2.

2.2 The tests specification part

In the tests specification part, it is possible to define rules involving different categories of objects, by means of expressions, written using the IDL syntax. IDL expressions, can be grouped in two types: geometric expressions, which always produce, as a result, a geometric object (that can possibly be a `null` object) and logical expressions, which evaluate to `true` or `false`. Different operators are available in IDL to write expressions. There are Geometric Boolean operators (`union`, `intersection` and `difference`), Unary operators (`scale`, `grow` and `shrink`), Relational operators ("==" and "`!=`") and Compound operators (`subtract` and `compose`). There is also an operator performing attribution. Geometric Boolean algebra is the underlying mathematical theory behind our approach to interference detection. It describes the basic set theory of mathematics but, in our case, applied to three-dimensional geometric objects. According to this algebra, several objects can be combined, with the help of *Geometric Boolean operators*, to produce new objects. With Geometric Boolean algebra it is possible to write an IDL expression like **C** = **A** *op* **B**, where **A** and **B** are two original objects, *op* is a Boolean operator and **C** is the object that results from applying *op* to **A** and **B**. The basic Boolean operators are demonstrated in Figure 1.

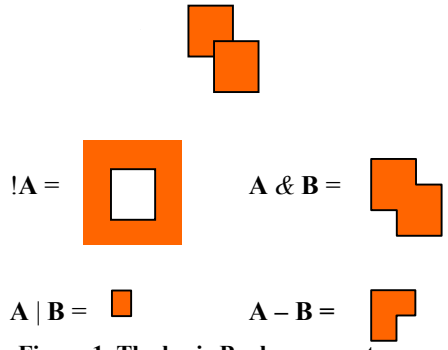


Figure 1. The basic Boolean operators.

The IDL language allows the results of geometric Boolean operations to be stored in new objects and these objects to be used in subsequent computations. In this way, it is equivalent to write:

```
func(object1 & object2), or
new_object = object1 & object2
func(new_object)
```

for some IDL operator `func`. In the remainder of this section, all the arguments to IDL operators that expect geometries use named objects but they can be replaced by any other valid IDL geometric expression, since it always evaluates to an object.

In design verification tasks, it is sometimes necessary to change the volume of an object by enlarging or shrinking it to some amount. This requirement has lead us to the definition of Volume Changing operators, which are unary operators since they take only one argument. As an example, let us suppose that a design rule for a building, expresses the requirement that a person measuring 2.0 meters tall, could climb the stairs without bumping his head on any other AEC element, such as a roof slab. One way to address this requirement is to take the stairs object and enlarge it by 2.0 meters along the vertical direction. This would define a safe volume for the stairs. If we then compute efficiently, a geometric intersection between this safe volume and all other objects in the scene, we will be able to detect if the building obeys this design rule. If an intersection is found, it means that some object of the building is inside the safe volume and the stairs don't have the necessary space for a 2.0 meters tall person to climb. Volume-changing operators are thus useful to define tolerance volumes around objects and check for minimum distance interferences. There are, at present, three volume-changing operators defined in the IDL language. These are, as mentioned, the `scale`, `grow` and `shrink` operators. They all take one object and produce a new object. The `scale` operator is the simplest one. It takes the form:

scale(object,percent), or:

```
scale(object,(percent_x, percent_y, percent_z))
```

The first form scales the object by a global percentage. Scaling by 200% will make the object twice as big, and scaling by 50% will scale it down to half its original size. The second form of the scale operator is similar but applies different scaling factors along the X, Y, and Z directions. Figure 2 gives two examples of the scaling operator.

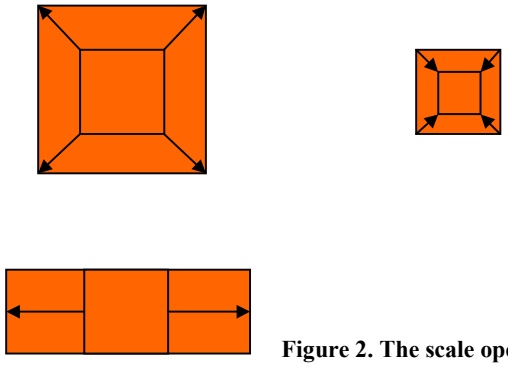


Figure 2. The scale operator.

Next, we have the `grow` and `shrink` operators. They enlarge or shrink, an object by some absolute amount along a chosen direction vector. The object is expanded, or reduced, only along the direction of the vector. They take the form:

```
grow(object,(vx, vy, vz))
shrink(object,(vx, vy, vz))
```

where (v_x, v_y, v_z) is the vector along which the object is enlarged or shrunk. The magnitude of this vector gives the amount of displacement of the object. Figure 3 gives examples of these two operators. In both cases, the object is displaced along the vertical direction by a distance of 1.0 meter. Note that, while the `scale` operator changes the volume of an object by a relative percentage value, the amount of displacement of the `grow` and `shrink` operators is absolute and expressed in meters.

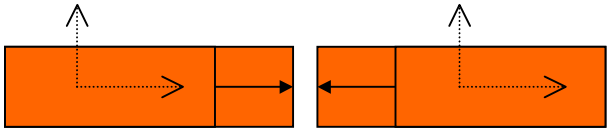


Figure 3. The grow operator for a parallelepiped object. The two dash arrows show two axes of symmetry for the object. The longest arrow is the main axis of symmetry.

These two operators still take another form, which can only be applied to parallelepiped objects. Parallelepipeds are fairly common objects in building projects. Most structural pillars and beams take their shape. Parallelepipeds have a simple geometric structure and it is possible to compute their three axes of symmetry. The `grow` and `shrink` operators, when applied to parallelepiped objects, can take the special form:

```
grow(object, dist)
shrink(object, dist)
```

where it is understood that `object` must return a parallelepiped. The object is enlarged, or shrunk, along its main axis of symmetry by the absolute amount *dist*, in meters. Figure 3 gives an example for the case of the `grow` operator. The `shrink` operator works the same way, with the difference that the volume of the parallelepiped is reduced, rather than expanded. Note that growing or shrinking a parallelepiped by an absolute displacement vector is still a valid operation. So, this special kind of objects can use either type of the volume-changing operators. The displacement amount *dist* can take positive or negative values. If *dist* is positive, the facet of the parallelepiped in the direction of the main axis of symmetry is selected for enlargement or shrinkage. If *dist* is negative, the facet, opposite to the main axis, is selected. Unfortunately, this special form of growing and shrinking operators cannot be used for general objects because these do not always have well defined axes of symmetry.

The Relational operators, used in IDL logical expressions, are the equality and its counterpart, the inequality operator. They are both used to compare two objects. The result of these operators is not another geometric object, but a Boolean value (true or false), depending on the result of the comparison. The equality operator takes the form:

```
object1 == object2
```

and the inequality operator has the form:

```
object1 != object2
```

Often one wishes to compare an object against the empty set. In that case, the special keyword `null` is used to represent the empty set, and the comparison becomes:

```
object == null
```

and equivalently for the inequality operator. These two operators are almost always used in the `return` statement of an IDL test, which will be explained later.

Lastly, the Geometric compound operators operate on whole categories of M3D objects. They are called *compound operators* because they execute, repeatedly, a sequence of basic Boolean operators over all the objects belonging to some given category. At present there are two compound operators defined in the IDL language: the `compose` operator and the `subtract` operator. The compose operator can take two forms:

```
compose(object, category)
```

or:

```
compose(object,(category, category,…,category))
```

This operator computes the intersection of an object of a target category against all the objects belonging to the specified source categories. The first form of the operator above computes the intersection against a single source category, while the second form specifies a list of source categories, enclosed in parenthesis and separated by commas.

To be more specific, if `object` is some geometric object of a target category and `category` is a list of objects $\{v_1, v_2, \ldots, v_n\}$ belonging to some specific source category, the result of the `compose` operator is:

```
compose(object, category)
=(object & v1)|(object & v2)|…|(object & vn)
```

It is the union of all intersections between the first object and all the objects from the source category. This is a useful operator when we wish to check if an object intersects a whole category of objects (interior walls or floor slabs, for instance). It would be tiresome to write explicitly the whole Boolean expression on the right side of the above equality. The `compose` operator conveniently encapsulates such a complicated expression in a simple form.

The `subtract` operator is written is a similar manner:

```
subtract(object, category)
```

or:

```
subtract(object,(category, category,…,categ))
```

This operator takes an object and removes all the portions of its volume that are intersected by objects from the specified categories. Again, this operator can be expanded according to the expression:

```
subtract(object, category) =
  object – v1 – v1 - … - vn
```

where, as before, $\{v_1, v_2,\ldots, v_n\}$ is the list of objects belonging to a source `category`. The `subtract` operator is useful to remove portions of an object that are common to another category, or categories, of objects.

And finally, at the end of every IDL test there must be a `return` statement. This `return` statement determines if the test failed (a geometric interference was found) or not (the target object does not violate the specified design rule). It contains a Boolean expression featuring one of the two equality operators. The result of this Boolean equality, or inequality, is the final result of the test. The `return` statement takes the form:

```
return error_object if test_object == null;
```

or:

```
return error_object if test_object != null;
```

The `error_object` in the above `return` statements represents the geometric shape of the inconsistency. It is sent to the M3D editor and attached to the scene graph to be visualized, if the Boolean expressions evaluate to `false` for the test object. If we consider again the case of checking if concrete columns are inside of walls, the geometry of the inconsistency will be the portion of a column that is outside of all the walls. This volume of the pillar is flagged in the editor with a blinking colour or a dashed shape for easy visualization. The `test_object` in the `return` statement is what determines the outcome of the test. It should either be equal or different from the empty set, depending on the version of the `return` statement that is used. As an example, in Table 2, we have a typical test performed on the LuisaZ[2] project.

After analysing the IDL script, we can see that it describes a test to verify if air-conditioning ducts (target category) intersect with structural beams (source category). The test simply checks for an interference between these two object categories (using the `compose` operator) and scales the result by a factor of 110%, so that the interference will be enlarged in order to see it more easily in the M3D Editor. Note that the target object category (ACDUCTS-) is referenced in the test specification part by the $$ symbol (which evaluates to all the objects of its category), and all the source objects, of given categories, are identified, respectively, by $1, $2, $3, etc.

[2] Luisa Zambuginho or LuisaZ is a small house project developed by Architect Jorge Silva from Oficina de Arquitectura, Lisbon and used as one of the benchmarks for M3D

```
DEF
v!ACDUCTS-,SRC-BEAM!
Air_ducts_intersects_concrete_beams
Info
{
        string "
        result = compose($$,$1);
        return scale(result, 1.1) if result!=null;
        "
}
```

Table 2: Example test

As we have seen, the IDL language, although powerful and flexible in the description of geometric-based design rules, can be complex for non-technical users. It requires some "a priori" knowledge and understanding of the language constructs, and of geometric Boolean operations, to create the desired design verification tests. We obviously recognized that this was too much of a burden to the users that just wanted to check for interferences between specific objects of their 3D scenes. Users should not be required to learn the language syntax in detail to be able to perform automatic design verification in building construction projects. The Visual IDL tool was created, to support this user requirement.

2.3 Development of the IDL language

The IDL language was implemented as a Yacc-based grammar file. This file was translated into a code parser by the Yacc tool and this parser was then integrated into the rest of the application.

The IDL scripts are translated internally by the application into a tree of operators. Each IDL operator has one geometric output and one or more inputs. We build the tree by instancing such operators and connecting the outputs of some to the inputs of others. Invoking the operator at the top of the tree will cause the subsequent invocation of all the other operators beneath it. The operators at the bottom of the tree receive the original objects that comprise the geometry of the building construction project. These objects are processed through the chain of operators and the operator at the top will return the final geometry of the IDL test implemented by the tree.

Using this implementation strategy, it is very straightforward to extend the IDL language with new operators. This was a need that had been envisaged from the early stages of design of the language. One could not foresee all the types of geometric tests that architects and engineers would wish to use on their construction projects. Whenever some functionality is desired that is not already supported by the language, one or more appropriate operators can be designed, its syntax coded into the Yacc grammar file and a corresponding instance added to the tree of operators.

3. THE NEED FOR EFFICIENCY IN DESIGN VERIFICATION

Visual IDL was developed to allow the users to create their design verification rules (or tests) rapidly, efficiently and (as this is always desirable) without too much effort. In fact, most of the time a user just needs to make a typical test (like the ones using the `compose` operator) and doesn't require anything too elaborate. So, Visual IDL is a simple and intuitive, yet powerful editor allowing a user to write simple as well as complex design verification rules. With this tool it's possible to create new test files, save them to disk, open existing files, add, modify and remove tests, etc. Its user interface is shown below:

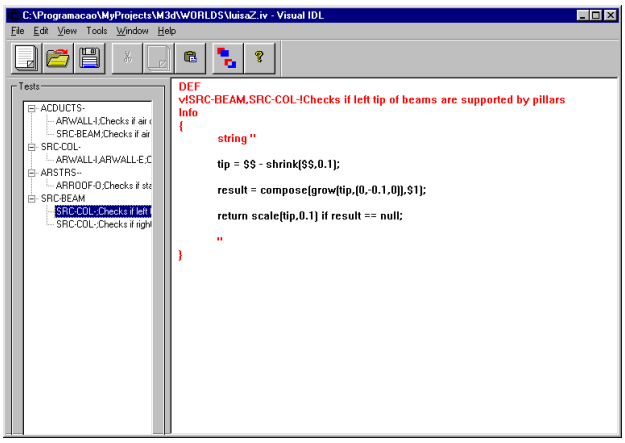


Figure 4: The Visual IDL program

3.1 A Closer Look to Visual IDL

The main Visual IDL window shows the currently active test, which is selected in the tree bar, at the left. Only one test is displayed at a time, as this is easier to understand and to work with. The tree bar displays all the tests and is organized as follows: the main nodes of the tree show the target object category of the test. By opening the node, the source object categories are displayed. By clicking on one of these child nodes, the corresponding test is displayed in the main window, replacing the previous one. Therefore, the tests are organized in the tree bar in a hierarchal mode, giving the user an overall look and an ordered view of the test file. As mentioned, in order to write a test, the user must know the object category to be tested (the target), the objects categories which will be tested against it (the sources) as well as what kind of test he would like. It can be a simple `compose`, a `scale`, `grow` or even `shrink` type of test or a programmed mixture of the above, written in IDL script. To aid this task we have developed a Visual IDL wizard.

3.2 The core service: the Wizard

The Visual IDL wizard directly helps the user to write the test. It will take him in a series of steps, posing several questions about what kind of test the user wants and then creates it, with little effort from the user. To work with the wizard, Visual IDL needs to be launched from the M3D Editor application, which must have a 3D architectural scene already opened. This scene can either be complex or relatively simple like the one of Figure 5.

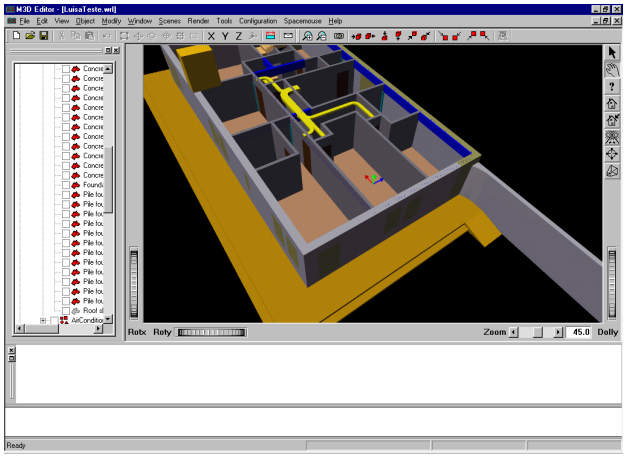


Figure 5: The M3D Editor application with an open scene

This scene is made of several AEC objects, which are all correctly "labelled" with their corresponding ISO codes. For example, the yellow air ducts have an ISO code of ACDUCTS-. A wall has an ARWALL- ISO name attached. Internally, the M3D Editor, the ADV and Visual IDL recognize each object by these eight character names. Since the M3D Editor and Visual IDL are separate applications, M3D must hand-over to Visual IDL all the ISO names that are currently in the scene. Therefore, there's a "two-way communication channel" between the two applications for data transfer and synchronization of tasks.

3.3 A Step By Step Tour

Once the user has the scene loaded, he may want to see, as an example, if the air ducts intersects with any of the interior wall objects. This is an easy and typical task for Visual IDL. After it's launched from M3D, and the wizard is activated, the user will be able to choose which will be the target and the source object categories. These are displayed in a list of checkboxes according to their ISO names (see Figure 6).

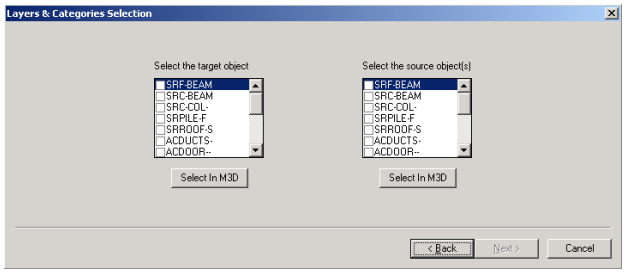


Figure 6: Selecting objects in the Wizard

There are two possible ways of choosing the desired objects. If their exact ISO codes are known, it is possible to choose them directly in the wizard checkboxes. However, users may also select objects by picking operations in the M3D Editor scene. This can be done for the target as well as the source objects. In this case, the Visual IDL window will reshape to a smaller size, put itself to the side of the screen and give the input focus to the M3D Editor (Figure 7).

```
DEF
v!SRC-BEAM,SRC-COL-
!Test1_Tests_if_left_tip_of_beams_are_supported_by_c
olumns
Info
{
        string "
        large = grow($$,(0.0,-0.1,0.0));
        short = shrink(large,0.1);
        tip = large - short;
        error = $$ - shrink($$,0.1);
        return error if compose(tip,$1) == null;
        "
}


DEF
v!SRC-BEAM,SRC-COL-
!Test2_Tests_right_tip_of_beams_are_supported_by_col
umns
Info
{
        string "
        large = grow($$,(0.0,-0.1,0.0));
        short = shrink(large,-0.1);
        tip = large - short;
        error = $$ - shrink($$,-0.1);
        return error if compose(tip,$1) == null;
        "
}


DEF
v!ACDUCTS-,SRC-BEAM
!Test3_Checks_if_air_ducts_intersect_beams
Info
{
        string "
        result = compose($$,$1);
        return scale(result,1.1) if result != null;
        "
}


DEF
v!ARSTRS--,ARROOF-O
!Test4_Tests_stairs_have_enough_space_above_them
Info
{
        string "
        large = grow($$,(0.0,1.0,0.0));
        intersect = compose(large,$1);
        return scale(intersect,1.1)
                if intersect != null;
        "
}
```

Table 3: Test file

In this example, the user has chosen air ducts (the yellow object) and one of the walls surrounding it. The wizard activates the corresponding checkboxes: ACDUCTS- and ARWALL-, which in fact means all objects of those categories. To create more specific tests, that deal only with limited set of objects, and not the entire category, the user may select in the Editor's scene tree, which particular objects he doesn't want to consider for the test. After this stage, the user is prompted for a short text that describes the test.

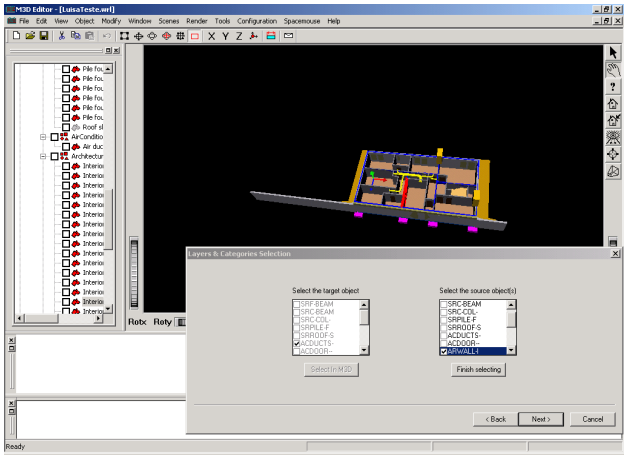


Figure 7: Picking objects interactively in M3D

Finally, the user chooses the type of high-level test to be performed as shown in Figure 8.

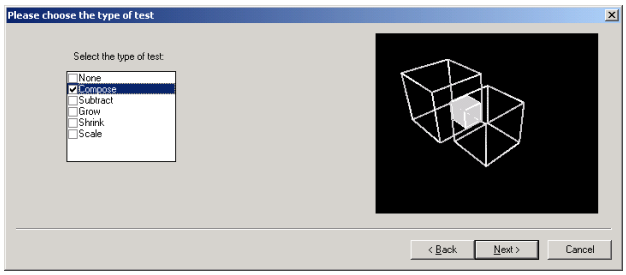


Figure 8: Choosing the type of test (with a graphical rendering aid)

On the left side, the user can choose the desired test. In the middle of the dialog, several optional parameters can be set according to the type of test chosen. And, in the right side, there's an OpenGL frame window graphically displaying in an interactive animation, the type of test and its effect on the geometry. As several values of the parameters are changed, the animation changes accordingly, enabling the user to have full knowledge of what the test will perform, and how. Whenever a simple intersection test is required, the `compose` type of test must be selected, as explained in section 2.2. After this stage, the wizard finishes and writes the IDL code, according to the target and source object categories chosen, the description entered and the type of test specified (and its parameters). After this process, the user can alter what the wizard has produced. The text in black (test specification section) can manually be changed. However, the text in red (declarations section) cannot be changed so easily. This is to prevent inexperienced users to accidentally change the header of the test. In order to change it (the target and source objects as well as the description text) the user can just double-click on the

red text and the wizard will pop up. This time, however, it will help him change this information according to his needs. The steps are identical so we will not cover them again. Users with a fairly good knowledge of the IDL language can, consequently, use the wizard to write the basis for a typical design verification test and then, change the body of the test to more complex IDL statements according to their needs. So, while writing typical and more common tests for most of the users, the wizard also provides advanced users with the 'skeleton', or 'building block', of a more powerful and possibly more interesting test.

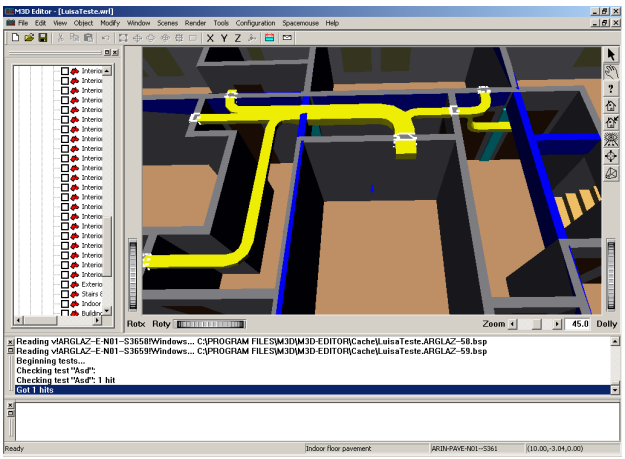


Figure 9: The interferences calculated by ADV

3.4 Performing Automatic Design Verification

After the definition of an IDL script file, the user may invoke the ADV program directly from the M3D editor, as an asynchronous process, which uses, as arguments, this file and a pointer to the 3D scene graph. This way, the editor does´nt block while waiting for the completion of the automatic design verification process. The input data format of the 3D scene is VRML, so ADV requires an initial stage where the proprietary data formats corresponding to the different specialised projects (produced by native CAD applications), must be converted into the VRML format, using the guidelines for layer naming convention described in the ISO 13567 standard. The input shape objects are described according to a BREP format. However, ADV requires a BSP spatial description. Therefore, a filter is executed that converts the VRML format with a BREP organisation, into the same format, but with a BSP tree organisation. Note that, this pre-processing stage is skipped by our algorithm, if the same IDL test is executed for the same 3D scene, a second time, and no shape is altered, since we maintain a cache of the BSP tree of the scene. After this stage, ADV parses the IDL file and executes the geometric-based design verification tests, as stated in the IDL script. It will eventually finish to calculate the geometric interferences, organised as a hierarchy of VRML nodes under a common VRML SoSeparator node. These are the geometries that have originated `false` results in the IDL logical expressions, thus corresponding to the shapes that fail the IDL design verification tests. These interference results will then be added to the M3D Editor's scene graph and visualised, highlighted in white (see Figure 9), under a distributed multi-user collaborative environment [Luo00]. As we can see in this figure, ADV has detected where the air-ducts intersect the walls (in the white areas).

4. RESULTS

Our object classification scheme, the Visual IDL wizard and ADV technique were extensively used in different stages of several building construction projects, between March 1999 and March 2001. Different specialities projects were fully designed in 3D, using proprietary CAD packages and later exported to M3D. An international design team was lead by Architect Jorge Silva from Oficina de Arquitectura, OA in Lisbon. The involved specialities included architecture, water and sewage (OA, Portugal), structural engineering (Betar, Portugal), electricity (IDOM, Spain) and air conditioning (ARQMAQ, Spain). The users, connected by means of a private IP (over ISDN) network, have performed in their offices at their own locations, the design tasks that correspond to different stages of the following projects:

- Luisa Zambujinho/LuisaZ, Almada - March 1999
- Aveiro Pavilion - July 1999
- School Pavilion Tiana, Spain - November 1999
- Mirante Library, Sintra - November 1999 (first phase)
- House in Palma de Mallorca- March 2000
- Industrial Pavilion in Barcelona- September 2000
- Mirante Library, Sintra - November 2000 to March 2001 (second phase)

For the specific case of the Mirante Library, starting from November 2000 up to March 2001, the user group made intensive international connectivity tests and online synchronous co-operative work sessions. The trials occurred weekly mainly among three locations: Lisbon (OA), Palma (Arqmaq) and Barcelona (Idom). Special user trial scripts where developed to structure the online sessions, which were used to test and fine tune the performance of the M3D Editor, ADV algorithm and Visual IDL tool and our AEC project integration technique, within a distributed collaborative work session, served by a central database system. During these real-life user trials, we have performed more than 20 design verification tests with success (that is, we have found a number of design-related issues, that were subsequently solved).

ADV and Visual IDL, were judged by the users has the "most innovative functionality of all the modules" of the M3D system (comprising Multi-user Editor, Database, Conference Management, ADV & Visual IDL) [Fonseca01], enabling operations of intersection, subtraction, reunion, growing, safe volume computation, etc, with AEC shapes. With these tools, users were able to check if the 3D geometries of different specialities

fulfilled the rules established by the architects, designers and engineers, or if, within the same speciality, the normative, geometric or common practice rules were verified. The possibility of discussing and checking in group and in real time, the found interference detection results, makes it, according to the users [Fonseca01], "an effective toll for validating the design integration of different specialities within a building construction project".

The following figures (Figures 10 to 13) depict the ADV results, which where embedded in the M3D Editor scene graph, that correspond the IDL test file of table 3 (section 3.4), used for the LuisaZ case. This was a small project with 1760 triangles, including architecture and structural engineering. In table 4 we present the CPU times used by our ADV algorithm, which we have split in the following phases:

- Phase 1: BREP to BSP conversion and creation of BSP files in the cache.
- Phase 2: Reading the BSPs files from the cache and parsing the IDL script file.
- Phase 3: Performing geometric interference tests.

In table 5, we present the CPU times, of executing the same IDL test in the LuisaZ project, but for the second time this test was executed, that is, when the BSP tree cache was already pre-calculated. The results show an average reduction of 70% in the total CPU time, used by the ADV algorithm, relatively to the cases of Table 4 (with no BSP files yet available in the cache).

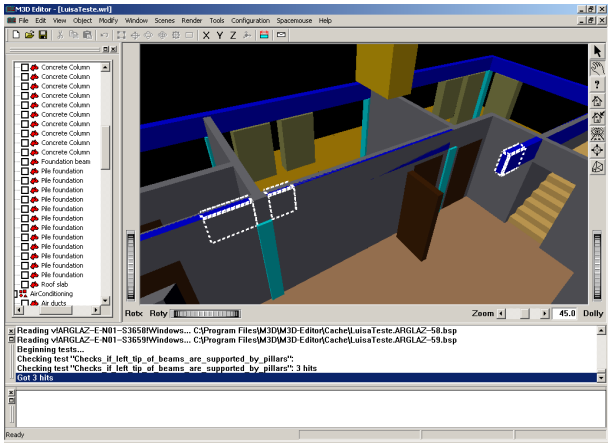


Figure 10: The interference results of test 1. The left tip of these beams are not supported by the columns, but by the walls.

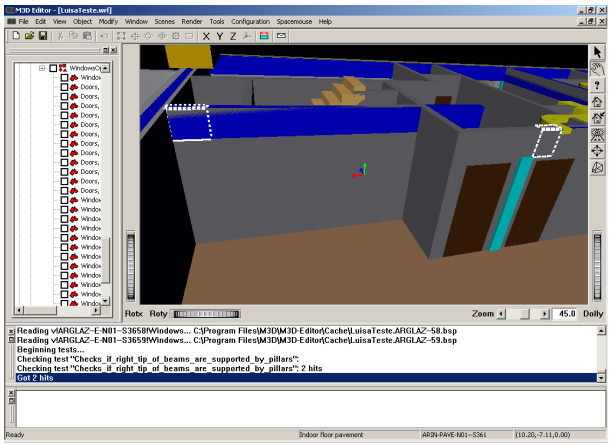


Figure 11: The interference results of test 2. The right tip of these beams are also not properly supported by the columns.

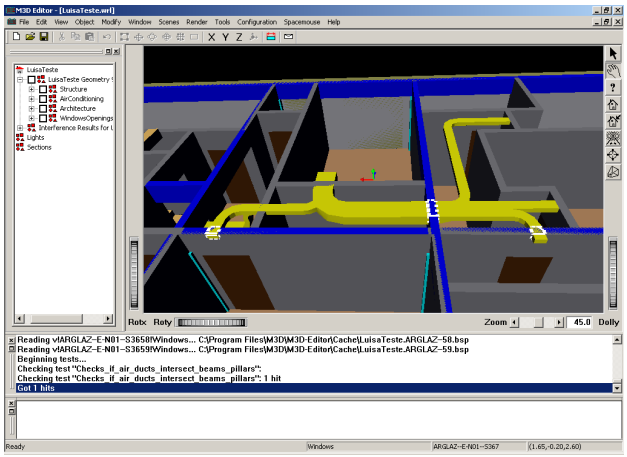


Figure 12: The interference results of test 3. As shown, the air ducts are intersecting the beams.

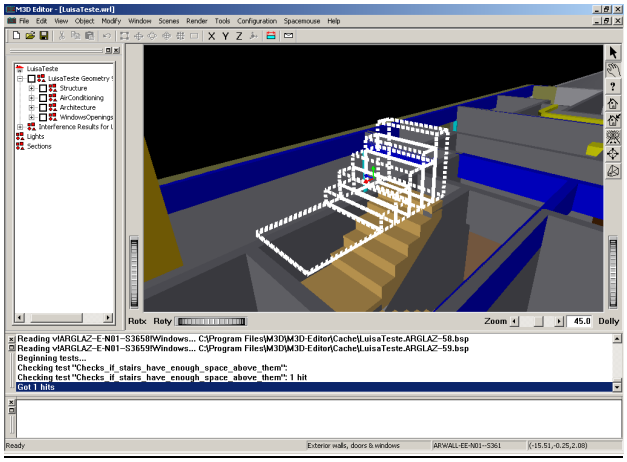


Figure 13: The interference results of test 4. According to the test, the stairs don't have enough space between them and the roof.

CPU time in *ms*	Phase 1	Phase 2	Phase 3	Total
1st Test	1471	171	425	2067
2nd Test	1463	160	418	2041
3rd Test	1451	198	473	2122
4th Test	1493	185	594	2272

Table 4 CPU time used by our ADV algorithm in the LuisaZ project with an initially empty BSP tree cache

CPU time in ms	Phase 1	Phase 2	Phase 3	Total
1st Test	17	154	412	583
2nd Test	17	168	418	603
3rd Test	17	162	431	610
4th Test	18	164	546	728

Table 5 CPU time utilised by our ADV algorithm in the LuisaZ project with an initially complete BSP tree cache

5. CONCLUSIONS

The IDL language has proven to be powerful yet flexible enough to be extensively used by the M3D project architectural and engineering users, such as Oficina de Arquitectura and Betar in Lisbon, or Arqmaq and Idom in Spain. With IDL, users can write tests that perform complex, valuable design verification tasks on geometric data. Great care was taken to ensure that the language remains consistent and yet powerful and efficient. With IDL, the ADV application has a standardized manner of knowing how to act on the geometry. The ADV tool reads an input IDL script file, parses it, analyses the tests according to the IDL syntax and calculates the interferences, attaching these to the M3D Editor's scene graph. Visual IDL was introduced as a bridge between the user and IDL. Geometric interference tests can now be designed, developed and deployed more easily and efficiently. Specifically, the wizard helps common users to write their typical tests with a fairly acceptable degree of complexity and also helps more advanced users with a good knowledge of the language, to write a simpler prototype version of a test first, allowing subsequent developments of the test, perfecting it and improving it until it's finished. Lastly, the wizard itself makes it more easier to choose the target and source objects (by direct picking operations) and the OpenGL frame window provides a clear, clean and simple vision of the types of tests, their operation and their final effects on the objects.

Future activities are planned to address the analysis of temporal-spatial conflicts (and not just spatial, as for the presented results) in design verification and planning tasks [Akinci2000]. These may occur during the time planning of the building construction operations, which involve design elements, construction systems, dedicated machinery, necessary operations, safety regulations, materials and workers. All these entities impose specific and well known time, volumetric and spatial constraints, as well as safety zones, in the construction site time-space continuum. These must be studied concerning its geometric interference, in order to improve safety, for a better humanisation of the construction site workplace, as well as to enhance the quality of the building construction.

ACKNOWLEDGEMENTS

We would like to thank, in no special order, Rui Silvestre, Rafael Bastos and Luís Monteiro for their ideas, comments, suggestions and support, as well as António Lopes and Paulo Lopes for their early work in the project. A special word of gratitude goes to the Oficina de Arquitectura team, namely Arq. Jorge Silva and Zé Manel, as well as for the Engº Miguel Vilar from Betar, for their ideas and requirements that are in the genesis of the M3D system and of ADV and Visual IDL.

6. REFERENCES

[Akinci2000] Akinci, B., "4D workplanner - A prototype System for Automated Generation of Construction Spaces and Analysis of Time-Space Conflicts", ICCCBE-VIII, Stanford, pp 740-747, August 2000

[Blaxxun] (www.blaxxun.com)

[Dias 99] Dias, J. M. S., Gamito, M., Silva, J., Fonseca, J., Luo, Y., Galli, R., "Interference Detection in Architectural Databases", Short paper Eurographics 99, Milan, September 1999

[Foley90] Foley, van Dam, Feiner, Hughes, "Computer Graphics – Principles and Practice – Second Edition", Addison-Wesley Systems Programming Series, 1990

[Fonseca01] Fonseca, Z, Silva, J., Torres, I., Vilar, M., Castañeda, D., Albiol, M., "Report on the M3D user trial evaluation", ESPRIT 26287, Deliv 3.2, M3D/OA/WP5/T5.3/2001/DL, March 2001 1990

[ISO 13567] ISO standard 13567 "Technical product documentation-Organisation and naming of layers for CAD"

[Luo00] Luo, Y., Galli, R., Sanchez, D., Bennassar, A., Almeida, A. C., Dias, J. M. S., "A Co-operative Architecture Design System via Communication Network", ICCCBE-VIII, Stanford University, USA, August 2000

[M3D] www.m3d.org

[Requicha80] Requicha, A.A.G., "*Representations for Rigid Solids: Theory, Methods and Systems*", Computing Surveys, 12(4), pp. 437-464, December 1980

[Tecoplan] www.tecoplan.com

[Thibault 87] Thibault, W.C., Naylor,B.F., "Set operations on polyhedra using binary space partitioning trees", Computer Graphics (SIGGRAPH '87 Proceedings) (July 1987), M.C. Stone, Ed., vol 21, pp.153-162

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Visualização Interactiva Tridimensional em Java3D

Bruno Caiado*
Lic. Engª. Informática, IST
Av. Rovisco Pais, 1049-001 Lisboa
bruno.caiado@megamedia.pt

Luís Correia*
Lic. Engª. Informática, IST
Av. Rovisco Pais, 1049-001 Lisboa
luis.correia@megamedia.pt

João Brisson Lopes
Dep. Engª. Informática, IST
Av. Rovisco Pais, 1049-001 Lisboa
brisson@ist.utl.pt

* presentemente: Megamedia,
R. António Pedro 111, 1º, 1150-045 Lisboa

Sumário
Esta comunicação descreve a concepção e realização em Java/Java3D da aplicação NISVAS para visualização interactiva de conjuntos de dados de grande volume resultantes da simulação numérica de fenómenos complexos. A aplicação, que é portável, visualiza estes dados através de cores, símbolos, vectores e animações. Descrevem-se os objectivos, a arquitectura e a funcionalidade do visualizador, com realce para a realização da interface com o utilizador que é simples, de fácil aprendizagem e suporta utilizadores com diferentes níveis de experiência. As reacções dos utilizadores permitiram concluir que os objectivos propostos foram alcançados, nomeadamente no referente ao desempenho e à simplicidade e flexibilidade de uso. Apresentam-se direcções para a continuação deste trabalho.

Palavras-chave
visualização interactiva, visualização científica, Java, Java3D, interacção, interface ao utilizador

1. INTRODUÇÃO

A simulação de fenómenos complexos tridimensionais origina conjuntos de resultados numéricos cuja análise e compreensão é muito difícil devido ao grande volume e à natureza diversa dos muitos resultados obtidos. Em fenómenos como, por exemplo, a deformação de estruturas em impactos violentos ou o escoamento num leito variável das águas de um rio durante uma cheia, obtêm-se campos de valores de grandes dimensões de deformações, estado de tensão, temperatura, velocidades e concentrações, que, muitas vezes, variam no tempo.

A solução para este problema consiste em visualizar conjuntamente os resultados, combinando o que é visível (uma peça deformada) com o que o não é, como o estado de tensão em toda a peça, "visualizando o invisível".

O número de grandezas escalares e vectoriais a visualizar pode facilmente atingir e ultrapassar a dezena de grandezas. O utilizador necessita dispor de uma interface simples e poderosa que lhe permita mudar rapidamente a forma como cada grandeza é visualizada ou substituí-la por outra. Além disto, a interface deve permitir ao utilizador o reconhecimento imediato das grandezas que pode manipular e que operações de visualização pode com elas efectuar. Isto coloca requisitos especiais ao desenho da interface, nomeadamente na apresentação do estado da aplicação, na sugestão de operações e na prevenção de erros.

Tradicionalmente, as aplicações de visualização são aplicações especializadas, dependentes de tecnologias proprietárias, do sistema operativo e do hardware gráfico, pois exigem grande capacidade de processamento gráfico. Em consequência, são aplicações concebidas para plataformas específicas e de custo elevado.

A dependência da plataforma também tem como consequência tornar a interface com o utilizador dependente do sistema de janelas da plataforma seleccionada que, por sua vez, impõe um "look and feel" específico à interface. É o caso de aplicações de visualização como o Wavefront Data Visualizer, PV-WAVE, ou VIS5D, entre outros. Este é também o caso do visualizador ISVAS (Interactive System for Visual Analysis, [Karlsson93], [Karlsson94], [Haase95]) que depende de hardware gráfico Silicon Graphics e da API GL, para a apresentação das visualizações sob o sistema de janelas 4Dwm do sistema operativo Irix, e do sistema de janelas Motif para o diálogo com o utilizador. Este tipo de solução era perfeitamente justificável na altura em que aquele visualizador foi concebido.

Entretanto, a capacidade do hardware aumentou e verificou-se igualmente uma evolução significativa dos paradigmas de construção de software e da interacção pessoa-máquina que se traduziram na adopção da Programação Orientada para Objectos e na separação clara entre o software gráfico e o hardware de saída gráfica. A generalização destes paradigmas e do uso de linguagens como C++, OpenGL, Java ou Java3D permitem desenvolver

aplicações de visualização de construção mais simples e mais fáceis de manter cujas interfaces com o utilizador mantêm o mesmo "look and feel" independentemente da plataforma que, no caso de Java, não é necessário recompilar quando se muda de plataforma.

Esta comunicação apresenta uma aplicação para visualização interactiva de dados de grande volume desenvolvida no âmbito de um Trabalho Final de Curso da Licenciatura em Engenharia Informática do Instituto Superior Técnico [Caiado01].

O desenvolvimento desta nova aplicação para visualização, NISVAS (New Interactive System for Visual Analysis), constituiu um desafio importante. Isto deveu-se não só ao objectivo de desenhar e implementar uma interface simples de aprender e utilizar, como também porque a linguagem que veio a ser adoptada para realizar a sua implementação, Java3D, representa ainda hoje uma tecnologia emergente cuja evolução continua.

2. OBJECTIVOS

A aplicação de visualização desenvolvida, NISVAS, partiu da análise da funcionalidade apresentada por aplicações para visualização existentes, principalmente a aplicação ISVAS, e de inquéritos realizados junto de utilizadores que permitiram identificar muitos dos problemas sentidos por estes. Desta análise resultou um conjunto de objectivos centrados em:

- Portabilidade
- Interface com o utilizador
- Modularidade
- Desempenho
- Documentação

2.1 Portabilidade

A maioria das aplicações para visualização interactiva não é portátil e, em muitos casos, emprega normas da indústria com elevado peso histórico que limitam a sua disseminação e utilização. Por este motivo, definimos como objectivo que a aplicação a desenvolver fosse implementada de forma potenciar a sua portabilidade. Assim, logo à partida, o leque de opções foi reduzido ao emprego de ferramentas como C++, OpenGL, Java e Java3D.

2.2 Interface com o Utilizador

Um dos maiores problemas encontrados em aplicações de visualização interactiva refere-se às interfaces com o utilizador que, em geral, são pouco intuitivas, de difícil aprendizagem e utilização e, não raras vezes, pouco consistentes. Um exemplo disto é o caso da aplicação ISVAS que, mesmo para a execução de comandos simples, exige um conhecimento apreciável. Este visualizador apresenta ainda uma profusão de diálogos com o utilizador com muito pouca consistência e de diferente apresentação no caso de diálogos semelhantes. Entre outras coisas, os utilizadores inquiridos salientaram o facto de a interface desta aplicação exigir bastante memorização e de nela se perderem muitas vezes, mesmo quando só pretendiam conhecer o estado da aplicação.

Uma situação que os utilizadores também apontaram como factor negativo foi a ordem inflexível de carregamento de ficheiros constituindo um conjunto de dados e de terem que explicitamente nomear todos os ficheiros a carregar.

Face aos comentários recolhidos junto dos utilizadores, estabelecemos então que a interface com o utilizador da nova aplicação teria que proporcionar diálogos simples e consistentes, fosse fácil de aprender e utilizar e, além disso, permitisse uma visão coerente do estado da aplicação. A interface deveria igualmente proporcionar atalhos (teclas de atalho e aceleradoras, por exemplo) e redundância na forma de realizar as tarefas, de modo a permitir o seu emprego quer por novos utilizadores, quer por utilizadores experientes.

A interface deveria também permitir a simplificação e a automatização de tarefas complexas, como o carregamento de conjuntos de dados por meio de um único comando.

2.3 Modularidade

Um outro objectivo fixado foi a modularidade da arquitectura. Com este objectivo pretendia-se uma identificação muito clara dos diversos módulos constituintes da aplicação e da sua funcionalidade, facilitar a reutilização de código, a sua manutenção e aumento de funcionalidade. Em relação à interface com o utilizador, a opção pela linguagem Java orientada para objectos pretende atingir os mesmos objectivos, além da segurança e robustez que proporciona.

2.4 Desempenho

O desempenho das aplicações para visualização é um factor crítico essencial a ter em conta na sua concepção e realização, dadas as exigências do processamento gráfico em termos de recursos. Isto significa que todas as opções a tomar, principalmente a solução para a arquitectura, deveriam ter em conta o seu impacte no desempenho e a necessidade de encontrar o equilíbrio entre facilidade de desenvolvimento, usabilidade da interface com o utilizador e o desempenho global.

2.5 Documentação

A maioria das aplicações para visualização interactiva é acompanhada por manuais extensos e complexos, de difícil leitura, e a ajuda on-line disponível é normalmente insuficiente. Deste modo, definiu-se como objectivo a atingir que a aplicação a desenvolver proporcionasse ajuda contextualizada, para além da documentação completa acessível a partir da aplicação, com vista a facilitar e diminuir o período de aprendizagem.

3. ARQUITECTURA

A arquitectura da aplicação NISVAS divide-se em dois subsistemas: a interface com o utilizador e o núcleo gráfico. A comunicação entre os dois subsistemas é realizada por módulos de comunicação, tal como a figura 1 apresenta.

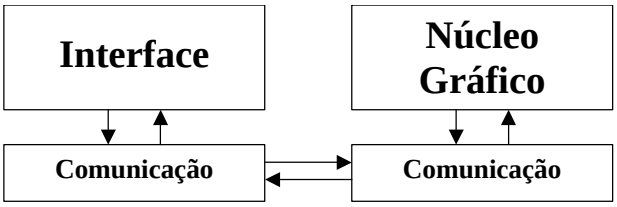


Figura 1 – Arquitectura geral da aplicação NISVAS.

O subsistema de interface é responsável pela interacção com o utilizador, pelo desencadeamento dos respectivos comandos, pela prevenção de erros e por transmitir ao utilizador todos os resultados não gráficos dos comandos executados.

O subsistema núcleo gráfico é responsável pela execução dos comandos gráficos e pela construção e disponibilização de todas as opções de visualização dos conjuntos de dados a visualizar. É também responsável pela criação e destruição das janelas de visualização gráfica e pela gestão de eventos ocorridos nessas janelas.

Esta solução para a arquitectura segue o paradigma da arquitectura cliente/servidor, em que a interface com o utilizador desempenha o papel de cliente e o núcleo gráfico o papel de servidor. A interface processa os eventos ocorridos nas suas componentes, enquanto o núcleo gráfico processa graficamente os pedidos que lhe chegam. A realização da comunicação e do protocolo de comunicação entre os dois subsistemas dependia da arquitectura adoptada.

3.1 Primeira Solução para a Arquitectura

Numa primeira aproximação, a arquitectura poderia consistir em dois subsistemas comunicando entre si por meio de sockets. Esta solução permitiria a portabilidade de um núcleo gráfico baseado em OpenGL e uma interface baseada em Java.

As vantagens desta solução consistiriam na portabilidade total (sem recompilação) da interface com o utilizador e na portabilidade do código fonte do núcleo gráfico proporcionada pela linguagem C++ e pela norma da indústria OpenGL. Esta solução exigiria uma definição muito clara do protocolo de comunicação dos dois subsistemas, mas permitiria a sua total modularidade.

A grande desvantagem apresentada por esta solução consistia no facto de a comunicação por meio de sockets ser difícil de portar, obrigando à reescrita dos módulos de comunicação. Mas, mais crítico ainda, o desempenho da aplicação seria afectado dado que se previa que a comunicação entre os dois subsistemas viesse a ser bastante intensa, o que foi confirmado em testes realizados com protótipos.

3.2 Solução Adoptada para a Arquitectura

A solução alternativa consistiria em implementar toda a aplicação em Java [Arnold96], com o núcleo gráfico empregando a API Java3D [Bouvier99]. Além do desafio constituído pelo emprego de uma tecnologia emergente, esta solução apresentava as vantagens de uma total portabilidade da aplicação (sem necessidade de recompilação), ser totalmente orientada para objectos, facilitar a integração de novas funcionalidades e da simplificação do protocolo de comunicação entre a interface com o utilizador e o núcleo gráfico. A única desvantagem desta solução poderia residir no facto de a linguagem Java ser uma linguagem executada por uma máquina virtual e, portanto, poder apresentar um desempenho inferior em relação ao desempenho de código equivalente escrito em C ou C++.

A solução adoptada permite definir o núcleo gráfico como um objecto que realiza transparentemente toda a manipulação gráfica dos conjuntos de dados, disponibilizando para tal uma interface bem definida constituída por métodos e propriedades a que a interface com o utilizador acede para executar os comandos desencadeados pelo utilizador. Cada conjunto de dados é, por sua vez, um objecto cujos métodos são invocados pelo núcleo gráfico.

A plataforma de desenvolvimento adoptada foi um computador pessoal Pentium III a 500 MHz com 128 Mb de memória, operando sob Windows 98. As ferramentas de desenvolvimento adoptadas foram o JDK 1.2.2, Java3D 1.1.3, JavaHelp 1.1 e o ambiente de desenvolvimento Forte for Java CE 1.0.

4. FUNCIONALIDADE IMPLEMENTADA

A aplicação para visualização interactiva NISVAS opera sobre conjuntos de dados referenciados a geometrias. Os dados podem ser dos tipos escalar e vectorial. A referenciação geométrica dos dados segue o modelo dos métodos de simulação numérica por Elementos Finitos[1] que definem formas geométricas bi ou tridimensionais com base em malhas de nós colocados no espaço de simulação. A figura 2 apresenta alguns destes elementos, que podem ser contíguos ou não, conforme a figura 3 apresenta.

Os conjuntos de dados podem ser carregados e eliminados da aplicação, sendo possível copiá-los ou alterar o seu nome. Um conjunto de dados pode ser lido pela aplicação de uma só vez ou carregado ficheiro a ficheiro, podendo-se adicionar um novo ficheiro ao conjunto de dados a qualquer momento ou removê-lo do conjunto.

Quando um conjunto de dados é carregado, a aplicação apresenta a sua geometria e selecciona uma iluminação ambiente uniforme. Através da interface, o utilizador pode então alterar a cor dos objectos ou atribuí-la segundo funções de transferência que fazem corresponder a gama de valores de uma grandeza escalar a uma escala de cores.

O utilizador pode também seleccionar grandezas escalares e vectoriais do conjunto de dados, fazendo-as corresponder à cor e tamanho de símbolos ou à cor, intensidade e direcção de vectores referenciados à geometria. A geometria dos objectos pode ainda ser deformada designando uma grandeza vectorial como a deformação dos objectos.

[1] Isto não impede a visualização de resultados de simulações realizadas por outros métodos, como os métodos de Diferenças Finitas.

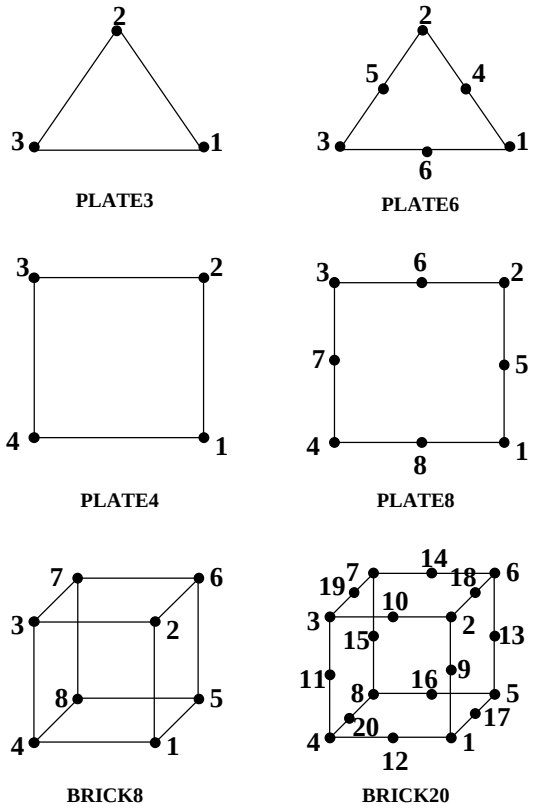


Figura 2 – Alguns elementos de malha empregues pela aplicação NISVAS, mostrando a localização e numeração de nós.

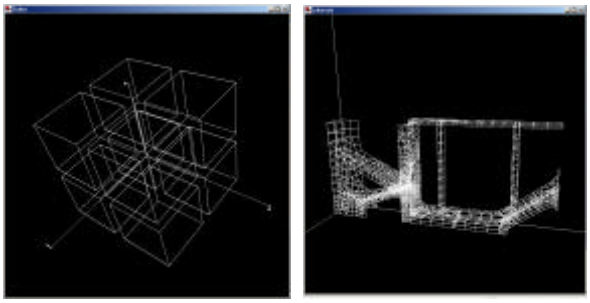


Figura 3 –Geometria de dois conjuntos de dados visualizados pela aplicação NISVAS.

A aplicação NISVAS permite a manipulação da iluminação, a criação de transparências com grau de transparência controlável e planos de corte (para análise do interior dos objectos) que é possível localizar precisamente.

Toda a informação sobre a interacção realizada com um conjunto de dados é guardada coerentemente, sendo possível armazenar externamente uma vista sobre um conjunto de dados. O carregamento posterior de uma vista implica o carregamento automático do respectivo conjunto de dados e a sua apresentação imediata com o mesmo mapeamento de grandezas, iluminação e vistas que possuía quando foi criada.

4.1 Behaviour

A aplicação NISVAS recorre ao conceito de Behaviour do Java3D. Este conceito consiste na definição da uma conexão entre um estímulo (evento) e uma acção. O visualizador NISVAS emprega não só Behaviours pré definidas em Java3D, como implementa novas Behaviours para realizar as operações de rotação, panning e zoom, quer através da interacção por meio de dispositivos apontadores como o rato, quer através do teclado.

4.2 Mapeamento de Grandezas nos Objectos

O visualizador NISVAS permite fazer corresponder grandezas escalares e vectoriais à cor dos objectos, símbolos e vectores e controlar a cor dos objectos pré definidos das cenas, como luzes e fundos, especificando a cor por meio dos modelos RGB e HSV, conforme a preferência do utilizador. No caso de grandezas escalares, a correspondência é estabelecida entre a gama de valores da grandeza escalar e uma escala de cores que pode ser de quatro tipos: Physics (escala iniciada no azul e terminada no vermelho, idêntica ao espectro de cor), Hueramp (do vermelho ao vermelho, passando por todas as cores saturadas do modelo HSV), Grayscale (do preto ao branco, passando pelos cinzentos) e Hotiron (do vermelho ao branco, passando por laranja e amarelo).

A cor dos objectos pode igualmente ser definida por uma grandeza vectorial cujas componentes são as componentes RGB da cor a atribuir a cada um dos nós da malha.

A geometria dos objectos pode ser alterada ou deformada, bastando para isto adicionar às coordenadas dos nós da malha os valores de uma grandeza vectorial cujas componentes são interpretadas como os deslocamentos a aplicar aos respectivos nós nas três direcções do espaço. Estes deslocamentos podem ser ampliados ou reduzidos por meio de um factor de escala que é possível controlar interactivamente, podendo ainda ser animados se os deslocamentos forem identificados como pertencentes a momentos diferentes da simulação.

A visualização da geometria dos objectos definidos pelos conjuntos de dados, assim como dos objectos globais de cena (eixos coordenados, caixa envolvente, etiquetas, luzes e fundo de cena), é controlada pelo utilizador, podendo o utilizador escolher torná-los invisíveis para melhor observar outros objectos como símbolos e vectores. Normalmente, a geometria dos objectos é representada por polígonos orientados correspondendo às faces dos elementos da malha. As suas arestas podem também ser visualizadas por opção do utilizador.

4.3 Símbolos e Vectores

A aplicação NISVAS permite fazer corresponder símbolos e vectores a grandezas associadas aos nós da malha, tal como a figura 4 apresenta. Os símbolos podem ser bidimensionais (círculos, triângulos ou quadrados) ou tridimensionais (esferas, octaedros ou cubos).

Na representação de grandezas vectoriais por meio de vectores, estes podem ser de vários tipos como linhas simples, pirâmides facetadas, pirâmides de faces arredondadas ou cilindros com topos cónicos.

Aos símbolos e vectores é possível atribuir cores tal como para os objectos da geometria. A dimensão dos símbolos

e vectores pode reflectir grandezas escalares cujo factor de escala pode ser controlado pelo utilizador.

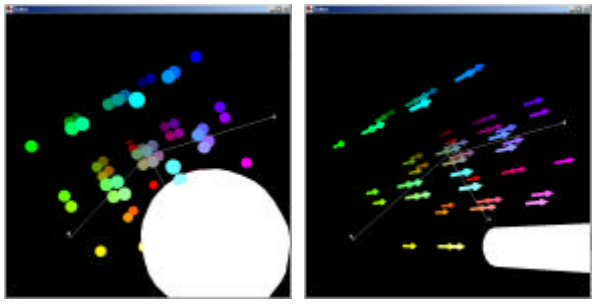


Figura 4 – Representação de uma grandeza escalar por meio de símbolos e de uma grandeza vectorial por meio de vectores (a geometria foi ocultada para facilitar a visualização).

4.4 Objectos Globais

O visualizador NISVAS define um conjunto de objectos globais para as cenas. Estes objectos (eixos coordenados, caixa envolvente da geometria, etiquetas, cor de fundo, panos de corte e luzes) podem ser tornados visíveis ou invisíveis, ou, no caso de caixas envolventes, serem representados por meio de pontos, linhas e polígonos, consoante a preferência e a necessidade dos utilizadores.

4.5 Luzes

Para além da luz ambiente cuja intensidade e cor podem ser controladas, o visualizador NISVAS permite ainda iluminar as cenas por meio de outras luzes colocadas em locais pré definidos. Estas luzes podem ser direccionais, pontuais ou do tipo foco. A cor e intensidade destas luzes são igualmente controláveis pelo utilizador.

O visualizador permite também controlar as propriedades cromáticas das superfícies dos objectos, como a sua difusibilidade e opacidade.

4.6 Opções de Visualização

Além da funcionalidade que permite definir e localizar vários planos de corte para exame do interior dos objectos, o visualizador NISVAS apresenta ainda funcionalidade de controlar a forma de apresentação das imagens na janela de visualização, permitindo visualizar só as faces anteriores ou só as faces posteriores, ver a cena em todo o ecrã e ligar ou desligar a iluminação.

Por omissão, o visualizador NISVAS apresenta imagens correspondentes à projecção de perspectiva, mas o utilizador pode optar pela projecção paralela ou por vistas seleccionadas (frontal, planta ou alçado) em separado ou simultaneamente.

5. INTERFACE COM O UTILIZADOR

Para concretizar os objectivos anteriormente definidos e disponibilizar toda a funcionalidade descrita, a interface do visualizador NISVAS foi cuidadosamente concebida e realizada. A metodologia seguida consistiu na análise das tarefas a realizar pelo utilizador e na avaliação de protótipos, incluindo protótipos de baixa fidelidade, que permitiram escolher e validar soluções.

O objectivo pretendido era o de maximizar o emprego da manipulação directa dos objectos e proporcionar aos utilizadores uma interface coerente e de aprendizagem rápida, em que os diálogos com o utilizador não distassem mais do que dois passos a partir da janela principal de diálogo, nem apresentassem simultaneamente mais do que cerca de seis opções [Mayhew92].

Pretendeu-se também proporcionar redundância das formas de acesso aos diálogos para que a interface se adaptasse tanto a utilizadores inexperientes ou pouco frequentes, como a utilizadores muito experientes na aplicação. Além disso, a interface deveria proporcionar ao utilizador uma visão clara do estado da aplicação que não o sobrecarregasse com demasiada informação, isto é, que para além do estado global da aplicação, os detalhes dos estado da aplicação fossem apresentados de acordo com o contexto dos diálogos seleccionados.

Finalmente, a interface deveria facilitar a identificação das opções de comando por parte do utilizador.

Para atingir estes objectivos, a interface do visualizador NISVAS apresenta dois tipos de janelas: a janela principal, a partir da qual o utilizador dialoga com a aplicação e introduz comandos, e as janelas de visualização, uma por cada conjunto de dados. A figura 5 apresenta o aspecto de um ecrã onde se vê a janela principal do visualizador e duas janelas de visualização correspondendo a dois conjuntos de dados que estão a ser visualizados.

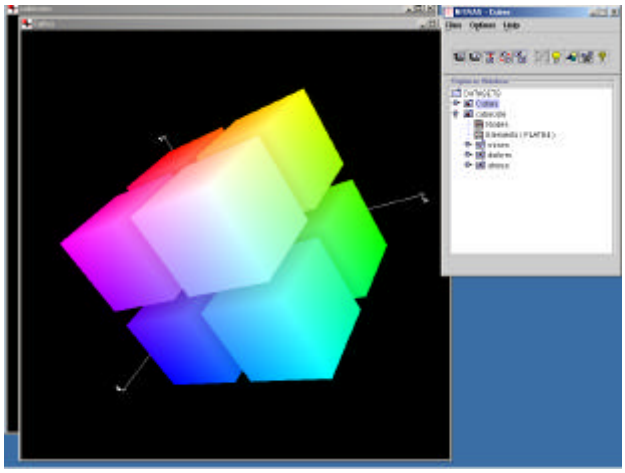


Figura 5 – Ecrã mostrando a janela principal do visualizador NISVAS e a visualização de dois conjuntos de dados, cada um na respectiva janela de visualização.

5.1 Manipulação Directa na Janela de Visualização

Ao carregar um conjunto de dados, a aplicação NISVAS instancia imediatamente a janela de visualização do conjunto onde, também imediatamente, é apresentada a geometria correspondente ao conjunto de dados carregado[2].

Os objectos presentes numa janela de visualização podem ser manipulados directamente deslocando o rato e premindo simultaneamente um dos seus botões. As opera-

[2] As restantes grandezas serão posteriormente mapeadas na geometria através dos diálogos disponíveis a partir da janela principal.

ções assim realizáveis são rotações dos objectos (botão esquerdo premido), zoom (premindo o botão central) e panning (botão direito). Como se trata de manipulação directa, as operações realizadas podem ser imediatamente desfeitas por deslocamento do rato na direcção oposta.

Todas as operações de rotação, zoom e panning realizadas numa janela de visualização podem ser desfeitas com um a única acção. Esta consiste em premir e largar o botão esquerdo do rato sem o mover.

5.2 Estado Geral da Aplicação

O estado geral da aplicação é apresentado ao utilizador através das janelas de visualização e da janela principal.

As janelas de visualização permitem que o utilizador avalie de forma imediata o estado da aplicação no que diz respeito à transformação de vista aplicada aos objectos de cada conjunto de dados e, em menor grau, do mapeamento das grandezas do conjunto, do controlo de luzes e dos objectos globais da cena.

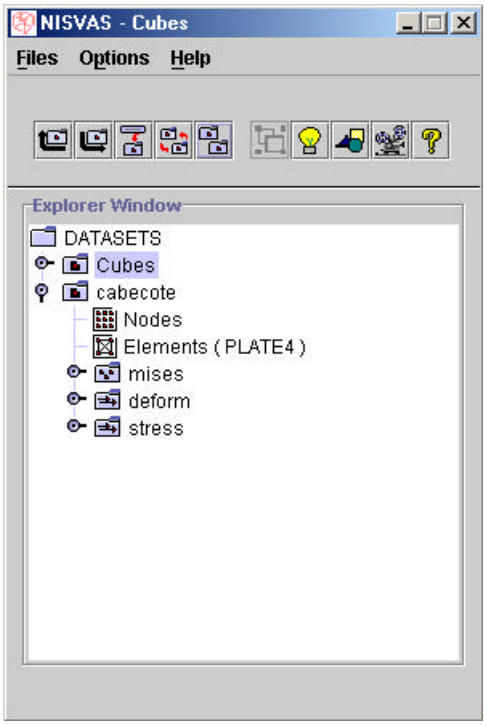


Figura 6 – Janela principal da aplicação NISVAS com barra de menu (topo), barra de ferramentas (no meio) e zona de exploração (na parte inferior) com dois conjuntos de dados.

A janela principal (veja-se a figura 6) encontra-se dividida em três zonas: zona da barra do menu principal, zona da barra de ferramentas e zona de exploração. Esta última zona apresenta o estado da aplicação referente aos conjuntos de dados correntemente carregados.

O conjunto de dados correntemente activo é assinalado pela aplicação NISVAS na zona de exploração, rodeando o seu nome com uma caixa com cor de fundo invertida. Para seleccionar um conjunto de dados como o conjunto activo basta clicar sobre o seu nome na zona de exploração.

Por omissão, cada conjunto de dados é apresentado de forma expandida na zona de exploração, isto é, os nomes de todos os ficheiros contendo as grandezas do conjunto de dados são apresentados, tal como a figura 6 o faz para o conjunto *cabecote*. Esta apresentação repete-se para as grandezas que sejam definidas em mais do que um instante de tempo (mises, deform e stress, neste exemplo). É possível ocultar esta lista de nomes, como acontece como o conjunto *Cubes* na mesma figura, clicando no símbolo – localizado à esquerda do nome do conjunto. O símbolo passa então a + e a lista pode ser de novo apresentada, voltando a clicar sobre ele.

5.3 Adaptação ao Utilizador e Redundância

A redundância na emissão de comandos ou na selecção de diálogos é essencial para a adaptação de qualquer aplicação interactiva às preferências, grau de perícia dos utilizadores e contexto das tarefas que pretendem realizar[3].

Na aplicação NISVAS, a redundância permite entre 4 a 5 formas distintas para aceder aos comandos da interface. A selecção de comandos pode ser realizada por meio de

- Menu principal
- Barra de ferramentas
- Menus PopUp na zona de exploração
- Teclas de atalho
- Teclas aceleradoras

A operação de eliminação de um conjunto de dados, por exemplo, pode ser realizada através do menu principal (Files->Unload Dataset), da barra de ferramentas (segundo ícone a contar da esquerda), por menu PopUp (clicando o botão direito do rato sobre o nome do conjunto na zona de exploração e seleccionando Unload Dataset), por teclas de atalho de navegação no menu principal (Alt+F, U) ou por teclas aceleradoras (CTRL-U).

O utilizador inexperiente pode assim realizar operações passo a passo através do menu principal, enquanto um utilizador experiente poderá realizar as mesmas operações através da barra de ferramentas ou de teclas aceleradoras.

5.4 Acesso aos Diálogos

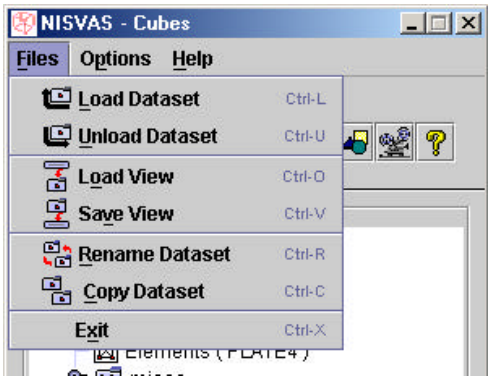
Como foi dito anteriormente, pretendeu-se proporcionar o acesso aos diálogos de comando através de um mínimo de acções de selecção, com o intuito de evitar que os utilizadores tivessem que construir modelos mentais complexos e difíceis de aprender, reduzindo assim também o número de erros na selecção.

O estudo de análise de tarefas, reflectido na estrutura do menu principal, permitiu a adopção de uma estrutura hierárquica com apenas dois níveis de profundidade e um número reduzido de opções em cada nível. Este esforço de sistematização foi particularmente impulsionado pela

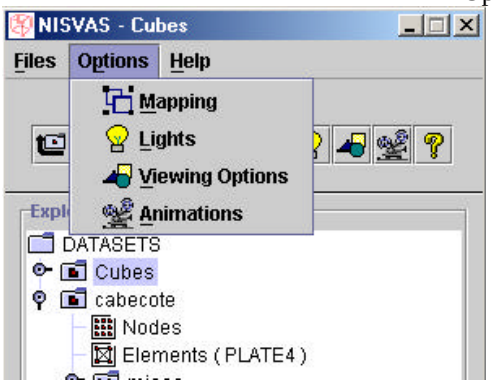
[3] Evitando, por exemplo, os incómodos devidos à necessidade de alternar entre o uso do teclado e do rato.

solução adoptada para o diálogo de mapeamento que apresentaremos em detalhe mais à frente.

Files



Options



Help

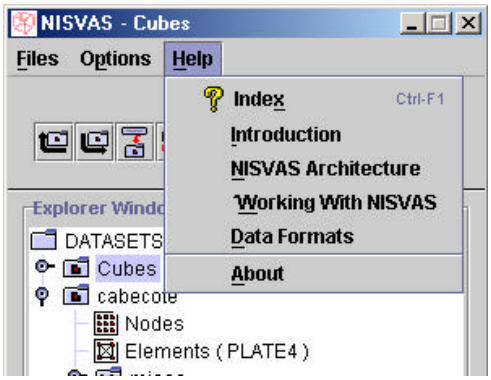


Figura 7- Opções disponíveis nos grupos do menu principal.

Assim, a hierarquização das tarefas a realizar permitiu dividi-las em três categorias:

- Files – carregamento e eliminação de conjuntos de dados, carregamento e armazenamento de vistas e saída da aplicação.
- Options – mapeamento de grandezas e controlo da iluminação, das opções gerais de visualização e das animações.
- Help – sistema de ajuda e documentação, constituída por índice, introdução, arquitectura, como utilizar a aplicação NISVAS e formato dos dados de entrada.

O ícone mais à esquerda na barra de ferramentas também proporciona o acesso directo à documentação.

A figura 7 apresenta as opções disponíveis a partir de cada grupo do menu principal, em que é claro o pequeno número de opções de cada grupo.

A análise de tarefas do grupo mais complexo, o grupo Options, permitiu separá-las em quatro diálogos distintos: Mapping, Lights, Viewing Options, e Animations, apresentados pelas figuras 8 , 9 , 10 e 11 , respectivamente.

Figura 8 – Diálogo de mapeamento (Mapping).

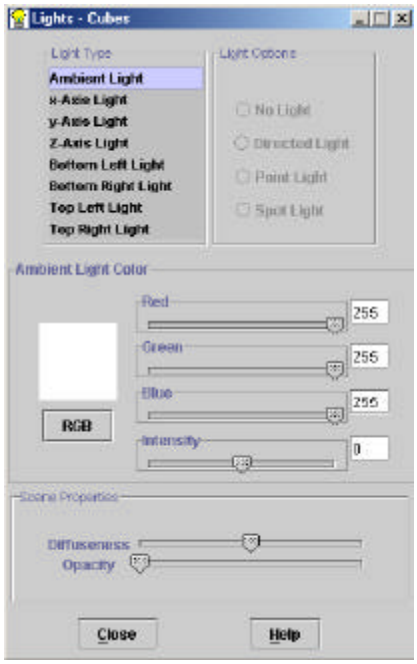


Figura 9 – Diálogo de controlo da iluminação (Lights).

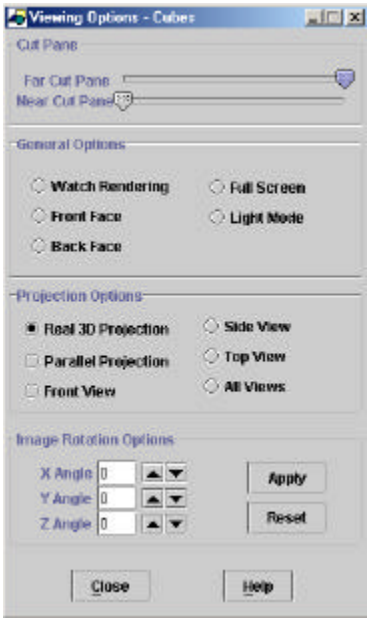


Figura 10 – Diálogo das opções gerais de visualização (Viewing Options).

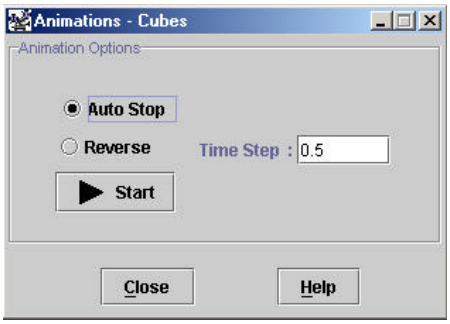


Figura 11 – Diálogo de controlo de animações (Animations).

À excepção do diálogo de mapeamento, a análise de tarefas permitiu simplificar consideravelmente o conteúdo de todos os diálogos, tal como é patente das figuras. Em todos os diálogos existe o botão Close para encerramento dos diálogos, à esquerda e em baixo, e, à direita, o botão Help que proporciona o acesso à ajuda contextualizada referente ao diálogo corrente.

5.5 Diálogo de Mapeamento

O diálogo de mapeamento é talvez o diálogo mais importante de toda a interface do visualizador NISVAS pois é a partir deste diálogo que o utilizador selecciona o mapeamento das grandezas dos conjuntos de dados em cores, símbolos, vectores e deslocamentos e especifica as respectivas propriedades. Este diálogo permite controlar também o aspecto e apresentação dos objectos globais pré definidos (caixa envolvente da geometria, rótulos e etiquetas, eixos coordenados e cor de fundo).

A análise das tarefas de mapeamento permitiu identificar que estas tarefas incidem sobre quatro grupos de objectos: geometria, símbolos, vectores e objectos globais. Existem algumas semelhanças entre as operações que é possível realizar sobre estes quatro grupos de objectos, mas existem igualmente bastantes diferenças.

A concepção deste diálogo deveria permitir um diálogo muito claro para o utilizador. Assim, adoptou-se por uma apresentação do tipo Tabbed Panel, disponibilizada pelo API da linguagem Java, em que as opções se encontram separadas por secções identificadas através de etiquetas separadoras e em que todas as subsecções da secção corrente são visíveis em simultâneo. Deste modo, evita-se a necessidade de o utilizador ter que memorizar a organização hierárquica deste diálogo. Esta característica é reforçada pelo facto de o número de subsecções de cada secção (4, no máximo) ser muito reduzido.

A tabela 1 apresenta as divisões em secções e subsecções resultante da análise de tarefas. As figuras 12 a 15 apresentam diálogos de mapeamento para cada uma das secções.

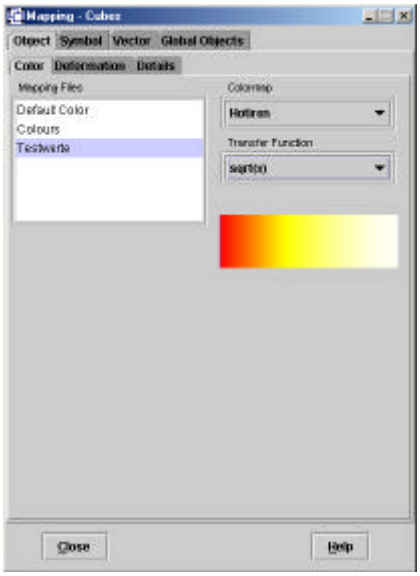


Figura 12 - Diálogo de mapeamento na geometria dos objectos.

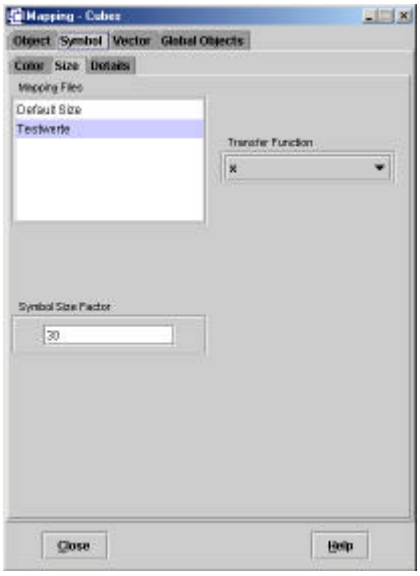


Figura 13 – Diálogo de mapeamento de grandezas em símbolos.

Grupo	Object	Symbol	Vector	Global Objects
Subsecção	Color	Color	Color	Bounding Box
	Deformation	Size	Length	Labels
	Details	Details	Details	Coordinate Axis
				Background Color

Tabela 1 – Secções e subsecções do diálogo de mapeamento.

los.

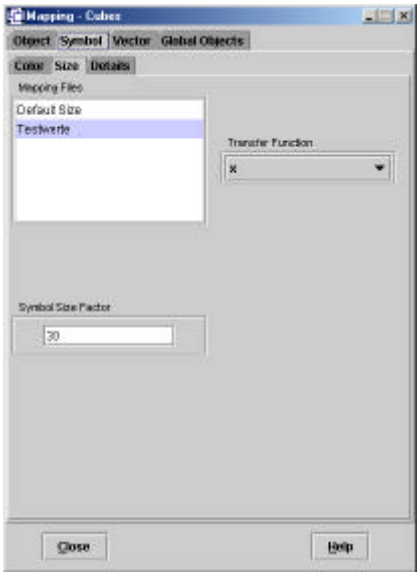


Figura 14 – Diálogo de mapeamento de grandezas em vectores.

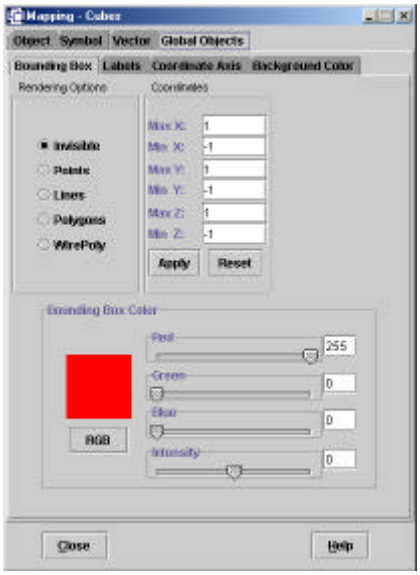


Figura 15 – Diálogo de mapeamento e controlo dos objectos globais.

6. EXEMPLO DE APLICAÇÂO

Como exemplo, apresentamos de seguida o caso do estudo da deformação de uma peça de protecção de carruagens ferroviárias contra choques. A geometria desta peça é constituída por 1452 nós a que correspondem 1545 elementos planares do tipo PLATE4 (veja-se a figura 2). Além da geometria, o conjunto de dados contém a informação vectorial da deformada da peça e do estado de tensão, conjuntamente com o valor de teste (escalar) para o critério de von Mises, em todos os nós da malha. Estes dados estão disponíveis para 21 momentos do processo de deformação da peça, o que torna este conjunto de dados bastante complexo.

A figura 16 apresenta a deformação da peça, obtida pelo mapeamento dos valores do campo da sua deformada como deformação (ampliada) da geometria, seguida pela animação. A peça é representada por meio de linhas (wireframe) para melhor visualização dos resultados.

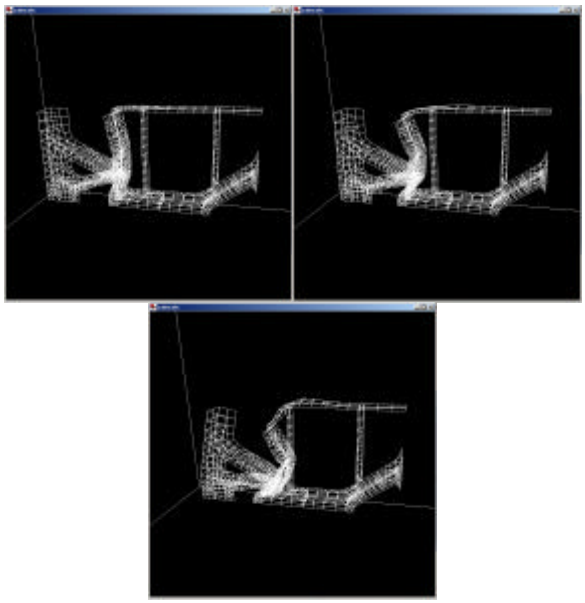


Figura 16 – Deformação de uma peça sob impacto violento em três momentos do processo.

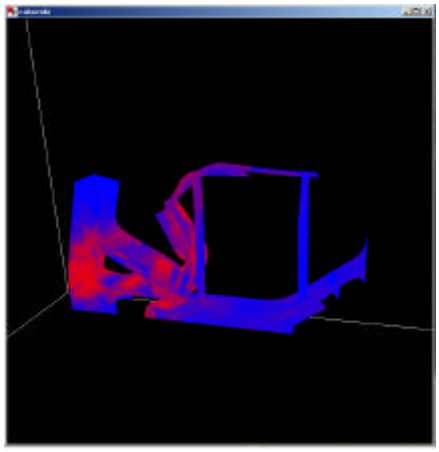


Figura 17 – Apresentação da deformação de uma peça simultaneamente com o módulo do estado de tensão local representado por cor.

A figura 17 apresenta a peça num dos momentos da deformação, permitindo apreciar simultaneamente a deformação e o estado de tensão local, traduzido em cor atribuída segundo uma escala do tipo Physics.

7. CONCLUSÕES

O desenvolvimento de raiz da aplicação NISVAS para a visualização de dados de grande volume resultantes da simulação de fenómenos complexos bi e tridimensionais constituiu um projecto complexo que requereu cuidados extremos na concepção da arquitectura, na selecção das ferramentas de desenvolvimento e, principalmente, na concepção e realização da interface ao utilizador.

Com efeito, o trabalho realizado enfrentou vários desafios importantes como o desenvolvimento segundo conceitos e paradigmas actuais, nomeadamente a Concepção Orientada para Objectos, e o emprego de tecnologias emergentes, como é o caso da tecnologia Java3D. É conveniente realçar que, durante a realização deste projecto, esta tecnologia foi alvo de constantes actualizações, razão pela qual houve que estar atento à publicação de novas versões e, por vezes, reescrever parte do código fonte para poder usufruir da novas funcionalidades entretanto disponibilizadas ou eliminar código escrito para ultrapassar erros (bugs) presentes em versões anteriores das ferramentas de Java e Java3D.

Mas foi ao nível da concepção e realização da interface com o utilizador que se colocaram os maiores desafios. Com efeito, o sucesso ou insucesso de uma aplicação interactiva para visualização, como a aplicação NISVAS, depende muito mais da interface com o utilizador e da facilidade com que o utilizador a emprega, do que da funcionalidade da aplicação. A metodologia de trabalho empregue, baseada na análise de tarefas e na construção e avaliação de protótipos, foi determinante para os excelentes resultados obtidos.

Destes resultados, é de salientar o nível de simplificação da hierarquia de comandos, conseguido sem diminuir a funcionalidades pretendida da aplicação, e que permite ao utilizador ter uma visão permanente das opções ao seu dispor simultaneamente com a possibilidade de seleccionar comandos pela forma que melhor se adapte ao seu perfil de utilizador (adaptação e redundância). Estas características da interface permitem que a utilização da aplicação NISVAS seja simultaneamente simples, poderosa e de fácil aprendizagem, não requerendo a construção de modelos mentais complexos da aplicação por parte do utilizador.

As reacções positivas registadas junto dos utilizadores confirmaram que os objectivos delineados no início deste trabalho tinham sido efectivamente alcançados. Os utilizadores realçaram as potencialidades disponibilizadas pela aplicação NISVAS, a facilidade de uso da sua interface, além do bom desempeno da aplicação. Algumas das reacções recolhidas salientaram que, à partida, nunca teriam esperado que uma aplicação tão complexa e desenvolvida em Java3D pudesse apresentar as características e nível de desempenho verificados.

7.1 Trabalho Futuro

As reacções dos utilizadores e avaliadores nos testes e demonstrações realizadas constituem um encorajamento para continuar o desenvolvimento da aplicação de visualização NISVAS. Com efeito, a arquitectura e as soluções encontradas constituem uma excelente base de trabalho para futuros desenvolvimentos.

Um destes desenvolvimentos incidirá no suporte a outros formatos para os dados de entrada que não o formato actual. Os utilizadores apontaram a conveniência em permitir a entrada de dados em formatos normalizados, como formatos neutros ou outros, como o formato HDF. Foi também identificada a necessidade de poder comunicar directamente dados à aplicação NIVAS a partir das simulações (visualização on-line).

O trabalho futuro incidirá igualmente no aumento de funcionalidade de animação de conjuntos de dados, com o mapeamento temporal de mais do que uma grandeza e na animação de sistemas de partículas para visualização de escoamentos de fluidos, estacionários ou transitórios, como, por exemplo, escoamentos atmosféricos, escoamentos em rios, estuários e em alto mar, com transporte de partículas e poluentes ou túneis de vento virtuais.

Um outro desenvolvimento programado consistirá na manipulação directa das grandezas constituintes dos conjuntos de dados para, através da metáfora da calculadora de bolso, combinar, por exemplo, grandezas escalares em grandezas vectoriais, escalar grandezas ou aplicar-lhe transformações.

8. REFERÊNCIAS

[Arnold96] K. Arnold, J. Gosling, The Java Programming Language, Addison-Wesley, 1996.

[Bouvier99] D. J. Bouvier, Getting Started with Java3D API, Sun MicroSystems Inc., 1999.

[Caiado01] B. Caiado e L. Correia, Visualização de Dados de Grande Volume, Trabalho Final de Curso, Licenciatura em Engenharia Informática, Instituto Superior Técnico, 2001.

[Haase95] H. Haase e outros, ISVAS 3.2 Interactive System for Visual Analysis User's Guide, Fraunhofer Institute for Computer Graphics, 1995.

[Karlsson93] K. Karlsson, ISVAS 3.1 Interactive System for Visual Analysis User's Guide, Fraunhofer Institute for Computer Graphics, 1993.

[Karlsson94] K. Karlsson, Ein Interaktives System zur Visuellen Analyse von Simulationsergebnissen, PhD diss., THD, Darmstadt, 1994.

[Mayhew92] D. J. Mayhew, Principles and Guidelines in Software User Interface Design, Prentice Hall, 1992.

[Preece98] J. Preece, Human Computer Interaction, Addison-Wesley, 1994.

[Shneiderman98] B. Shneiderman, Designing the User Interface, Addison-Wesley Longman, 1998.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Alisamento e dizimação de malhas triangulares

Roberto Lam
Escola Superior de Tecnologia

Robert Loke
Hans du Buf
Faculdade Ciências e Tecnologia

Universidade do Algarve
{rlam,loke,dubuf}@ualg.pt

Sumário
Este artigo apresenta uma solução do problema levantado pelo enorme número de triângulos associados à visualização de modelos tridimensionais. Apresentamos um método de alisamento com preservação da forma e tamanho do objecto. O alisamento laplaciano de objectos elimina as arestas e conduz a uma contracção das formas convexas. A utilização do alisamento auto-adaptável, no qual cada vértice é deslocado em função da curvatura local à vizinhança permite utilizar o método de forma iterativa e preservar as arestas, importantes características dos objectos. Após a operação de alisamento é possível eliminar os triângulos das áreas planas tornando as malhas triangulares mais apropriadas para aplicações de visualização interactivas.

Palavras-chave
3D, visualização volumétrica, triangulação, alisamento, dizimação triangular.

1. INTRODUÇÃO

Em 1997 a universidade do Algarve foi admitida como cooperante do projecto ISACS: Integrated System for Analysis and Characterization of the Seafloor[Berntsen97]. A visualização dos resultados obtidos pela segmentação dos dados acústicos em 3D é efectuada com recurso a VRML [Nikolov97]. Existe uma inúmera variedade de áreas onde são produzidos ou adquiridos modelos tridimensionais. Actualmente existem dispositivos que permitem a digitalização de objectos tridimensionais. A utilização de ultra-sonografia, da tomografia axial computadorizada, da imagem por ressonância magnética, da fotografia por satélite, do CAD, etc., facilmente gera modelos com milhares de polígonos. Aplicação simples do algoritmo *Marching Cubes* (MC) [LorCli87] e a sua visualização sob o modelo de iluminação Gouraud é muitas vezes frustrante por três motivos: i) as fronteiras ao nível dos *voxels* podem possuir ruído, não só devido à segmentação mas também pela discretização, ii) o algoritmo poderá gerar centenas de milhares de triângulos e iii) um sistema interactivo com iluminação Gouraud e uma simples rotação só poderá ser possível em sofisticadas e dispendiosas estações gráficas. Estas razões justificam o objectivo de optimização da malha triangular. Por exemplo, um cubo perfeito pode ser descrito por milhares de triângulos, contudo essa representação poderá ser reduzida a 12 triângulos, ou 6 quadrados. O principal objectivo deste trabalho é melhorar e acelerar os *interfaces* utilizados. Isso implica reduzir o número de triângulos da malha triangular resultante do processo de segmentação. Para atingir o objectivo proposto desenvolvemos um algoritmo que permita o alisamento da malha poligonal resultante da aplicação do algoritmo MC, ao volume segmentado. Para teste do algoritmo desenvolvido iremos aplica-lo em volumes sintéticos e reais, nomeadamente provenientes do projecto ISACS, de ressonância magnética para diagnósticos médicos, etc. Existem levantamentos efectuados sobre este assunto, Elvins apresenta um resumo dos métodos utilizados em visualização [Elvin92], Heckbert e Erikson apresentam um vasto conjunto de técnicas de optimização, simplificação e redução de malhas triangulares [Heck97], [Erikson96].

2. VISUALIZAÇÃO VOLUMÉTRICA

Os algoritmos de visualização volumétrica podem ser classificados em dois grupos: DRV-visualização volumétrica directa e SF-extracção de iso-superfícies [Elvin92]. Os métodos DVR são caracterizados pela simulação do processo real de visualização. Basicamente estes métodos utilizam a projecção de raios para mapear os elementos a visualizar no écran sem utilização de primitivas geométricas. Apresentam a desvantagem de ser necessário repetir o processo sempre que se altere o ângulo de visão mas como vantagem apresentam a possibilidade de acelerar o processo pela utilização de uma baixa resolução. Os métodos SF efectuam a extracção da iso-superfície, no caso dos campos escalares, pelo ajuste de primitivas geométricas, triângulos ou polígonos, no conjunto de dados. A extracção da iso-superfície é definida pela selecção de um valor de referência e o algoritmo percorre o conjunto de dados verificando quais são os *voxels* que pertencem à superfície. Os *voxels* cujos cantos possuam valores acima ou abaixo do valor de referência estão fora da superfície. Uma vez que estes métodos utilizam primitivas geométricas podem ser construídas listas de polígonos bem como dos seus vértices. Estas listas permitem uma rápida visualização após mudança do ângulo de visão, pois só é ne-

cessário efectuar os cálculos referentes à mudança do referido ângulo. Os métodos SF são de um modo geral mais rápidos e com melhor interactividade do que os DVR e nele se classifica o algoritmo MC que passamos a descrever de um modo sumário.

2.1 Marching Cubes

O algoritmo MC é relativamentemente simples. O *voxel* utilizado pelo método é formado pelo cubo volumétrico definido à volta do ponto amostrado. Os oito valores existentes nos cantos do *voxel* são obtidos por interpolação dos pontos vizinhos e são utilizados para determinar a existência, ou não, da superfície no seu interior. Se esses valores forem tais que alguns deles estejam acima do valor de referência e outros abaixo, então a superfície passa pelo *voxel*. De modo a implementar o algoritmo, Lorensen e Cline numeraram os cantos, de um a oito, e para armazenar o resultado da análise ao *voxel* utilizam um *byte*. Se o canto possui um valor superior ao de referência, a posição do canto no *byte* é colocado a "1" caso contrário a "0," ver Fig. 1.

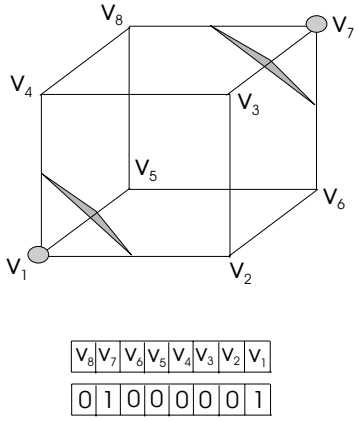


Figura 1: Exemplo de um *voxel* com numeração dos cantos e respectivas representações num *byte*.

Os *bytes* são posteriormente processados para se determinar a localização da configuração triangular correspondente. Num *voxel* podem existir 256 configurações possíveis de malhas constituídas por quatro ou menos triângulos, no entanto este número pode ser reduzido a 15 por meio de rotações e simetrias, ver Fig. 2. Durante a fase de aquisição da malha triangular do *voxel* são calculados os vectores normais aos triângulos bem como os vectores normais aos vértices, possibilitando assim a aplicação do modelo de iluminação Gouraud [Watt93]. Este algoritmo apresenta uma ambiguidade na definição da malha triangular entre dois *voxels* contíguos. Esta ambiguidade possibilita a produção de superfícies topologicamente incorrectas. Chernyaev apresentou um estudo extendendo o número de configurações básicas de 15 para 33 [Chernyaev95] . Desde a sua apresentação têm sido construídos algoritmos derivados do MC. De modo a diminuir o enorme número de triângulos gerados pelo algoritmo, os autores do MC apresentaram um método para redução da malha triangular, o *Dividing Cubes* [Clin88]. Basicamente, o que este algoritmo faz é sempre que encontra um *voxel* que pertença à superfície, efectua a projecção da sua malha triangular no plano de visualização. Se a área a visualizar for inferior a um pixel será visualizada como um ponto da superfície.

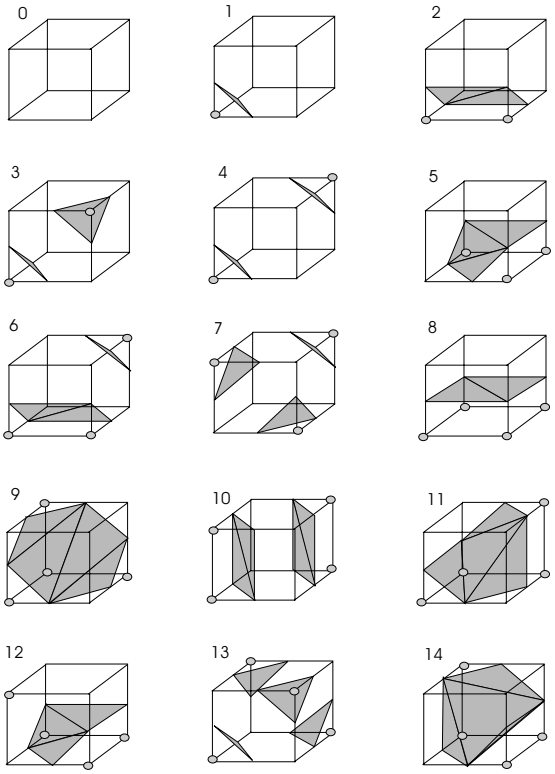


Figura 2: Configurações básicas do *Marching Cubes*. Os cantos assinalados são interiores ao objecto.

Caso contrário o *voxel* deverá ser subdividido e processado de novo. Devido ao facto do volume de dados só possuir uma pequena percentagem de *voxels* que pertencem à superfície[Wilh92], Shekhar et al.[Shek96] apresentaram uma melhoria. Considerando o facto das superfícies serem contínuas, o algoritmo proposto tem em consideração que se um determinado *voxel* pertence à superfície então existem *voxels* contíguos que também pertencem à superfície, logo a progressão do algoritmo efectua-se sobre a superfície. Para evitar o excesso de triângulos este algoritmo aplica o *voxel* do MC a agrupamentos de 2x2x2, 4x4x4 ou de ainda maiores dimensões nas regiões de baixas frequências, gerando assim triângulos grandes nas regiões planas e pequenos nas irregulares. Montani et al, apresentaram *Discrete Marching Cubes*[Mon94], onde em vez de interpolação, utilizam o ponto médio entre dois cantos do *voxel* como ponto de cruzamento da superfície com a aresta. Para além de 12 vértices, considerados pelo MC, Montani et al. consideram o centróide do *voxel* como vértice da malha triangular local. A redução obtida na irregularidade da malha triangular global permite, numa fase posterior, fundir os triângulos em polígonos co-planares maiores.

2.2 Dizimação de Triângulos

Este algoritmo foi apresentado por um dos autores de MC [LorWi92]. O seu objectivo principal é a redução do número de triângulos numa fase de pós-processamento. Este algoritmo é constituído por três passos: primeiro, caracte-

rização da geometria local ao vértice, segundo, avaliação do critério de dizimação e por último a re-triangulação resultante da eliminação do vértice. A caracterização do vértice consiste em classificá-lo numa das seguintes categorias: simples, complexo, de fronteira, de aresta e canto. O vértice simples é caracterizado pelo facto dos vizinhos não pertencerem à fronteira e a superfície por eles definida não possuir arestas ou cantos. Se na superfície existirem arestas ou cantos então o vértice é classificado como de aresta ou canto. Os vértices complexos definem a divisão de uma superfície em duas, e por último, os vértices de fronteira pertencem à fronteira, ver Fig. 3.

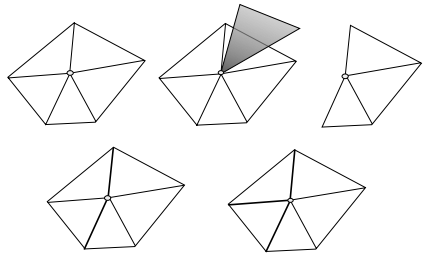


Figura 3:Tipo de vértices, da esquerda para a direita, cima: simples, complexo e fronteira, baixo: aresta e canto.

O passo seguinte é a avaliação do critério de dizimação que consiste em avaliar se o vértice é candidato à eliminação. O critério utilizado é função do tipo de vértice. Se o vértice é simples, o critério utiliza a distância do vértice ao plano médio para decidir pela eliminação, ou não, do vértice. No caso dos vértices de fronteira ou de aresta, é utilizada a distância entre o vértice e a linha que une os dois vértices que definem a fronteira ou aresta, ver Fig. 4.

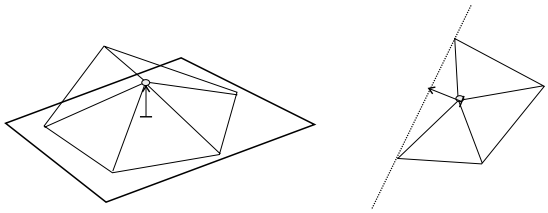


Figura 4: Critério de dizimação, da esquerda para a direita, plano e fronteira.

Schroeder et al. chamam a atenção para o facto de ser possível que a existência de ruído na malha crie situações em que vértices de aresta ou de canto não sejam eliminados. Para estes casos sugerem a utilização do critério da distância ao plano médio associado a um parâmetro definido pelo utilizador. A última fase do algoritmo consiste em efectuar a triangulação do buraco originado pela eliminação do vértice. Para isso o algoritmo utiliza um ciclo recursivo de sub-triangulação. Basicamente o polígono a re-triangular é dividido em duas metades e a cada uma delas é aplicado de novo o método de triangulação. Shroeder apresentou a utilização deste algoritmo num algoritmo[Schroeder97] destinado à dizimação, transmissão e reconstrução de malhas triangulares.

3. ALISAMENTO E REDUÇÃO DE MALHAS TRIANGULARES

Nesta secção apresentamos um algoritmo ``cosmético" que pretende ultrapassar os problemas mencionados antes. De modo a eliminar o ruído proveniente da segmentação e discretização, é aplicado de um modo auto-adaptável, o alisamento laplaciano à malha triangular. Posteriormente, os triângulos existentes nas regiões planares, ou quasí-planares, poderão ser eliminados e o polígono resultante re-triangulado com menor número de triângulos. O alisamento laplaciano consiste em substituir cada um dos vértices pelo centróide dos vértices vizinhos. O centróide poderá ser calculado com ou sem o vértice em processo. Em qualquer dos casos o ruído poderá ser eliminado por uma filtragem iterativa. O problema é a criação de arestas arredondadas. Por exemplo, um cubo é distorcido para uma forma elipsóide e a aplicação repetida do filtro irá finalmente transformá-lo num ponto, ver Fig.5. Vollmer et al. apresentaram um método [VolMen99] que força o vértice a retornar à posição inicial em função dos deslocamentos dos vértices vizinhos.

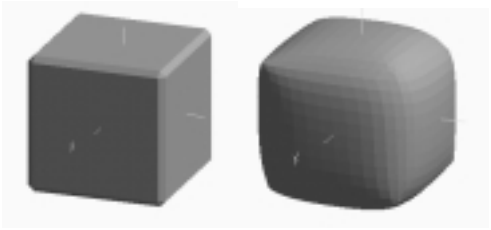


Figura 5: Da esquerda para a direita: cubo original, alisado com 10 iterações do filtro Laplaciano.

O método apresentado na subsecção 3.1 é baseado nas técnicas de alisamento com preservação de bordas utilizadas no processamento de imagem [duBuf90]. Este utiliza um parâmetro de controlo do deslocamento do vértice, que é função da curvatura local, evitando assim a contracção do objecto e preservando as arestas.

3.1 Alisamento

Para este trabalho utilizaremos a representação proposta por [Hoppe94]. Seja V a lista de vértices, T a lista de triângulos e V_i as coordenadas de cada vértice definidas da forma $V_i = (x_i, y_i, z_i)$. Cada vértice V_i possui vértices vizinhos e para os objectivos deste trabalho definiremos dois tipos de vizinhança: S_1 que possuirá todos os vértices que distem de V_i uma aresta e S_2 com todos os vértices que distem de V_i uma e duas arestas. O centróide C_i é calculado sobre S_1, do seguinte modo:

$$C_i = \frac{1}{N}\sum_{j=1}^{N} V_j \quad ; \quad j \in S_1$$

sendo N o número dos vértices conectados a V_i. O cálculo do centróide poderá incluir o vértice em processamento V_i (j=i) ou não (j ≠ i). O alisamento laplaciano consiste em trocar as coordenadas de V_i pelas do seu centróide C_i. Isso é efectuado através da criação de uma nova lista de vértices em cada iteração, ou, por restrições de memória, na utilização da mesma lista com distintos modos de pro-

cessamento nas iterações. No caso do alisamento com preservação da topologia iremos utilizar um parâmetro associado à curvatura da vizinhança local a S_2. Este parâmetro, designado por α, é definido como função da variância das distâncias entre os vértices $V_j \in S_2$ e o plano médio a S_2. O plano é obtido através da soma dos vectores normais aos triângulos pertencentes à vizinhança, ver Fig. 6. Uma vez determinado o plano é necessário movê-lo para o centróide da vizinhança. Isto é efectuado de modo a garantir que a variância reflicta a curvatura local. De modo a melhor preservar os cantos e arestas durante o processo de alisamento iremos utilizar a maior vizinhança S_2 e não S_1.

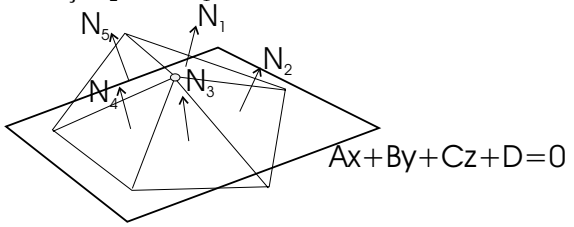


Figura 6:Vectores normais da vizinhaça S_1 e respectivo plano médio.

A equação do plano médio $Ax+By+Cz+D=0$ será calculada com os coeficientes A, B e C:

$$A=\frac{1}{N}\sum_{j=1}^{n} n_j^x, \quad B=\frac{1}{N}\sum_{j=1}^{n} n_j^y, \quad C=\frac{1}{N}\sum_{j=1}^{n} n_j^z$$
$$com \quad j \in S_2$$

O coeficiente D é calculado de modo a que o plano passe pelo centróide. A variância é calculada de acordo com:

$$\sigma^2=\frac{1}{N}\sum_{j=1}^{n}\left(dist_j-\mu\right)^2 \quad com \quad j\in S_2 \quad e \quad \mu=0$$

e reflecte a curvatura local da superfície. Finalmente o deslocamento do vértice V_j para o centróide é controlado pelo parâmetro α que é função da variância, sendo o deslocamento:

$$V_i^{\text{depois}}=(1-\alpha)V_i^{\text{antes}}+\alpha C_i$$

com

$$\alpha=\begin{cases} 1 & se \quad 0{,}0\le\sigma^2\le 1{,}0 \\ -\frac{2}{3}\sigma^2+\frac{5}{3} & se \quad 1{,}0\le\sigma^2\le 2{,}5 \\ 0 & noutros \quad casos \end{cases}$$

Assim, quando a variância é pequena o vértice desloca-se para o centróide. Quando a variância é grande, o vértice permanece na posição. Os valores limites 1,0 e 2,5 foram determinados experimentalmente em malhas triangulares provenientes da aplicação do MC em volumes sintéticos.

3.2 Medidas de Convergência

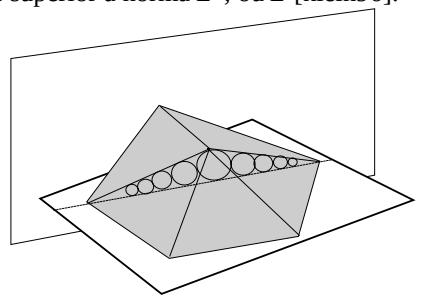
Por definição dois conjuntos estão a uma distância *r* se e só se qualquer ponto de um dos conjuntos está no máximo a uma distância *r* de qualquer ponto do outro conjunto. Formalmente podemos definir a distância de Hausdorff [Barn88] com base no conceito matemático de vizinhança. Seja X um espaço métrico, d a distância euclidiana, $A \subset X$ e r > 0, então a vizinhança aberta de A com raio r será:

$$V_r(A)=\{y : d(x,y)<r \qquad \forall x\in A\}$$

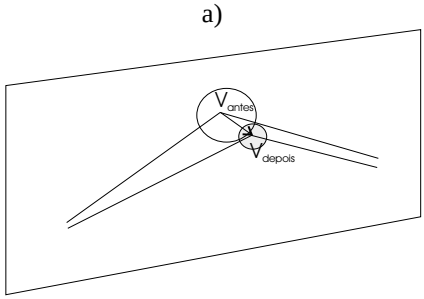
A distância de Hausdorff poderá ser definida em termos desta vizinhança. Sejam A e B dois subconjuntos de X, a distância Hausdorff entre eles será:

$$h(A,B)=\text{Inf}\{r: \quad A\subset V_r(B) \quad \wedge \quad B\subset V_r(A)\}$$

Para controlo da evolução do processo de alisamento como critério de paragem, utilizamos a distância de Hausdorff, h, entre duas malhas consecutivas, ver Fig.7. No processo de alisamento não ocorre eliminação de vértices sendo portanto fácil de implementar uma medida de erro baseada na distância euclidiana entre V_{antes} e V_{depois}. Contudo, essa medida não será eficientemente descritiva do erro verificado, pois como se pode observar na ver Fig. 7b), os círculos em redor de V_{antes} e V_{depois} representam h. Klein et al. demonstraram com alguns exemplos que a utilização da distância de Hausdorff como medida de erro, é superior à norma L^∞, ou L^2[klein96].



a)



b)

Figura 7: a)As esferas representam as distâncias de Hausdorff entre o plano e as arestas dos triângulos, b)detalhe transversal do deslocamento do vértice V_i.

Para superfícies fechadas podemos também calcular o volume do sólido a visualizar [Rourke93]. No caso do objecto a visualizar não possuir uma superfície fechada, o cálculo será relativo ao volume compreendido entre a superfície e a origem do referencial utilizado. Assim, é possível calcular o volume compreendido entre duas malhas sucessivas e conjuntamente com a distância de Hausdorff, seguir as alterações resultantes do alisamento da malha.

3.3 Redução

Após a aplicação da suavização são extraídos da malha os vértices em áreas co-planares. Para a implementação do método de eliminação dos vértices, utilizou-se o conceito matemático de plano associado a um coeficiente de rela-

xação. Sendo utilizado como coeficiente de relaxação a variância das distâncias perpendiculares ao plano dos vértices de S_1. Os valores de relaxação utilizados foram $0{,}5x10^{-3}$, $0{,}5x10^{-5}$ e $0{,}5x10^{-10}$. O método é aplicado sobre a vizinhança S_1 do vértice em processamento V_i. De modo a ser possível a utilização de triangulação em duas dimensões utiliza-se uma mudança de referencial, exemplificada na Fig. 8.

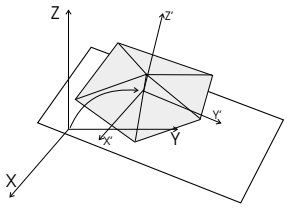


Figura 8: Mudança de referencial. O novo eixo OZ' coincide com o vector normal à vizinhança S_1.

A mudança de referencial é efectuada assumindo que as malhas são provenientes de dados colhidos, ou produzidos, em coordenadas cartesianas e utilizando uma taxa de amostragem superior à de Nyquist. Com este pressuposto é possível aplicar-se qualquer algoritmo de triangulação ao polígono originado pela projecção da vizinhança S_1 no plano. Para a triangulação foi utilizado o algoritmo de Seidel implementado em [Manocha94]. Este algoritmo decompõem o polígono em *n-2* triângulos, sendo *n* o numero de vértices do polígono.

3.4 Implementação

Para a implementação dos algoritmos descritos anteriormente utilizamos três listas dinâmicas para o armazenamento da malha original. Na primeira fase, a lista possui os vértices gerados pelo algoritmo MC. Os triângulos são constituídos por três vértices consecutivos: os primeiros três vértices pertencem ao primeiro triângulo, os segundos três vértices ao segundo triângulo e assim sucessivamente até ao final da lista. O facto de dois triângulos adjacentes possuírem um ou dois vértices comuns implica a existência de repetições de um determinado vértice na lista. Na segunda fase é criada uma segunda lista com os triângulos, sendo estes descritos pelos índices dos vértices que os compõem. Nesta fase são eliminados os vértices repetidos. Na terceira e última fase é criada uma terceira lista idêntica à primeira mas possuindo cada vértice ponteiros para os vértices que simultaneamente façam parte da sua vizinhança S_1, ver Fig. 9. A eliminação de vértices nas áreas co-planares é efectuada através da eliminação dos ponteiros que ligam o vértice em processamento aos seus vizinhos, e inserção de ponteiros nos vértices vizinhos, de acordo com a re-triangulação efectuada, como pode ser visto na Fig. 10. A implementação das rotinas foi efectuada em ANSI C e o processo de visualização é efectuado com OpenGL (Open Graphics Library) e GLUT (Graphics Library Utility Toolkit). Para interacção com o utilizador foi desenvolvido um interface em ambiente de janelas com a possibilidade de aplicar a rotação e ampliação (ou redução) dos objectos, e utilização dos modelos de iluminação *flat* e Gouraud.

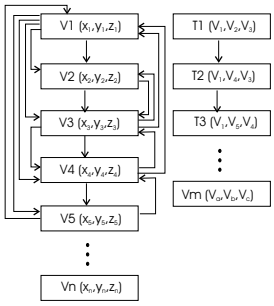


Figura 9: Lista final de vértices

Devido à utilização de código aberto a aplicação poderá ser executada em sistemas SGI ou PC, quer com Linux quer com Windows. A implementação do código das rotinas de alisamento e redução apresentou alguns problemas não previstos, nomeadamente a contracção das fronteiras das superfícies e o surgimento de buracos após sucessiva aplicação do processo de redução.

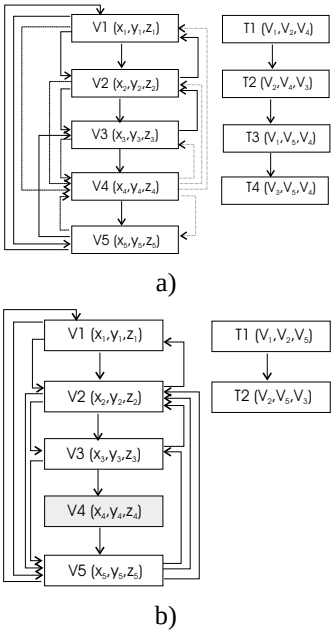


Figura 10: a) Malha triangular com vértice V4 a eliminar, b) malha sem o vértice V4 a a re-triangulação.

Relativamente à contracção das fronteiras, a solução encontrada foi a aplicação do método de alisamento só aos vértices vizinhos que pertençam à fronteira.

4. RESULTADOS

Para a realização de testes utilizámos um conjunto de dados constituídos por dados sintéticos e reais. Os resultados obtidos num octante de uma esfera são mostrados na Fig. 11.

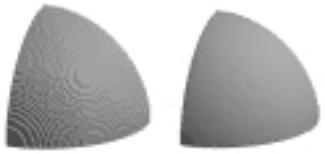


Figura 11: Alisamento auto-adaptável. Octante de esfera visualizado com *flat shading*, da esquerda para a direita: MC sem alisamento e com 6 iterações do filtro.

Foi utilizado o modelo de iluminação *flat shading* para demonstrar a eficiência do algoritmo, pois a utilização do modelo Gouraud efectuaria o alisamento do objecto na visualização. Como pode ser observado, o ruído proveniente da discretização foi completamente eliminado após seis iterações e as arestas foram preservadas. Na Fig. 12, é apresentado o octante de esfera rodado, de modo a mostrar a preservação das arestas perpendiculares e a comparação com o alisamento laplaciano.

Figura 12: Octante de esfera, da esquerda para a direita: filtrado com 6 iterações laplacianas e com 6 iterações do nosso filtro.

Os resultados relativos à redução da malha triangular são relativamente bons, conforme se pode observar na Tab. I existe uma redução do número de triângulos nas áreas planas.

Coef. Relaxação	Triângulos	Vértices
Original	8.804	4.404
$0,5x10^{-10}$	6.550	3.277
$0,5x10^{-5}$	6.314	3.161
$0,5x10^{-3}$	5.022	2.512

Tabela I: Redução da malha triangular (relativos a um octante de esfera).

A utilização de um coeficiente de relaxação nulo ou próximo de zero implica que só os vértices das áreas planas sejam eliminados. Com o coeficiente de relaxação $0,5x10^{-10}$ foram eliminados na primeira iteração aproximadamente 25% do número inicial de triângulos. A aplicação iterativa do filtro de alisamento pode ser controlada pela evolução da distância *h*. Nos testes efectuados sobre o conjunto de dados verifica-se que a distância *h* média apresenta uma diminuição pequena a partir da terceira iteração, sendo a filtragem do ruído maioritariamente efectuada nas cinco primeiras iterações, como se pode ver no gráfico da Fig. 13. A redução da malha triangular sem filtragem pró-alisamento constitui uma solução (pós-processamento) para o enorme número de triângulos gerados pelo algoritmo MC, como se pode observar na Tab. II. A utilização de coeficientes de relaxação elevados permite uma maior redução do número de triângulos, como se vê na Tab. III. O aspecto visual dos modelos sintéticos, após alisamento e redução pode ser visto na Fig. 14.

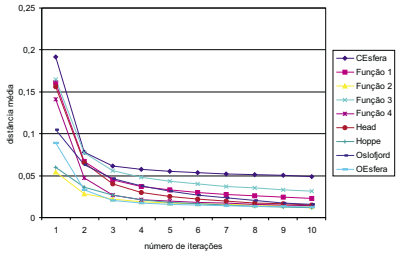


Figura 13: Evolução da distância *h* média.

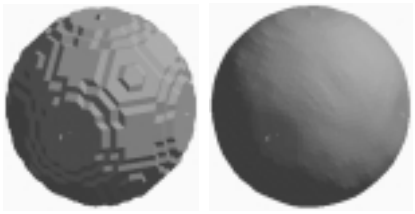
Volume	Original	1ª iteração	2ª iteração
O. Esfera	8.804	4.502	3.730
Esfera	7.528	4.512	4.372
Função 1	4.055	2.587	2.503
Função 2	2.720	446	444
Função 3	4.988	2.986	2.960
Função 4	5.537	3.499	3.474
Hoppe	19.048	7.925	7.087
Oslofjord	21.612	19.834	19.732

Tabela II: Resultado da aplicação de 2 iterações de redução de triângulos a malhas produzidas pelo MC, com coef. relaxação $0,5x10^{-10}$(nº de triângulos)

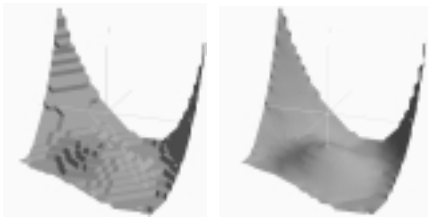
Volume	Original	$0,5x10^{-10}$	$0,5x10^{-3}$
O. Esfera	8.804	6.550	5.022
Esfera	7.528	-	7.052
Função 1	4.055	-	3.611
Função 2	2.720	2.360	1.632
Função 3	4.988	4.954	4.590
Função 4	5.537	5.247	3.621
Hoppe	19.048	15.513	12.279

Tabela III: Resultado da aplicação de 1 iteração de redução de triângulos, após 3 de alisamento (nº de triângulos)

Esfera

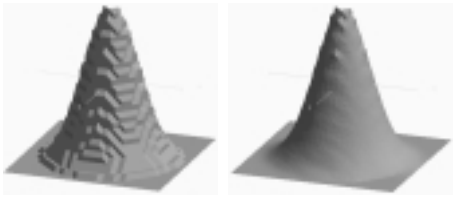


Função 1

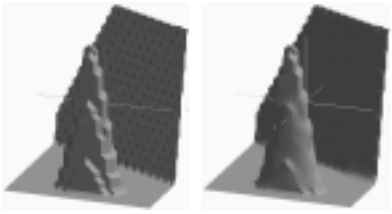


Função 2

Função 3



Função 4



Hoppe

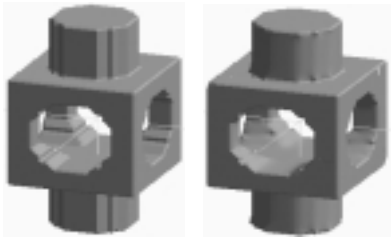
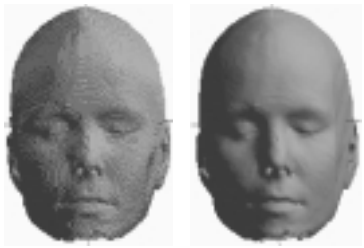


Figura 14: Aspecto visual com *flat shading*, esquerda: superfície original, direita: após 3 iterações de alisamento e uma redução (coef. rel. 0,5x10^{-10}).

A aplicação dos métodos de alisamento e redução em dados reais apresentam valores, quer quantitativos quer qualitativos, semelhantes aos obtidos em dados sínteticos, como se pode observar na Fig. 15 e na Tab. IV.

Head



Oslofjord

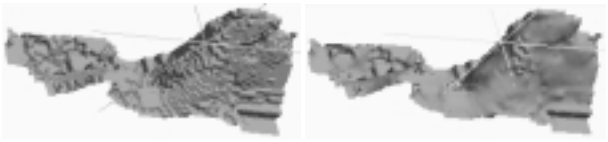


Figura 15: Aspecto visual com *flat shading*, esquerda: superfície original, direita: após 3 iterações de alisamento e uma redução (coef. rel. 0,5x10^{-3}).

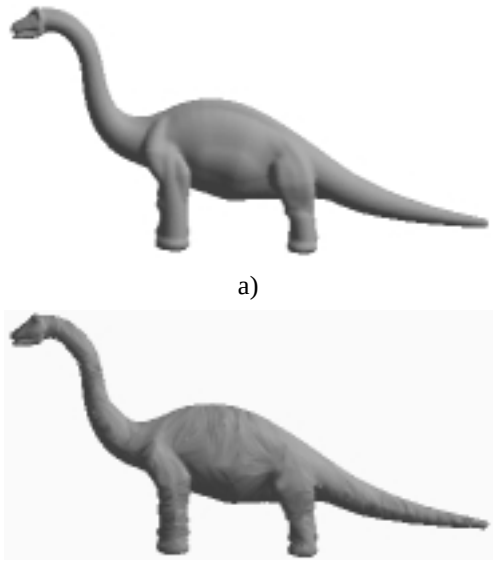
Volume	Original	0,5x10^{-5}	0,5x10^{-3}
Head	159.404	156.878	125.059
Oslofjord	21.213	21.213	20.246

Tabela IV: Resultado da aplicação de 1 iteração de redução de triângulos, após 3 de alisamento (nº de triângulos)

Os dados reais utilizados pertencem, nomeadamente ao conjunto de dados de teste para visualização volumétrica do SoftLab (Software Systems Laboratory), Department of Computer Science, University of North Carolina, USA, e ao projecto ISACS (dados acústicos obtidos pelo sonar Topas na Noruega, em Horten). Na Tab. V pode-se observar os tempos de execução das rotinas de alisamento e redução. Para a redução utilizaram-se as malhas geradas pelo MC com coeficiente de relaxação 0,5x10^{-10} e foram efectuados testes em diversos sistemas, nomeadamente: Linux com processador Pentium III 730 MHz e 768 MB de RAM, Windows 98 com processador Pentium II 350 MHz e 512 MB de RAM, e SGI o200 R10K 180 MHz com 256 MB de RAM, utilizando apenas um processador dos 2 (P III) ou 4 (R10K). A utilização de coeficientes de relaxação altos, igual ou superior a 0,5x10^{-3}, produzem uma distorção na superfície de modelos complexos. Para teste da aplicabilidade do método de redução a superfícies pré-alisadas (modelizadas) foi utilizado uma malha triangular de um dinossauro, apresentando-se os resultados na Fig. 16.

	Linux		Windows 98		SGI o200	
Volume	Alisam.	Reduç.	Alisam.	Reduç.	Alisam.	Reduç.
O.Esfera	0,4417	77,192	0,8517	144,2	0,5767	116,6
Esfera	0,3917	45,963	0,7417	87,4	0,5000	47,8
Função 1	0,2066	11,508	0,3850	21,7	0,2550	18.0
Função 2	0,1350	8,915	0,2667	17,1	0,1733	13.3
Função 3	0,2567	19,563	0,4750	36,9	0,3250	31,5
Função 4	0,2767	21,448	0,5217	40,4	0,3557	35,5
Hoppe	0,9767	213,97	1,7933	403,7	1,2617	721,3
Oslofjord	1,1517	91,47	2,0967	175,4	1,4583	351,0

Tabela V: Tempos de execução médios para alisamento e redução (segundos).



a)

b)

Figura 16: Redução do numero de triângulos, a) original com 41.168 triângulos e b) utilizando o coef. Relaxação 0,5x10^{-3} , 112.636.

5. CONCLUSÕES E FUTUROS TRABALHOS

O método aqui desenvolvido é indicado para a visualização de objectos e superfícies tridimensionais. A sua implementação é relativamente fácil pois as estruturas de dados utilizadas são listas de triângulos e vértices, empregues pela maioria dos visualizadores. A maior vantagem deste método é a melhoria da qualidade das superfícies curvas, com a preservação das arestas. A aplicação do filtro de alisamento numa fase inicial conduz a que as áreas quasí-planas se tornem planas, permitindo assim a eliminação dos triângulos não conectados às arestas. Uma vez que se sabe que um polígono de *N* vértices só poder ser triangulado por *N-2* triângulos, parece que o processo de redução está limitado, mas o problema é que as malhas geradas pelo MC possuem centenas de milhares de triângulos, permitindo assim a redução do seu número e consequente aumento da eficiência e rapidez dos algoritmos de visualização. No entanto, como se pode verificar, a redução de malhas quase completamente curvas será por definição limitada (p.ex. Oslofjord). Mas em geral podemos concluir que os resultados obtidos são melhores em comparação com os apresentados na literatura. No desenvolvimento deste trabalho foram encontrados alguns problemas relativos à implementação do código de redução de triângulos. A sucessiva aplicação da rotina sobre um modelo causa a omissão de triângulos na superfície final. Este facto, para além de penalizar o aspecto do objecto a visualizar, causa problemas no cálculo do volume envolvido, quer em sólidos quer em superfícies abertas. Em termos de duração, a sua execução leva demasiado tempo na localização dos triângulos vizinhos ao vértice em processamento. Note-se que a paralelização dos algoritmos não foi um dos objectivos deste trabalho, mas pode ser considerado no futuro. Um aspecto relevante é o facto do ruído existente nas arestas e extremidades permanecer, ver Função 1 e Hoppe na Fig.14. Outras perspectivas de futuros desenvolvimentos poderão ser: primeiro, alterar a estrutura de armazenamento de lista para arvore octógona, segundo, experimentar com outras medidas da curvatura local e outras funções para o parâmetro α e por último combinar as fases de alisamento e redução numa única fase iterativa.

6. REFERÊNCIAS

[Berntsen97] B. Berntsen, "An Introduction to ISACS," *Proc. 20th Scandinavian Symp. on Physical Acoustics.* Ustaoset, 1997.

[Nikolov97] S.G. Nikolov, R.E. Loke , H.M.H.du Buf, "Interactive 3D visualization of sonar data using VRML," *Proc. Int. Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging.* Rhodes (Greece). Set., pp 36-39., 1997.

[LorCli87] W.E. Lorensen e H.E. Cline, "Marching Cubes: a high resolution 3D surface construction algorithm," *Computer Graphics, SIGGRAPH '87,* vol. 21, pp. 163-169, 1987.

[Elvin92] T. Elvins, "A survey of algorithms for volum visualization," *Computer Graphics,* vol. 26(3), pp. 194-201, Aug. 1992.

[Heck97] S.P. Heckbert e M. Garland, "Survey of polygonal simplification algorithms," *Multiresolution Surface Modeling Course, SIGGRAPH '97*, May 1997.

[Erikson96] C. Ericson, "Polygonal simplification: An overview," TR96-016, Department of Computer Science, UNC-Chapel-Hill, 1996.

[Watt93] A. Watt, "3D Computer Graphics," Addison Wesley, 2ed., 1993.

[Clin88] H.E. Cline, W.E. Lorensen, S. Luke, C. Crawford e B. Teeter, "Two algorithms for three-dimensional reconstruction of tomograms," *Medical Physics*, vol. 15 (3), pp. 320-327, 1998.

[Wilh92] J. Wilhems e A. van Gelder, "Octrees for faster isosurface generation," *ACM Trans. On Graphics,* vol. 11(3), pp. 201-227, July 1992.

[Shek96] R. Shekhar, E. Fayyad, R. Yagel e J.F. Cornhill , "Octree Based Decimation of Marching Cubes Surface," *Proc. On Visualization '96*, pp. 335-342, 1996.

[Mon94] C. Montani, R. Scateni e R. Scopigno, "Discretized marching cubes," *Proc. on Visualization '94*, pp. 281-287, 1994.

[LorWi92] W.J. Schroeder, J. Zarge e W.E. Lorensen, "Decimation of triangle meshes," *Computer Graphics*, vol. 26(2), pp. 65-70, july 1992.

[Schroeder97] W.J. Schroeder, "A topology modifying progressive decimation algorithm," *Proc. On Visualization '97*, pp. 205-212, 1997.

[Amenta97] N. Amenta, M. Bern e D. Eppstein, "Optimal point placement for meshing smoothing," *Proc. 8th ACM-SIAM Symp. On Discrete Algorithms*, pp. 528-537, 1997.

[VolMen99] V. Vollmer et al, "Improved lapalcian smoothing of noisy surface meshes," *Computer Graphics Forum*, vol. 18, pp. 131-138, 1999.

[duBuf90] J.M.H. du Buf e T.G. Campbell, "A quantitative comparison of edge preserving smoothing tecniques," *Signal Processing*, 21:289-301, 1990.

[Barn88] M. Barnsley, "fractals Everywhere," Academic Press 1988.

[Hoppe94] H. Hoppe, "Surface Reconstruction From unorganized Points," Ph.D. Thesis, University of Washington, 1994.

[klein96] R. Klein, G. Liebich e W. Straer, "Mesh reduction with error control," *Proc. IEEE Visualization '96*, pp. 311-318, 1996.

[Rourke93] J. Rouke, "Computational Geometry in C," Cambridge University Press, 4ªed., 1994.

[Manocha94] A. Narkhed e D. Manocha, "Implementation report: fast polygon triangulation algorithm based

on Seidel's algorithm," Dep. of Computer Science, UNC-Chapel Hill, 1994.

[Chernyaev95] E.V. Chernyaev, "Marching Cubes: construction of topologically correct isosurfaces," Relatório Tecnico CN/95-17, CERN, 1995.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Wavelet Compression and Transmission of Deformable Surfaces over Networks

António Calado Lopes
antonio.calado@iscte.pt

Manuel Gamito
mag@iscte.pt

José Miguel Salles Dias
Miguel.Dias@iscte.pt

ADETTI/ISCTE, Associação para o Desenvolvimento das Telecomunicações e Técnicas de Informática, Edifício ISCTE, 1600-082 Lisboa, Portugal, www.adetti.iscte.pt

Abstract

An animation of a deformable surface with fixed topology is formed by a set of connected points (particles), which follow a trajectory over time. The transmission of such an animation is a task that consumes large amounts of bandwidth, as the position of each particle, in each time instant, needs to be transmitted.

We propose a wavelet based compression and streaming mechanism that allows the minimization of the size of the animation data, hence decreasing the total transmission time. The proposed transmission scheme, based on the time-localized feature of the wavelet transform, will be able to stream the animation so that a receiver can immediately view the fraction of the animation received, while wavelet coefficients are still arriving.

Keywords

Geometric compression, streaming, deformable models, wavelet transform

1. INTRODUCTION

In this paper, we present the specification and development of a wavelet compression, transmission and decoding scheme for deformable surfaces[1].

ADETTI has developed over the years a significant know-how in the simulation of deformable surfaces [Dias97,Dias98]. The simulations are physically based and describe the elastic behaviour of surfaces, such as cloth, that evolve over time under the action of external forces and constraints. The surfaces are represented with a triangular mesh, where the vertices of the mesh constitute a dynamic particle system, obeying physical laws. The end result of any simulation is an animation file that stores the positions of the particles for consecutive instants. We have developed a special purpose viewer, using OpenInventor components, to visualize the animation of deformable surfaces.

The next logical step, where visualization is concerned, would be to incorporate the animation data into standard VRML files and be able to access them remotely with any VRML browser. Theoretically, one could use the features of VRML that are already in place, like keyframe interpolation nodes, to represent objects whose shape changes over time. Unfortunately, the data size resulting from the physical simulation of objects is usually quite large given that the objects need a large number of particles to represent them accurately. This would lead to very large VRML files with unacceptable transmission delays through the network. We are aware that the Web3D consortium has a specific working group to discuss a possible compressed binary format for VRML that would be useful in this context [VRML]. However, no definite results have come out from that group yet. In the interim, we propose a streaming mechanism for dynamic data, much like audio or video streaming, where a remote VRML client would display the animation from the stream as it arrives, without having to wait for the end of the transmission. This specification is also applicable to other 3D geometrical formats, such as the AvatarMe .ame proprietary format, or 3D Studio MAX .3ds format [AME].

In the subsequent sections, a small introduction to the various types of deformable models will be presented, followed by an analysis of a complete compression/decompression block scheme for the type of model chosen by us. The notion of wavelet transform will be briefly analysed in order to explain how compression can be obtained. Finally, some of the compression results will be presented, as well as a future implementation of a streaming mechanism, which takes advantage of the time-localized nature of the wavelet transform in order to reconstruct an animation of a deformable model at the same time it is viewed by a receiver.

[1] This work is funded by the European Community IST Project Fashion-Me: “Fashion Shopping with individualized avatars”

2. DEFORMABLE MODELS

Since 1986, the Computer Graphics community as devoted some attention to the problem of modelling deformable objects, such as cloth or flesh, based not only in the geometrical/mathematical intrinsic characteristics of surfaces and volumes, but also in their material proprieties in the context of continuum and discrete mechanics [Feynman86, Weil86]. This approach has been included in a broader modelling framework entitled "Physically-Based Modelling", where rigid or deformable objects or even liquids and gaseous phenomena have been modelled and visualised according to the macroscopic mechanical laws that govern their static or dynamic behaviour [Barr88, Terzopoulos87, Gamito95, Stam95]. Some of the main objectives of this line of activity have been to produce images and computer animation sketches where objects show new "realistic-looking" behaviours. The inherent mathematical/mechanical modelling, include such areas as linear and non-linear dynamics or energy minimisation, whose computational complexity increase with the number of "physically-based" objects in a 3D scene and with the number of degrees of freedom that those objects show.

Interesting applications of this kind of modelling paradigm, for 3D cloth draping and synthetic fashion show simulation, have also appeared, inline with the research carried in this work. Some of the authors have developed pure geometrical methods, while others have adopted a physically-based approach, modelling cloth as a discrete set of coupled particles in interaction towards the minimisation of a potential energy, or using a Newton dynamics equilibrium formulation [Weil86, Hinds90, Feynman86, Breen94 Haumann98]. Several other researchers felt that cloth is best modelled as a continuum, where the classical macroscopic theories of elasticity or differential geometry apply and, in accordance with this idea, have developed methods for determining and visualising static or dynamic equilibrium solutions [Okabe92, Terzopoulos87, Aono90, Li95]. Mixed discrete and continuum models also exist and this has been the approach used in our modelling method: cloth can be assumed as an orthotropic linear elastic continuum, discretised by a mesh of triangles [Volino95, Dias00]. Engineers from the Materials and Textile sciences have also approached the highly non-linear problem of cloth modelling (both in geometric, as well as in elastic material terms), with the Finite Element Method achieving various degrees of success [Loyd80, Zucchini93, Eichen96]. In fact, the Textile community has developed impressive research activity in this field, starting already in 1930 that generally fall in the categories referenced before [Peirce30]. A good survey of these methods and techniques can be found in [Shanahan78, Hearle72].

Generally, we can conclude that all the authors agree in the following: physically-based simulation of deformable objects can be a CPU intensive task, and thus real-time interaction is difficult to achieve without using parallel machines and if proper modelling simplifications and numerical algorithms enhancements are not used [Baraff00, Desbrun00].

The technique reported here, enables the inclusion of deformable objects in 3D virtual environments, by exploiting the time coherence of deformable object motion and developing geometry compression/decompression and transmission methods. With our system, a user is able to interact with the scenario by means of 3D input techniques or a 3D navigation metaphor, where he can recognize, in real-time, realistic-looking (physically-based) behaviors of deformable objects. We have achieved this purpose by decoupling the modeling part of these types of objects, from the visualization/interaction parts.

In the **modeling part**, we've addressed cases of deformable surface draping over rigid body objects and we've defined linear elastic models for the computational simulation of the physically-based surfaces [Terzopoulos87]. For each surface with a known topology, we've associated a coupled particle-system and simulate its behavior by computing the dynamics of the motion of each particle belonging to the close-coupled set of the particle system. So the modeling of such objects, include also the capability to realistically visualize, in real-time, their motion evolution through time. We are able to maintain and store in a file, an history of the simulation of all the particle systems included in the 3D scene, allowing the playback of each deformable object animation at any time, at some future stage.

The simulation results file is then subjected to a coding scheme, streamed using narrow band links and sent to a remote client, where its is decoded for the **visualization/user interaction part.**

This method is quite efficient since it enables the real-time playback of the pre-computed deformable motion behavior of a number of large particle sets. With this technique, there is no need for the time consuming mechanical simulation of those sets, at the time of the deformation visualization, nor is it needed to store, at the client side, the complete history of the particle system deformation.

3. A TRANSMISSION STRATEGY

In order to compress and stream an animation of a deformable surface, a transmission strategy needs to be considered. The following figure shows the complete transmission strategy proposed.

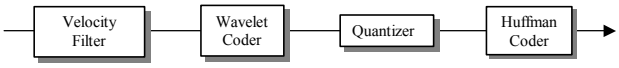


Fig. 1 – The complete encoding scheme

3.1 Pre-filtering the particle trajectories

At the beginning of the transmission scheme, we can choose to transmit any one of the time-dependent variables associated with the particles: the position, the velocity or the acceleration. In the context of a geometric visualization, only the particle positions are of interest. A deformable surface, however, can undergo large changes in its geometry, like a piece of cloth that is dropped over a table. All the position variables will exhibit, as a consequence, a very large dynamic range of values. This causes serious problems during the quantization stage of the transmission and seriously degrades the quality of the animations as seen in the browser.

The velocity is therefore a better candidate for transmission because it mostly oscillates around zero. It is a well-known fact in physics that all systems tend to evolve to minimize their internal energy. A system in such a minimum energy state will be at rest, with all the velocities being zero.

However, an even better candidate would be the acceleration given the characteristics of the systems that we have been simulating. A deformable object can be subject to very large internal stresses that also result in large forces applied at the particles. But all the forces acting at any given particle mainly cancel out each other and only a small resultant force remains. So, not only do the accelerations oscillate around zero, but they also tend to have very small magnitudes. This fact would make the acceleration the best candidate to be transmitted over the network. In what follows, we assume the time increments to be unitary. This is always true after a convenient scaling of the time variable.

The acceleration filter is described by the equation:

$$\mathbf{a}_i(k) = \mathbf{p}_i(k) - 2\mathbf{p}_i(k\text{-}1) + \mathbf{p}_i(k\text{-}2) \qquad (1)$$

where $\mathbf{a}_i(k)$ stands for the acceleration of the particle i in the time instant k and $\mathbf{p}_i(k)$ stands for the position of the particle.

However, when the acceleration filter was implemented, the results achieved weren't exactly what we expected. If we observe the signal power through the entire time span of the animation and compute the average of all trajectories powers (the variance of the signal), we can prove that low magnitude values were obtained.

$$\sigma^2 = \frac{\sum_{i=0}^{N}\left(\sum_{k=0}^{T}(a_i(k))^2\right)}{N} \qquad (2)$$

Equation (2) allows one to calculate the average power of all the accelerations vectors, given N particles and knowing that the time span of each particle ranges from 0 to T. The same formula can be applied to trajectories or velocities. Applying (2) for the acceleration vectors of a sample animation we obtain a value of $1.8\text{x}10^{-5}$.

However the acceleration originated huge oscillations around zero, thus creating spiky functions as the one shown in the next figure:

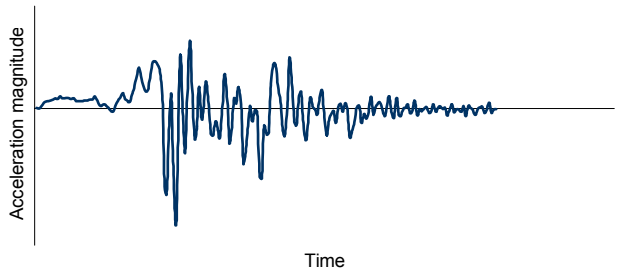


Fig.2 – The acceleration of a sample particle. This plot refers only to the particle's "x" component

This type of function is not the best to be wavelet transformed, for the cubic B-Spline wavelet[2] transform achieves excellent compression ratios with smooth functions due to the wavelet's basis function smooth nature.

Therefore we turned our attentions towards the velocity instead of the acceleration. The velocity is simply given by the first derivative of the particle's position in relation to time ($d\mathbf{p}_i/dt$), and is implemented by the equation:

$$\mathbf{v}_i(k) = \mathbf{p}_i(k) - \mathbf{p}_i(k-1) \qquad (3)$$

where $\mathbf{v}_i(k)$ stands for the velocity of particle i in the time instant k.

When transforming the trajectories in space into sets of velocity vectors we obtain a function with larger magnitude values (using (2) again we obtain $1.2\text{x}10^{-3}$ for σ^2) than the one produced by an acceleration filter, but we gain a lot in function smoothness. This fact can be easily seen in the next figure:

[2] The type of wavelet transform to be used in this work

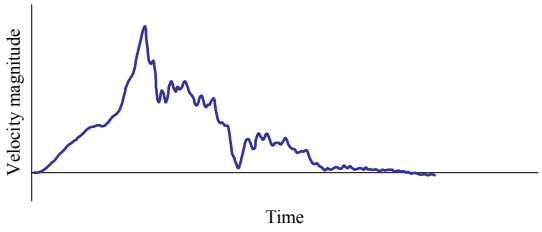


Fig.3 – The velocity of a sample particle. This plot refers only to the particle's "x" component

As our objective is to optimally compress the surface's animation, we need the smoothest function possible with the minimum amplitude range possible. The velocity is the obvious solution and the one chosen.

3.2 From velocity to trajectory

All the blocks presented in figure 1 have their own respective inverse block, which is implemented in the receiver's end. Therefore, to convert velocity vectors into trajectories one needs an inverse velocity filter that is simply implemented by the following equation:

$$\mathbf{p}_i(k) = \mathbf{p}_i(k-1) - \mathbf{v}_i(k) \qquad (4)$$

The decoding sequence must be initialized with the value $\mathbf{p}_i(0)$. The vector $\mathbf{p}_i(0)$ is the starting position of the particle, therefore, for each particle, $\mathbf{p}_i(0)$ needs to be transmitted prior to the transmission of the stream of velocity vectors. It can be included in the first packet, where the information about mesh topology is contained.

3.3 The Wavelet Transform

It is possible to further increase the transmission efficiency, by the application of a suitable transform on the stream of velocity vectors. We seek, in particular, to achieve a lossy transmission scheme, where some information is lost, in a controlled manner, without a significant loss in the end quality of the animation. Basically, this amounts to applying a linear transformation, expressed as a matrix equation that, given the sequence of velocities, returns a sequence of transform coefficients.

For a given particle i and the sequence of vectors $\mathbf{v}_i = [\mathbf{v}_i(0), \mathbf{v}_i(1),\ldots, \mathbf{v}_i(T)]$, we have:

$$\mathbf{T}\boldsymbol{\varphi}_i = \mathbf{v}_i \qquad (5)$$

where $\mathbf{T}$ is the transformation matrix and $\boldsymbol{\varphi}_i$ is the sequence of transform coefficients.

There are many transforms that can be expressed in the form (5) [Jain89]. Fourier or sine transforms could be possible options but they would require the complete set of coefficients to be transmitted before the animation could be correctly reproduced. Wavelets transforms, and the corresponding wavelet coefficients, have a much more localized nature, which enables us to generate portions of the animation on the fly, as new coefficients are being received [Strang97]. Wavelet coefficients fill a hierarchy of levels of detail. They are represented with:

$$\varphi_i^j \qquad \text{with } i = 0, \mathrm{K}\ , 2^j - 1 \qquad (6)$$

where j is the level of detail and i is the coefficient number for level j. The hierarchical structure of the coefficients becomes more evident in the following figure:

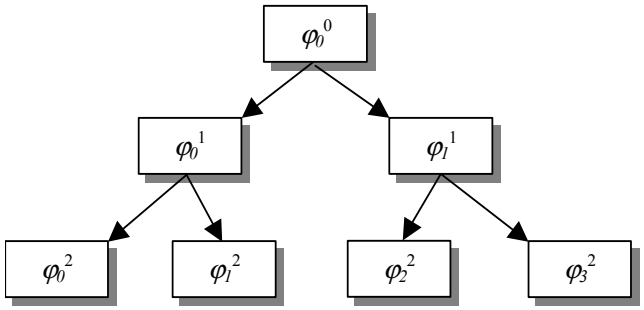


Fig. 4 - The hierarchy of wavelet coefficients.

Within the framework of wavelet transforms, there are still many wavelet bases that can be chosen to express the transformation matrix $\mathbf{T}$ in (5). One should chose a wavelet basis on the interval, in opposition to bases with infinite support, because our signals will only be valid inside some interval $[0,T]$ of time. One should also choose bases with a reasonably high number of vanishing moments. This number reflects the wavelet's ability to correctly reproduce signals. More specifically, a wavelet basis with n vanishing moments can reproduce exactly any polynomial up to degree n. The number of vanishing moments should not get too big, however, because the transform (5) becomes increasingly more expensive to compute. Good candidates for a wavelet base, according to the two previously mentioned criteria, are the n^{th} order B-spline bases on the unit interval [Stollnitz96].

The algorithm that performs the wavelet transform is depicted in the next figure:

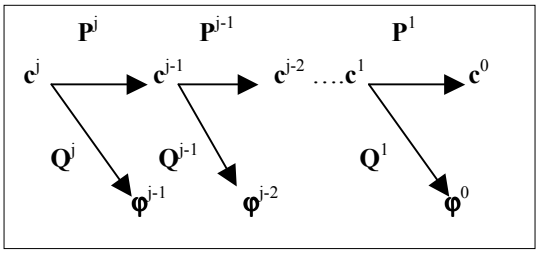


Fig.5 The filter bank

where $\mathbf{P}^j$ is the matrix containing the wavelet scaling function and $\mathbf{Q}^j$ is the matrix which represents the wavelet function itself (the mother wavelet). These two matrices, when concatenated, form the transformation matrix $\mathbf{T}$ in (5). It should be noted that, for each level of detail, there exists a unique transformation matrix, thus the set of all the transformation matrices form a filter bank. A major property of the transformation matrices is that they are mainly composed of zeroes, therefore they are sparse. This fact should be taken on account when solving (5) by using appropriate methods for sparse matrices systems, which can speed up the computation time significantly.

On the other hand, $\mathbf{c}^j$ represents the coefficients of level j, which are to be fed to the beginning of the filter bank. By the action of matrices $\mathbf{P}^j$ and $\mathbf{Q}^j$ the data is then split into a low frequency part and a high frequency part(the wavelet coefficients). Then the high frequency part is stored, and the procedure is repeated to the low frequency part, resulting again in a low and high frequency parts. The procedure is repeated until $\mathbf{c}^0$ and $\boldsymbol{\varphi}^0$ are reached. It should be mentioned that the size of the vector containing the original $\mathbf{c}^j$ must be a value belonging to the set $2^j + m$ where m is the degree of the wavelet transform (in our case, cubic B-Spline wavelets were used).

Using matrix notation, equation (5) takes the following form [Stollnitz96]:

$$\left[\mathbf{P}^j \mid \mathbf{Q}^j\right]\left[\mathbf{c}^{j-1} \mid \boldsymbol{\varphi}^{j-1}\right] = \left[\mathbf{c}^j\right]^T \qquad (7)$$

where our unknowns are $\mathbf{c}^{j-1}$ and $\boldsymbol{\varphi}^{j-1}$ and $[\mathbf{P}^j \mid \mathbf{Q}^j]$ means the concatenation of matrix $\mathbf{P}^j$ with matrix $\mathbf{Q}^j$. The concatenation should be made by interspersing the columns of both matrices in order to avoid zeros in the main diagonal of the resulting matrix. If only one zero is found in the main diagonal of matrix $\mathbf{T}$, the matrix could not be inverted leading to errors when we try to solve the system by iterative methods. By interspersing the columns of both matrices, the result is obviously mixed as well, so a reordenation is required before feeding the $\mathbf{c}^j$ to the next filter.

The method used to solve the linear system was the Bi-Conjugate Gradient Stabilized.

Wavelet compression is obtained according to a criterion of significance:

$$\left|\varphi_i^j\right| < \varepsilon \qquad (8)$$

where ε is a chosen tolerance factor. Any coefficient that obeys (8) will not be transmitted. The receiver will assume it to be equal to zero. The percentage of compression ratio is given by the number of coefficients eliminated in relation to the total number of coefficients.

Given the excellent decorrelation properties of wavelet transforms, together with their inherent locality, one can achieve significant compression ratios with very little degradation of the output signals. The result of a compression using two different tolerance factors can be seen in the next figure:

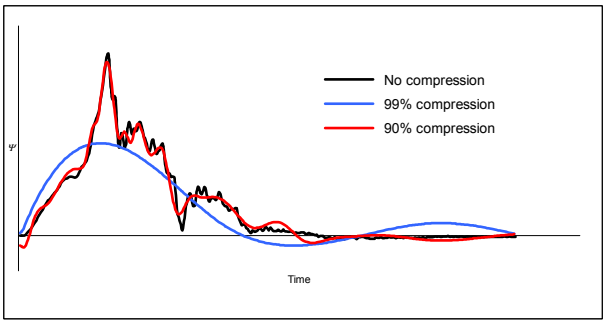


Fig.6 – A sample particle trajectory with no compression, 90% compression and 99% compression ratios.

Following the common practice in transmission schemes, the data is then submitted to a quantizer and then entropy coded using the usual Huffman encoding scheme.

3.4 The Inverse Wavelet Transform

The inverse wavelet transform allows us to recover the original signal given the wavelet coefficients. It's simply implemented by the following equation:

$$\mathbf{c}^j = \mathbf{P}^j \mathbf{c}^{j-1} + \mathbf{Q}^j \boldsymbol{\varphi}^{j-1} \qquad (9)$$

The original coefficients $\mathbf{c}^j$ are obtained by interpolating up $\mathbf{c}^{j-1}$ using the scaling function coefficients represented in the $\mathbf{P}^j$ matrix. The second term of the sum introduces a perturbation on the first term, interpolating up the wavelet coefficients [Stollnitz96]. This interpolation is obtained by the multiplication of the wavelet coefficients by $\mathbf{Q}^j$.

4. RESULTS AND DISCUSSION

By applying the complete procedure to a folding table cloth animation[3], it's possible to discern the differences between different compression ratios.

[3] A circular piece of fabric with 0.36 m of diameter (796 triangles) was placed on top of a cylinder with 0,18 m of diameter. The free surface of the fabric could bend under the gravity field. The simulation produces 400 frames, with 9.36 CPU time (secs)/frame.

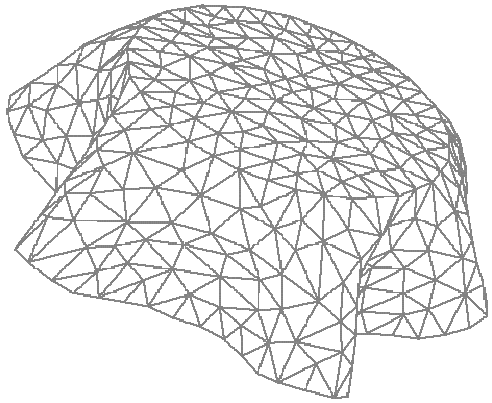


Fig.7 – A table cloth animation with no compression

In figure 7 one can see the last frame of the complete folding cloth animation with no compression. This image will be our base of comparison.

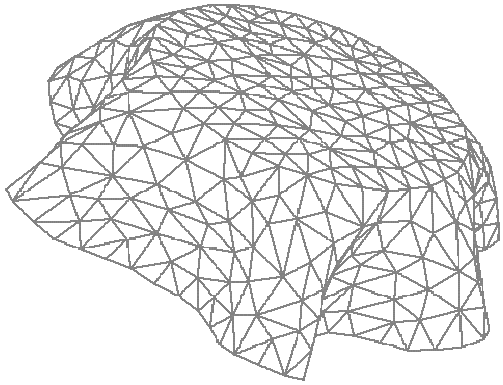


Fig. 8 – The same table cloth with 90% compression

In figure 8 it can be seen that there isn't a significant change in the geometry of the folding cloth although it was represented only with 10% of the original values, hence introducing a 90% compression ratio.

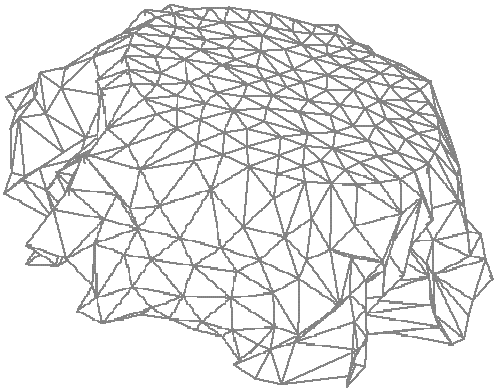


Fig.9 – The same table cloth with 99% compression

In figure 9 an exaggerated compression ratio of 99% was applied. It's evident the change in the geometry, deforming it completely, though an idea of the general animation can still be extracted.

5. FUTURE WORK: A WAVELET-BASED ADAPTIVE DISPLAY STRATEGY

In all the previous sections of this document an encoding scheme for deformable surfaces was considered. In this section it will be discussed a transmission and display strategy to be implemented in the near future.

Given the wavelet hierarchy tree depicted in figure 4, and knowing that each of the coefficients there presented refer to specific time intervals (due to the wavelet's transform time-localized nature), one can think of an effective streaming mechanism for deformable surface's animations.

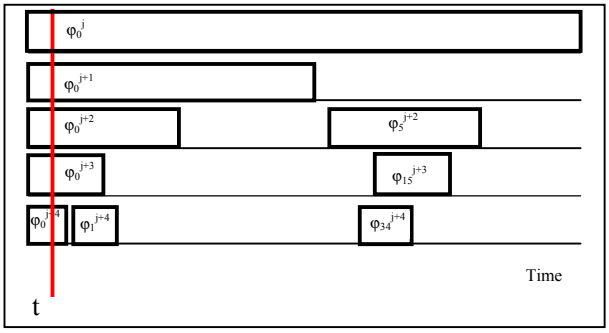


Fig 9. The wavelet coefficients of different levels involved in the representation of time instant t.

The time-localized feature of the wavelet transform is depicted in fig.9, which shows the time span associated with an arbitrary set of coefficients. Coefficients from a lower level of detail occupy a much greater length along the time axis than high level coefficients, which have a much shorter support. It is possible to see from this figure which wavelet coefficients from different levels of detail **j** are involved to represent the animation on time instant **t**. Below is the two scale relation which relates wavelets from different levels:

$$\varphi_i^j(x) := \varphi\left(2^j x - i\right) \qquad i = 0.....,2^j - 1 \qquad (10)$$

Given a specific level **j**, the mother wavelet function is scaled accordingly and translated by **i**.

From (10) we can see that coefficients at higher levels are more numerous but also have much shorter time spans. So, only a limited number of coefficients is involved in the representation of any given animation time sample.

On the other hand, the support that the mother wavelet offers follows equation (11)

$$\text{supp}\ \varphi_i^j(x) := \text{supp}\left(2^{-j}x - i\right) \quad i = 0...,2^j - 1 \quad (11)$$

or if we express (11) in interval notation:

$$\text{supp}\ \varphi_i^j(x) = \\ = \left[i2^{-j}, (m+i)2^{-j}\right] \quad i = 0..,2^j - 1 \quad (12)$$

where m is the degree of the wavelet and taking on account one of the B-Spline wavelet properties [Chui92] which states that:

$$\text{supp}\ \varphi = [0,m] \quad (13)$$

If we transmit the wavelet coefficients by traversing the wavelet hierarchy tree (Fig.4) following a depth-first, left-to-right order, we can generate portions of the animation as soon as the coefficients arrive at the receiver's end. For example, if we wanted to transmit the coefficients that support the first time instant, one should send coefficients φ_0^0, φ_0^1, φ_0^2 by this order.

This procedure can be translated as sending the low frequency components of the trajectories first (those who mostly contribute to the animation) followed by the high frequency components, which introduce detail in the animation.

The smoothness of the playback animation will be dependent on the transmission rate of coefficients. If this transmission rate is lower that the display rate of the time samples, we can choose one of two strategies:

1. Stop the animation and wait for the arrival of the next coefficients
2. Keep playing the animation with the coefficients that are available

The second strategy is possible because the animation will always be smooth no matter what the coefficients that have been received. The objects will display large scale motions, in this case, because only the large scale coefficients will be available. When the small scale coefficients finally arrive, there will be a jump in the animation as the objects are updated to their correct position.

6. CONCLUSIONS

An introduction to deformable models was presented and from all the types analysed, we have chosen a mixed discrete and continuum model applicable for plain woven cloth. Taking into consideration the problem of storing, handling and transmitting over IP networks, pre-computed deformable cloth animation, which produces large data sets, we have developed a geometry compression scheme applicable to this case. Our method requires a trajectory to velocity pre-filtering, concerning cloth particles state, to minimize the amplitude range of the original signal, corresponding to the pre-computed law of motion of the particle system. To further increase the compression rate of the pre-filtered signal, a wavelet based scheme was implemented. The cubic B-Spline wavelet was used, and we have obtained 90% compression rates without no visible degradation of the original signal.

The wavelet based adaptive strategy that was explained earlier is currently under development, therefore no results have yet been obtained. Nevertheless we are optimistic about the future performance of the transmission scheme here presented.

7. REFERENCES

[Aono90] Aono, M., "A Wrinkle Propagation Model for Cloth", CG International´90, Computer Graphics Around the World, Springer-Verlag, Tokyo, pp 95-115, 1990.

[AME] www.avatarme.com

[Baraff00] Baraff. D., Witkin, A., "Rapid Dynamic Simulation", in Cloth Modelling and Animation (edited by Donald House and David Breen), A K Peters, Ltd, pp 145-169, 2000

[Barr88] Barr, A. "Teleological Modelling", ACM SIGGRAPH Tutorial 27 Notes, 1988, B.1-B.6

[Breen94] Breen, D. E., House, D..H., Wozny, M. J., "A Particle-Based Model for Simulating the Draping Behaviour of Woven Cloth", Textile Research Journal, Vol. 64, Nr. 11, pp. 663-685, November 1994

[Chui92] Chui, C., "An Introduction to wavelets", Academic Press Limited, London, 1992

[Desbrun00] Desbrun, M., Meyer, M., Barr, A, "Interactive Animation of Cloth-Like Objects for Virtual Reality", in Cloth Modelling and Animation (edited by Donald House and David Breen), A K Peters, Ltd, pp 219-236, 2000

[Dias97] Dias, J.M.S., Galli, R., Almeida, A.C., Belo, C.A.C, Rebordão, J.M., "mWorld: A Multiuser 3D Virtual Environment", *IEEE Comp. Graphics & Appl*, 17(2), pp. 55-65, March-April, 1997.

[Dias97] Dias, J.M.S., Gamito, M.N., Rebordão, J.M., "Modelling Cloth Buckling and Drape", *Eurographics ´98 Short Presentations*, pp. 2.1.1-2.1.5, September, 1998.

[Dias00] Dias, J. M. S., Gamito, M., N., Rebordão, J. M., "A Discretized Linear Elastic Model for Cloth Buckling and Drape", Textile Research Journal 70 (4), pp 285-297, April 2000

[Eichen 96] Eichen, J, Clapp, T., "Finite-Element Modeling and Control of flexible Fabric Parts", IEEE CG&A, Vol 16-5, pp 71-80 1996.

[Feynman86] Feynman, R. *Modelling the Appearance of Cloth*, Msc Thesis, Department of Electrical Engineering and Computer Science, MIT, Cambridge, 1986

[Gamito95] Gamito, M. N., Lopes, P. F., Gomes, M. R., "Two-dimensional Simulation of Gaseous Phenomena Using Vortex Particles", In D. Thalmann e D. Terzopoulos, ed, Computer Animation and Simulation´95, pp 3-15, Springer, New York, 1995

[Haumann98] Haumann, D. R., Parent, R. E., "The Behavioral Testbed: Obtaining Complex Behaviours from Simple Rules", The Visual Computer, Vol 4, pp. 332-347 (1998).

[Hearle 72] Hearle, J, et al, "On Some General Features of a Computer-Based System for Calculation of the Mechanics of Textile Structures", Textile Research Journal, pp 613-626 (October 1972).

[Hinds90] Hinds, B. K. McCartney, J. "*Interactive Garment Design*", The Visual Computer, Vol 6-2, pp 53-61, March 1990.

[Jain89]Jain, A.K, "Fundamentals of Digital Image Processing", Prentice-Hall, Englewood Cliffs, N.J., ISBN 0133361659, 1989.

[Li95] Li, Y., et al, "Physical Modelling for Animating Cloth Motion", in *Computer Graphics: Development in Virtual Environments* (Proc. CGI) R. Earnshaw and J. Vince, eds, Academic Press, UK, pp 461-474, 1995.

[Loyd80] Loyd, D., "The Analysis of Complex Fabric Deformations", in *Mechanics of Flexible Fibre Assemblies* (edited by Hearle, J., Thwaites, J., Amirbayat, J., NATO ASI Studies), Sijthoff & Noordhoff, Alphen aan den Rijn, Holland, pp 311-342, 1980

[Okabe92] Okabe, H., et al, "Three Dimensional Apparel CAD System", Computer Graphics 26, 2 (Proc. ACM SIGGRAPH 1992), pp 105-110 (July 1992).

[Peirce30]Peirce, F. T. "The Handle of Cloth as a Measurable Quantity", J. Textile Inst., 21 pp 377-416 (1930).

[Shanahan78] Shanahan, W., J., Loyd, D. W., Hearle, J., W., S., "Characterising the Elastic behaviour of Textile Fabric in Complex Deformations", Textile Research Journal, pp 495-505 (September 1978).

[Stam95] Stam, J., Eugene, F. "Depicting Fire and Other Gaseous Phenomena Using Diffusion Processes", University of Toronto, *Computer Graphics Proc., ACM SIGGRAPH 1995*, 129 – 136

[Stollnitz96] Stollnitz, E., Derose, T. & Salesin, D. H. (1996). "Wavelets for Computer Graphics". S.Francisco:Morgan Kaufmann, ISBN 1-55860-375-1

[Terzopoulos87] Terzopoulos, D., Platt, J.,Witkin, A., Fleicher, K. "Elastic Deformable Models", Schlumberger Palo Alto Research, Computer Graphics Proc., ACM SIGGRAPH 1987, 205 – 214

[Strang97] Strang, G., Nguyen, T., "Wavelets and Filter Banks", Wellesley-Cambridge Press, Wellesley, MA., ISBN 0961408871, 1997.

[Volino95] Volino, P., Courchesne, M., Thalmann, N., "Versatile and Efficient Techniques for Simulating Cloth and Other Deformable Objects", Computer Graphics (Proc. ACM SIGGRAPH 1995), pp 99 – 104, 1995.

[VRML] The VRML Compressed Binary Format Working Group,http://www.vrml.org/WorkingGroups/vrml-bf/cbfwg.html

[Weil86] Weil, J., "The Synthesis of Cloth Objects", Computer Graphics (Proc. ACM SIGGRAPH 86), Vol. 20, N. 4, pp. 49-54, July 1986

[Zucchini93] Zucchini, R., Orsi, R., "Finite Element Analysis of Geometrically Non-Linear Shells with Snap-Through", Technical Report RT/INN/93/51, ENEA, Italy, December 1993.

Sessão 4

PROCESSAMENTO DE IMAGEM

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Automatic Feature Extraction on Pages of Antique Books Through a Mathematical Morphology Based Methodology

Isabel Granado Pedro Pina Fernando Muge
CVRM / Centro de Geo-Sistemas, Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa
{igranado,ppina,muge}@alfa.ist.utl.pt

Abstract
This paper presents a mathematical morphology based methodology to identify and extract several components on antique printed books in order to automatically build metadata. These components were previously classified into five different sets (drop capitals, stripes, figures, annotations and text matter) each one characterised by particular geometric features. Based on that assumption several novel algorithms appealing to morphological operators are proposed. The evaluation of the methodology is performed on pages of XVI century books.

Key-words
Digital antique books, mathematical morphology, geometric features, feature extraction

1. INTRODUCTION

Antique books constitute a heritage that must be preserved also with the intention of being available to users. In order to consult these works without damaging and degrading them a suitable media format should be constructed and be put at the disposition of interested people over a suitable system. Presently the tendency points to a digital media support eventually accessed through computer networks.

Although fundamental, the intention of creating digital versions of the documents should be enriched with the possibility of being accessed in a interactive form by users, allowing their consultation about, for instance, the presence or absence of certain specific graphical features or even given keywords. For achieving those objectives it is necessary to firstly identify the components that constitute the pages of the books. The possibility of using automatic procedures may be of great interest once the introduction of objective actions may allow the processing of large amounts of data. Digital image analysis techniques have been used widely with this purpose, mainly over the last decade, in order to automatically process digital versions of documents. Different work has been carried out on this difficult subject [Agam96][He96][Montolio93] that is highly dependent on the objectives and requirements proposed. It must be pointed out that there is not a general methodology that can perform the identification and recognition of the components that constitute the pages of antique books and that these approaches seldom appeal to the geometric nature of the structures present in book pages. The motivation for presenting a novel methodology that exploits the geometric features of the images appears this way. Among the several digital image analysis methods available, in order to deal with the geometric features of images, it emerges mathematical morphology. It is a theory used for image analysis created in the middle 1960's by Georges Matheron and Jean Serra from the *École des Mines de Paris*, with the purpose of quantifying structures according to their geometry. Its theoretical evolution over the years has provided to users a powerful tool not only to deal with the geometry of the structures but also with almost all other chapters in image analysis and in modelling and simulation procedures. To get acquainted with the level of generalisation of this theory and with the details related to the definitions readers are advised to consult the seminal work on the subject [Serra82] and the most recent and updated work [Soille99].

2. DIGITAL IMAGE ACQUISITION AND PAGE SEGMENTATION

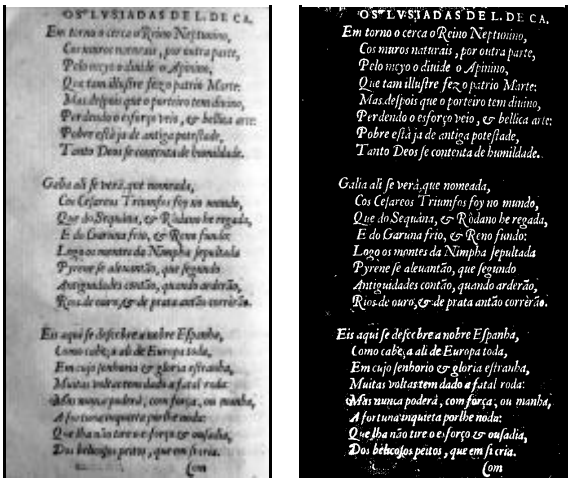
Books in general and antique books in particular must be handled with care to avoid damage. In what concerns digital image acquisition procedures, not only special illumination must be used (for instance, cold lights) but also any contact with glass or any pressure on the bookbinding is forbidden. Once is not allowed to flatten the pages on the digitisation process, geometric distortions appear for certain and, in addition, pages may be degraded due to humidity or to other natural causes. Since the quality of the images depends on the acquisition device, the acquisition task, *i.e.*, the creation of digital images of the pages of antique books in order to become understandable by computers should already

include some procedures, namely to perform geometric corrections and filtering operations. The digitising procedures for obtaining the digital images used in order to develop the current methodology were performed using the *DigiBook* scanner from *Xerox*. The images were acquired in grey level (8 bits) with a spatial resolution of about 600 dpi with the above mentioned pre-processing operations, namely geometrical corrections to attenuate the distortions and filtering to attenuate noise due to the document itself or due to the digitising procedures. Each digital image pixels refers to one complete page of the book varying its dimension from book to book (for instance, each page/image has commonly a typical dimension of about 2000 x 3000 pixels, which occupies 6 Mbytes).

In order to segment the grey level images of the pages, *i.e.*, to create binary images, several approaches were developed (adaptative thresholding, multi-resolution, mathematical morphology) and intensively tested. All of these methods provide advantages as well as some drawbacks in terms of computing time versus the quality of the segmented images. The adaptative thresholding and multiresolution approaches are quite fast but provide sometimes lesser quality binary images, being its description available in [Debora00].

The mathematical morphology approach has a higher computational cost but, generally, provides higher quality segmented pages. The concept guiding this methodology to segment the images refers to a topographic analogy, where the foreground (drop capitals, stripes, figures, annotations and text) can be considered as the "valleys" or "furrows" (darker regions) carved on a generally plane background (lighter regions). Its detection or segmentation consists, in brief, on filling them and subtracting the original image through a procedure inspired on a study developed for segmenting mail letters and postcards [Beucher96] and by introducing some concepts that exploit the information provided by the morphological gradient on the contours of the structures. The complete description of this approach can be consulted on [Mengucci00].

An example of a segmented page using the morphological approach is presented in figure 1. In the original grey level image (figure 1a) the interesting regions (text) are the darker ones where some noisy larger regions originated by humidity appear in dark grey over the background. In the segmented page (figure 1b) the regions that were the direct object of segmentation (in this example, the text) appear in white and the background in black. Although some puzzlement may exist at once due to the creation of a "negative" image, it was decided to maintain the "rule" that is often followed in binary digital image analysis: the regions of interest are represented with the value 1 (white) while the other ones are represented with the value 0 (black). The segmentation procedures do not produce directly binary images free of noise, as can be seen in figure 1b, where some white small regions over the background still remain, but that can be suppressed by adequate binary filtering [Mengucci00b].



(a) (b)

Figure 1 – Morphological page segmentation on a page of the 1st edition of "Os Lusíadas"(1572) by Luís de Camões: (a) initial image (grey level); (b) segmented image (binary).

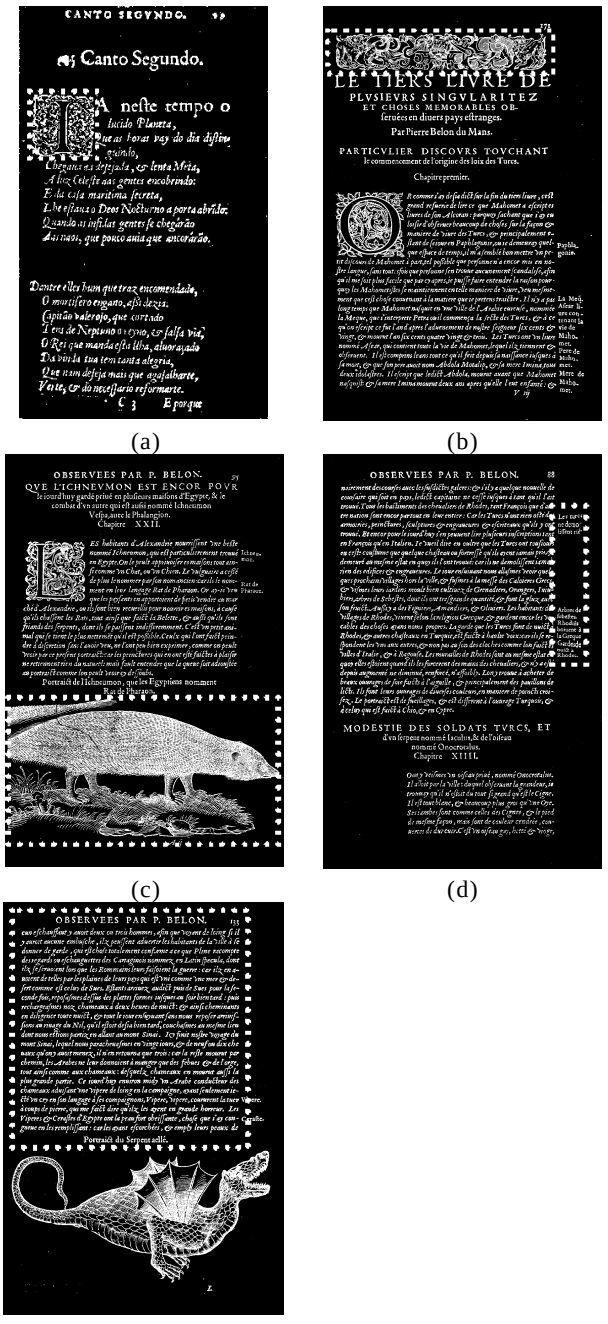
3. PROPOSED METHODOLOGY

A methodology to identify automatically the components that constitute the pages was developed. A page of a book of the XVI century contains several and varied components being previously chosen the following five sets of components, among a larger set, in order to be automatically identified (figure 2): drop capitals, stripes, figures, annotations and text matter. The illuminated letter or drop capital is a capital letter, often ornamental, located at the beginning of a paragraph or chapter (an example delimitated by a dashed square is shown in figure 2a). The strip is a non-decorative element with a long horizontal shape, which adorns the top of the pages, with the purpose of separating chapters and marking the beginning of paragraphs (figure 2b). The figure or illustration concerns all kind of images, engravings, maps or medallions (figure 2c). The annotation or marginal note is a brief explanatory note of text placed in the margins of the text matter (figure 2d). Finally, the text consists on the text matter of the book (figure 2e).

The methodology is hierarchically based once the different components are extracted sequentially (see the scheme in figure 3). The input is constituted by binary images of the pages, being the three levels of the tree processed as follows:

- Level 1 – Consists on the separation of the non-text components (constituted by drop capitals, stripes and figures) from the text components (constituted by annotations and text matter). Although the illuminated letter is a text character its ornamental features indicates that is more suitable to place it in the non-text set.

- Level 2 - Consists on one side (non-text set) on the separation of figures from non-figures (strip and illuminated letter) and on the other one (text set) on the separation of annotations from the text matter.
- Level 3 – At this level the non-figures are separated into stripes and drop capitals. The text matter and also the annotations can then be object of a detailed study, which is out of the scope of the current paper, by detecting the lines and words [Muge00] and by its recognition [CaldasPinto01].



(a) (b) (c) (d) (e)

Figure 2 – Examples of the components of a page delimitated by dashed rectangles/squares: (a) drop capital; (b) strip; (c) figure; (d) annotations and (e) text matter.

4. ALGORITHMS

This phase at level 1 consists on the automatic creation of two sets from the initial binary images: one containing the annotations and text matter (text set), the other one containing drop capitals, stripes and figures.

4.1 Separating text from non-text (level 1)

Based on the segmented pages, the objective in this phase is to separate non-text sets (figures, stripes, drop capitals) from text sets (annotation an text matter). The methodology developed uses, fundamentally, the main directions in which the components of the images evolve, *i.e.*, it is assumed that the characters that constitute the text are aligned along horizontal lines (orientation of 0º) and present a vertical disposition (orientation of 90º). Even for italic fonts this assumption can be considered valid once the "distortion" of these fonts is not accentuated.

Most of the figures are composed of several small lines, whose main directions may vary, not including in the majority of the situations, the ones concerning text characters. Thus, the key-idea is to reinforce the structures oriented on every direction except vertical and horizontal ones, in order to better resist to further filtering operations. On the case of the hexagonal digital grid used, besides its 3 main directions (0º, 60º and 120º) the intermediate directions (30º, 90º and 150º) were also considered.

The developed algorithm consists on the following main phases:

1. Reinforcement of the regions oriented in all directions except the ones of alignment of the text (0º and 90º): applications of directional closings with straight lines as structuring elements. The structures become more compact: holes are filled, concavities disappear or are attenuated and closer objects are connected (figure 4b). This is valid for all the objects present in the image, but at a lower degree to the text once its main orientations are not considered.

2. Suppression of the text set: application of directional openings on the direction normal to the main orientation of the text using a straight line as structuring element with a size equal to the maximum thickness of the lines. The lines are suppressed, remaining the regions bigger than the dimension of the structuring element used (figure 4c).

3. Reconstruction of non filtered regions: application of the geodesic dilation till idempotence. The exact shape of regions that are marked by, at least, a single point can be reconstructed. In this example, only the drop capital and the figure are marked by several sets of points. These structures are reconstructed (figure 4d) while the text, because is not marked by any point, is not recovered. The intersection between the reconstructed image (figure 4d) and the initial one (figure 4a) gives the desired result, an image containing only non-text sets (figure 4e). The text set is now easily obtained by a set difference between the

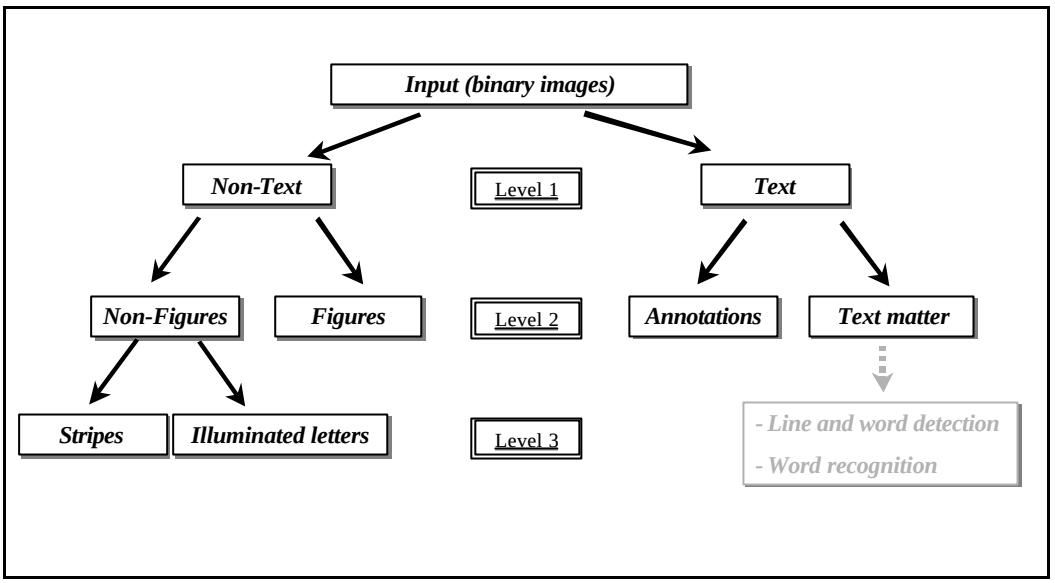


Figure 3 – Schematic methodological approach.

initial image (figure 4a) and the non-text image (figure 4e), whose final result is presented in figure 4f.

Some additional steps of this algorithm, not presented here, are also included in order to deal with some particular features of the images.

Although this algorithm is designed for a general application, it is necessary to adapt its parameter values to each type of book, due to its intrinsic features (typographical and printing features) as well as to its preservation conditions.

4.2 Processing the non-text sets

Within this branch of the hierarchy, the objective is to separate the images that contain only non-text sets, the figures from non-figures (level 2) and, in the following, in the resulting non-figures images, to separate stripes from drop capitals.

4.2.1 Separating figures from non-figures (level 2)

This step consists on creating one set of images containing only the figures and another set of images containing both stripes and drop capitals. The automatic creation of both sets is achieved through the steps of the following algorithm:

1. Creating "strong" objects: It consists on creating one single object per each entity by agglomerating or connecting closer regions (figures and non-figures are constituted by several connected components (figure 5a)). The application of a closing transform with an isotropic structuring element produces the desired result (figure 5b);
2. Creation of a marker of the figures: It is assumed that, at this stage, figures are the bigger structures present in the images. By application of an erosion with an isotropic structuring element (figure 5b), the structures not containing completely at least in some position the structuring element are suppressed. Thus, only some regions of the figures remain and are able to mark them (figure 5c).
3. Identification of figures: It is now simple to identify the figures, by reconstructing the markers obtained in the previous step (figure 5c) in the mask of the strong objects (figure 5b). The set intersection between the initial image (figure 5a) and the reconstructed one (figure 5d) gives the images containing only figures (figure 5e).
4. Identification of non-figures: The set difference between the initial and figures images produces the non-figures sets (figure 5f);

4.2.2 Separating stripes from drop capitals (level 3)

The automatic separation of stripes from drop capitals in images free of text and figures is obtained through the following algorithm stripes:

1. Creation of pseudo-convex hulls: It consists on creating quasi-convex hulls, *i.e.*, single objects for each entity by connecting closer regions through the application of a closing transform (figure 6b) and filling the remaining holes (figure 6c);
2. Creation of markers of the stripes: It exploits the shape of each entity, being the stripes long horizontal structures and the drop capitals more compact (approximately square). The application of a directional erosion along the horizontal direction leaves some regions of the stripes and suppresses all the points of the illuminated letters (figure 6d);
3. Identification of the stripes: Using the markers obtained in the previous step (figure 6d), the reconstruction of these sets with the mask of the pseudo-convex hulls (figure 6c) gives the pseudo-convex hulls of the stripes (figure 6e), whose final shape (figure 6f) is simply obtained by a set

intersection operation between these reconstructed sets and the original image (figure 6a);

4. Identification of the illuminated letters: Once the initial images at the beginning of this step only contain two types of structures, the illuminated letters (figure 6f) are obtained by a set difference between the initial image (figure 6a) and the image containing only the stripes (figure 6e).

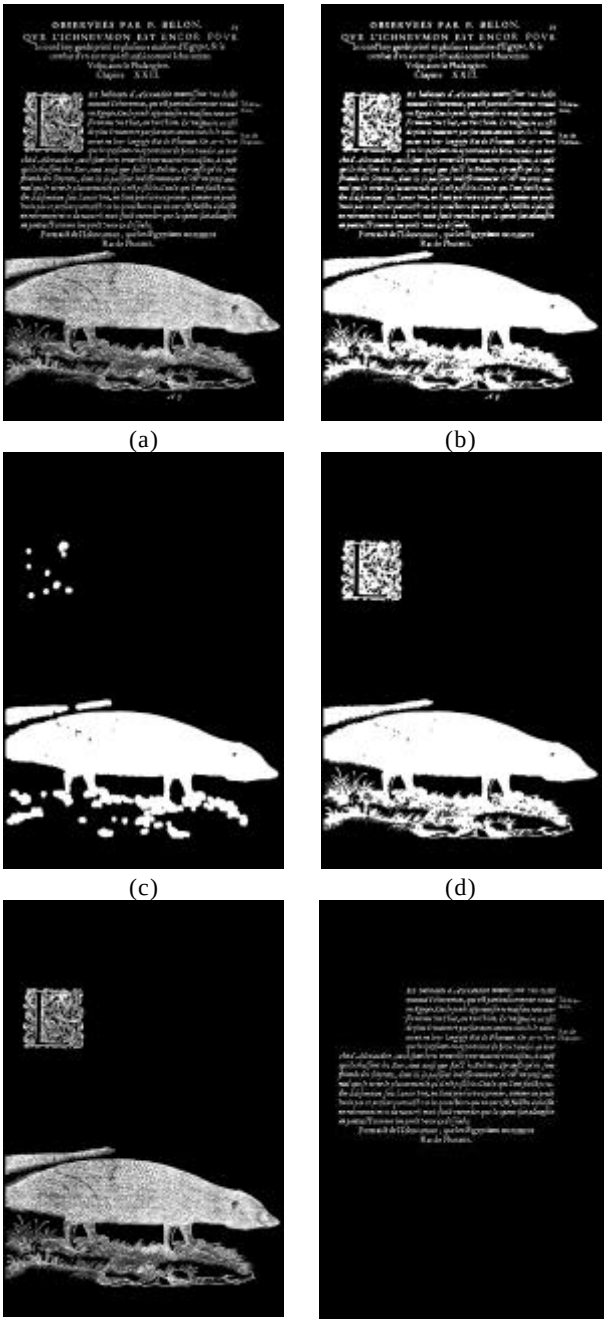


Figure 4 – Algorithm for separating text from non-text: (a) Initial image; (b) Directional closings; (c) Directional openings; (d) Reconstruction of (c) in (b); (e) Intersection of (d) with (a) (final result of non-text set); (f) subtraction of (e) from (a) (final result of text set).

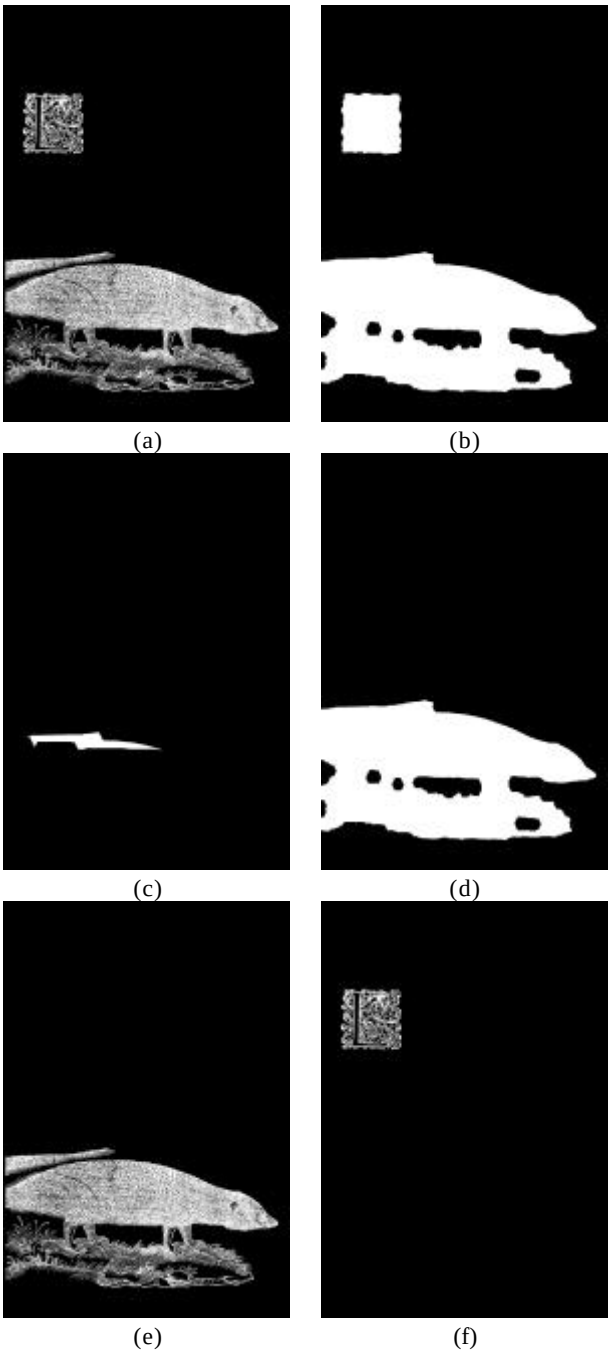


Figure 5 – Algorithm for separating figures from non-figures: (a) Initial image; (b) Closings; (c) Erosion; (d) Reconstruction of (c) in (b); (e) Intersection of (d) with (a)(figure final result); (f) Difference between (a) and (e) (non-figure final result).

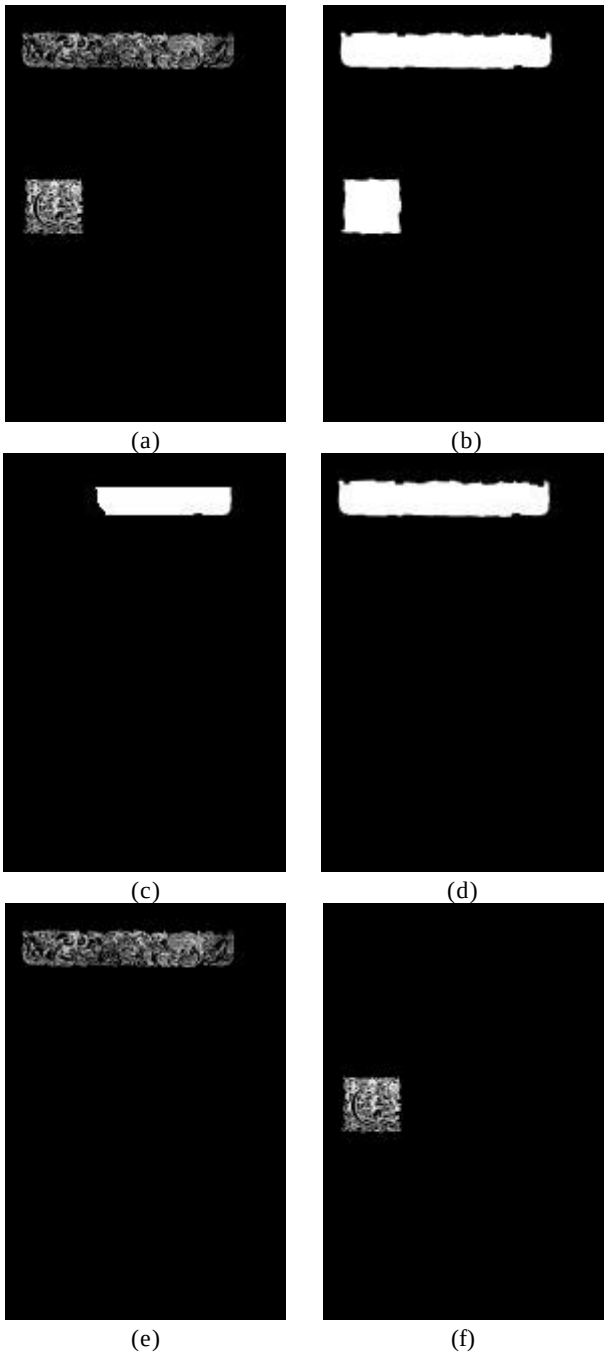


Figure 6 – Algorithm for separating stripes from drop capitals: (a) Initial image; (b) Closing and Hole filling; (c) Linear erosion; (d) reconstruction of strip; (e) Intersection between (a) and (d) (strip final result); (f) subtraction between (a) and (e) (drop capital final result)

4.3 Processing the text sets

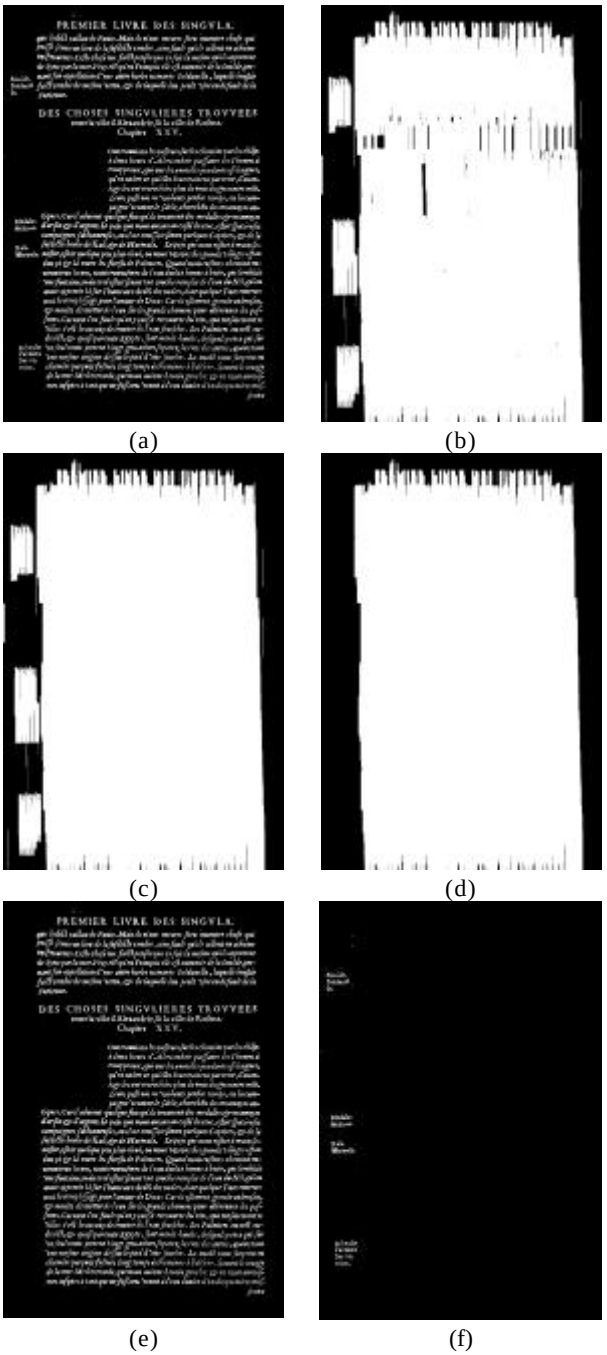
4.3.1 Separating annotations from text matter (level 2)

The separation of these two kinds of text explores the disposition of the annotations and text matter in the pages and also its relative dimensions: the annotations are located in the margins (normally only on one side, on the left or on the right side of the pages) and consist on few lines, while the text matter occupies the central regions of the pages, normally from top to bottom (except the regions occupied by figures and illuminated letters). In order to deal with these geometric features presented by these sets in the pages, the following algorithm was developed:

1. Creation of vertical pseudo-convex hulls: It consists on creating hulls or blocks of the different regions of text. The application of directional closings in the normal direction of the main orientation of the text (horizontal) permits constructing these sets (figure 7b).
2. Creation of markers of text matter: The annotations consist on few lines, being the blocks obtained smaller than the similar blocks of text matter. The application of a suitable erosion permits obtaining markers of text matter (figure 7c).
3. Identification of text matter: The reconstruction of the blocks of text matter produces the masks (figure 7d). The set intersection of these masks with the departure image gives text matter images figure 7e).
4. Identification of annotations: The set difference between the previous images and the initial ones gives the annotations (figure 7f).

4.3.2 Text recognition (level 3)

The recognition of the text in the pages is out of the scope of this paper. The fonts used in printed books of XVI century are not stable, presenting high variations not only from one book to another but also within the same book. The OCR techniques that give good results in recent typographic fonts cannot be applied successfully in these types of books. Several recent studies under development are proposing novel approaches to deal with this difficult task. Readers are advised to consult some preliminary results on [Muge00] and the most recent ones on [CaldasPinto01] where the advances and robustness of the algorithms are successfully increasing the classification rates.



(a) (b) (c) (d) (e) (f)

Figure 7 – Algorithm for separating annotations from text matter: (a) Initial image; (b) Directional closing; (c) Hole filling; (d) Erosion and reconstruction of marker of text matter; (e) Intersection between (a) and (d) (text matter mask); (f) Set difference between (a) and (e) (annotations)

5. CASE STUDY: APPLICATION TO PIERRE BELON'S BOOK

The developed algorithms were tested on an entire digital version of a book previously segmented. The book that was chosen is *Les observations de plusieurs singularités et choses mémorables trouvées en Grèce, en Asie, en Judée* from *Pierre de Belon*, dates from 1554 and belongs to the *Bibliothèque Municipale de Lyon* in France. The available input consisted on 451 pages/binary images of 2084 x 3244 pixels.

The automatic application of the developed algorithms followed the sequence presented in figure 3. In the following, each one of the 451 pages was manually analysed in order to verify the result of application of each algorithm in order to quantify the errors committed. The results obtained at each level for each type of separation are presented in table 1, where the rate concerns the successful cases in number (Nb) and in percentage (%).

Level	*Type of separation*	*Rate*	
		Nb	*%*
1	Text ↔ Non-text	441/451	97.78
2	Non figures ↔ Figures	254/280	90.71
2	Annotations ↔ Text matter	222/298	74.50
3	Stripes ↔ Drop capitals	5/5	100.00

Table 1 – Classification rates obtained on Pierre Belon's book (451 pages)

At level 1, the separation of text from non-text is excellent producing very few errors (classification rate of 97.78%). The situations where the algorithms failed (10 pages out of 451) are very particular and are related to figures that contain text characters inside them.

At level 2, the separation of non-figures from figures was successful in 90.71% of the cases, which is also excellent. The separation of annotations from text matter also gives good results (classification rate of 74.51%), residing the problem from a not very satisfying previous geometric correction. This is problematic once some pages do not present a similar distortion along the pages. Thus, the global geometric correction must be corrected by using other algorithms that can act differentially from region to region within the same image or page. This modification would enable a more correct application of this algorithm.

At level 3, the separation of stripes from drop capitals is completely correct.

6. CONCLUSIONS

In this paper a general methodology to extract the components of pages of antiques books was proposed. It is based on mathematical morphology operators and explores the geometric features of the structures that are present in the books. The classification rates obtained are excellent and indicate that the approach followed is correct. Although this methodology was intensively applied to a single book, its degree of generalisation was also achieved, once pages from other books of XVI century were also analysed with similar successful results.

Anyhow, there are some improvements that still are under development, namely related to the choice of the parameters of each algorithm that vary from book to book. Presently, that choice is done after a previous analysis of the components of the pages of each book. It is envisaged to develop some approach in order to determine the automatic choice of the values of the parameters of each algorithm.

7. ACKNOWLEDEGMENTS

The methodology presented and the results obtained are being developed under the European Union project named DEBORA (1998-2001, contract no. LB5608/A) under the *DGXIII-Telematics for Libraries Programme* (http://www.enssib.fr/divers/debora/).

8. REFERENCES

[Agam96] G. Agam, I. Dinstein, 1996, Adaptive Directional Morphology with Application to Document Analysis, *in P. Maragos, R.W. Schafer, M.A. Butt (eds.),* Mathematical Morphology and its Applications to Image and Signal Processing, 401-408, Kluwer Academic Publishers, Boston.

[Beucher96] S. Beucher, S. Kozyrev, D. Gorokhovik, 1996, Pré-traitement morphologique d'images de plis postaux, in *Actes CNED'96 - 4^{éme} Colloque National Sur L'Écrit Et Le Document*, 133-140, Nantes, France.

[CaldasPinto01] J.R. Caldas Pinto, A. Marcolino, M. Ramalho, F. Muge, N. Sirakov, P. Pina, Comparing Matching Strategies for Renaissance Printed Words, in *10EPCG – 10º Encontro Português de Computação Gráfica*, in this volume.

[Cumplido96] M. Cumplido, P. Montolio, A. Gasull, 1996, Morphological Preprocessing and Binarization for OCR Systems, *in Maragos P., Schafer R.W., Butt M.A. (eds.), Mathematical Morphology and its Applications to Image and Signal Processing*, 393-400, Kluwer Academic Publishers, Boston.

[Debora00] R. Bouché, H. Emptoz (eds.), *Debora, European Project LB5608A*, ENSSIB, Lyon, France, 177 pp.

[He96] S. He, N. Abe, 1996, A Clustering-Based Approach to the Separation of Text Strings from Mixed Text/Graphics Documents, in *Proceedings of ICPR '96 - 13th International Conference on Pattern Recognition* (Vienna, Austria), 706-710, IEEE Computer Society Press, Los Alamitos, California.

[Mengucci00a] Mengucci M., Granado I., Muge F., Caldas Pinto J., 2000, A Methodology based on mathematical morphology for the extraction of text and figures from ancient books, in *Campilho A.C. & Mendonça A.M. (eds.), Proceedings of RecPad'2000 - 11th Portuguese Conference on Pattern Recognition*, 471-476, Porto.

[Mengucii00b] M. Mengucci, I. Granado, 2000, Morphological segmentation of text and figures in renaissance books (XVI Century) in *Goutsias J., Vincent L.& Bloomberg D.S. (eds.), Mathematical Morphology and its Applications to Image and Signal Processing*, 397-404, Boston, Kluwer Academic Publishers.

[Montolio93] P. Montolio, T. Gasull, L. Corbera, F. Marqués, 1993, Character Recognition and Document Analysis by Morphological Techniques, in Salembier Ph (ed.), International Workshop on Mathematical Morphology and its Applications to Signal Processing, Barcelona.

[Muge00] F. Muge, I. Granado, M. Mengucci, P. Pina, V. Ramos, N. Sirakov, J.C. Pinto, A. Marcolino, M. Ramalho, P. Vieira, A.M. Amaral, 2000, Automatic feature extraction and recognition for digital access of books of the Renaissance, in *Borbinha J. & Baker Th. (eds.), Research and Advanced Technology for Digital Libraries, Lecture Notes in Computer Science - vol. 1923*, 1-13, Springer, Berlin.

[Serra82] J. Serra, 1982, *Image Analysis and Mathematical Morphology*, Academic Press, London.

[Soille99] P. Soille, 1999, *Morphological Image Analysis*, Springer; Berlin.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Comparing Matching Strategies for Renaissance Printed Words

J. Caldas Pinto A. Marcolino M. Ramalho
IDMEC, Instituto Superior Técnico
Av. Rovisco Pais, 1049-001 Lisboa, PORTUGAL

F. Muge N. Sirakov P. Pina
CVRM, Instituto Superior Técnico,
Av. Rovisco Pais, 1049-001 Lisboa, PORTUGAL

Abstract
In this paper two different approaches for matching words from books from the Renaissance period are compared. The first methodology is a novel search-based approach, while the second one is based on FNS-Finite Numerical Sequences. Both methodologies are illustrated with several examples. The comparison of the results obtained with both methodologies are presented, being also discussed the contribution that both of these methodologies can give to word recognition tasks.
Key words
Document Image Processing, Matching, Renaissance Books, Digital Libraries

1. Introduction

Johannes Gutenberg invented the printing press in the middles 1400's. However it remained some centuries until graphical types were normalised and automatically done, allowing regularity that can be seen in the present text.

Among invaluable library documents are manuscripts or, commonly said, 'ancient books', which, beyond their rarity and value, represent the 'state of knowledge'. They constitute a heritage that should be preserved and used. These two considerations present conflicting aspects that should be overcome. How to use antique books, without damaging and degrading them? One response is to store them on a suitable media, a digital support, and to allow their use over a suitable system, as computer networks. An ergonomic interface that permits to a user to interact with books, allowing their efficient consultation, for example about the presence or absence of certain specific graphic characteristics or even given keywords.

Effectively XVI century books are irregular in format and graphic quality that makes them unique exemplars. This address particular problems to recognition, which, cannot be supported by a dictionary or grammatical database.

When can two binary images be considered equal in some sense? One answer is to consider that two binary images are equal when the object in one image represents the same object in another image. The problem of matching two identical images may be an easy task for humans but, unfortunately, is still a difficult task to be accomplished in general by a computer [Alder].

Since the semantic meaning of the words is not taken into account in case of shape matching, the notion of string will be used hereafter. A number of different methods have been employed for solving the string matching problem [Bouchaffra99] [Cha99] [Heute96] [Rigoll96]. Some of them are structural, statistical or combined , other are based on Hidden Markov Methods and Neural Networks [Bouchaffra99] [Rigoll96]. These methods are oriented mainly to handwritten character recognition.

Neural networks [Mitchell97] [Russell95] achieve good results in the recognition of machine printed characters. Nevertheless problems still occur in the presence of noisy documents, when broken and touching characters are common. Another problem comes up, when the recognition system is confronted with a typeset for which was not trained. The recognition rate in both cases decreases enormously.

In addition, character segmentation problems occur due to a weak binarisation process, where broken and touching characters are common. An OCR approach with neural networks is not a good solution to be applied due to this fact. In the subject of the handwriting recognition a statistical framework, such as a Hidden Markov Model [Rabiner86] [Rabiner89], is rather used, where the Bakis (Left-Right) model is normally chosen and trained with the Baum-Welch algorithm. The Viterbi algorithm is then applied to compute the desired results.

To employ a Hidden Markov Model a vector quantization [Gray84] of the extracted features is normally performed to reduce the space dimensionality. The choice of the features and the method to split them is a subject that varies from author to author

[Guillevic97] [Kuo94]. No common or standard set of features can be asserted as the best one. The performance of the Hidden Markov Model is directly related with the skills of the designer who has the ability to pick the best features to submit to the Baum-Welch algorithm. A trial and error iterative process is applied until the results became satisfactory.

In this paper are presented two approaches developed to perform the matching of words in binary images of Renaissance books. The first one is based on a novel search-based approach while the second one appeals to Finite Numerical Sequences.

2. Automatic Feature Extraction

A methodology that integrates several digital image analysis techniques to automatically extract and recognise features from books of the Renaissance was presented recently [Muge00]. After image acquisition procedures (grey level images), it consisted on the following four main phases: (1) page segmentation, (2) separation of text from non-text, (3) line detection and (4) word detection.

Using the grey level images, a mathematical morphology [Serra82] [Soille99] based methodology for page segmentation was developed in order to filter all the undesired regions (shadows, dirt spots) and to create binary images containing the information that constitute the pages of the books (text and figures). This methodology is based on global and local features of the image.

Based on the segmented pages, the separation of non-text (figures, labels, illuminated letters) from text characters was performed automatically, appealing once again to a mathematical morphology based approach. The use of operators that explore the horizontal orientation of the text lines and also operators that explore size and connectivity features are in the basis of the developed methodology.

The line detection phase is performed upon the binary images resulting from the previous algorithm, *i.e.,* on the text images only. A simple and fast algorithm that computes the projection profile across horizontal lines of the image was used taking always in consideration the necessity of performing local geometric corrections.

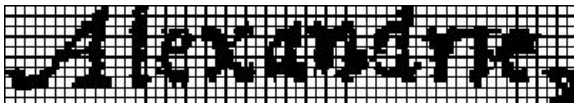
The process of word detection or segmentation consists on processing a binary image containing only text characters and the respective detected lines. This process has some steps that are very similar to the problem of line segmentation, once it is based on projection profile across vertical lines and by detecting the adequate thresholds to detect the spaces between adjacent words. In addition, a rectangular bounding box represents also each word.

3. Methodological approaches

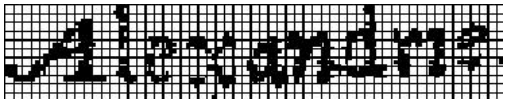
3.1. Novel search-based approach

This search-based approach suppresses the costly iterative trial and error process, by proposing of a novel solution to bypass the phase of designing the best features employed in Hidden-Markov Models.

Given two binary images it is intended to quantify how close or similar these images are. The binary images used in this experiment are composed by a series of connected components. The connected components represent characters, broken characters and joint characters obtained from a digitized version of a book printed in the 16th century. Let us assume that an image (denominated a) is represented by a series of disjoint sub-images (denominated a_i)). Then to consider that two images, a and ß, are identical, for each sub-image of a a corresponding sub-image in ß must exist. The method uses an algorithm that searches for each sub-image of a, a sub-image in ß that minimizes the cost of choosing to match the sub-image of a with the sub-image of ß. The cost function used in the experiments returns the sum of the values of the pixels of the image ?, where ? represents the difference between a_i and $ß_j$, on pixel per pixel basis. Collecting the value of all the pairs (a_i , $ß_j$) matched during the search, results in a measure that gives the degree of difference between a and ß. Conversely, if the degree of difference between a and ß is low, then a and ß can be considered similar. The two images of the word 'Alexandrie' from figure 1 are used to illustrate this approach. The top image (figure 1a) is considered as a and the bottom image (figure 1b) is considered as ß. Although to a human is easy to see that both words are the same, some problems may arise to a computer program to identify correctly all the characters: the last characters 'r' and 'i' are merged and can be easily confused as a 'n'.



(a)



(b)

Figure 1 – Two printed 'Alexandrie' words

The procedure starts by splitting a in a series of disjoint sub-images, as can be seen on figure 2. For each one of the sub-image of a (in this example 29 sub-images were considered), a corresponding sub-image in ß, will be searched. For instance, let us consider a_{26}, which corresponds more or less to the last 'e' of the word, displayed in figure 2. The corresponding sub-image in ß is obtained if a shift of **x** pixels along the x-axis is executed. In the same manner, a shift of **y** pixels may be executed along the y-axis to align the sub-images. The computation of the value of **x** and **y** is achieved by

executing a search around the neighbourhood of the location where the corresponding sub-image is expected to be found. A partial view of the tree constructed during the search is showed in figure 3.

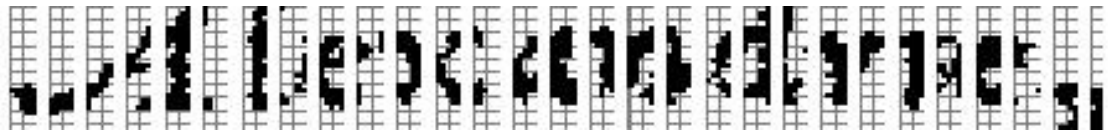
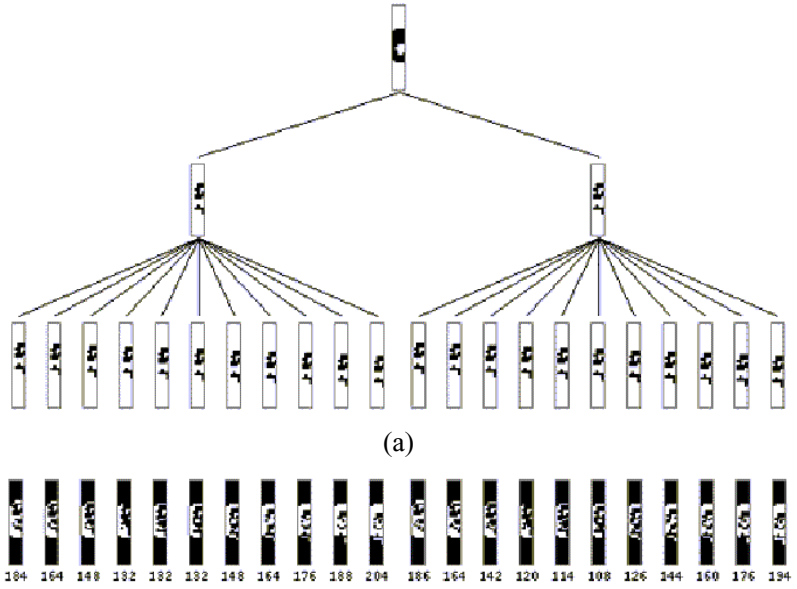


Figure 2 – 'Alexandrie' word split in 29 disjoint sub-images (each one is 10 pixels wide).



(a)

(b)

Figure 3 – (a) Partial view of the tree search. (b) Values of the objective function.

At each terminal node of the tree the objective function is evaluated. The objective function counts the number of zeros (*i.e.*, black pixels) of ?. ? is the difference between the image in the terminal node, and a_i, on a pixel per pixel basis. For each sub-image of a is chosen, as a match, the image looked in the neighbourhood, that gives the minimum value processed by the objective function. Summing up all these values, the result obtained enables to compare a to ß.

Finally to compare two images, it suffices to compare a with ß and, once there is no symmetry in the operation, also ß with a. The minimum value between these two values is the one returned.

3.2. Finite numerical sequences approach

Due to the ancient technology used for printing of Renaissance books the shapes of the letters can vary from page to page. Some letters are broken to two or more parts, some letters are damaged (some regions are missing) and other letters are fused together. But on the other hand, the Renaissance letters shape does not vary so much as in hand written case. According to these reasons the shape matching is an appropriate approach to explore.

The notions related to shape such as convexity and connectivity are suggested in [Russ99] as being useful to be applied for shape description of letters, but without proposing a general methodology. Shape factor, roundness, aspect ratio, elongation and curl parameters [Soille99] may be also applied as letters features descriptors, but normally needing additional information to provide a correct matching in case of digital Renaissance books.

An approach for shape matching of Renaissance letters and strings based on notions such as regularities, consistencies, essential border points [Sirakov96] [Sirakov94] is proposed in the following where a string model is developed as an aspect graph tree. The shape of each letter is presented as consistency (vector of consistencies, when several objects present the letter's shape (figure 4)). Further, each consistency falls apart to four finite numerical sequences (FNS). This way, the

problem of matching Renaissance letters and strings is reduced to the problem of FNS matching, allowing a significant reduction of the complexity of computations.

The model of a string (word) containing Renaissance letters is developed as an aspect graph tree, using the following notions: 'regularities', 'consistencies', 'essential points of the letter's border' and FNS [Sirakov96] [Sirakov94] [Sirakov]. The border of each letter (2D object) is approximated, with a given accuracy, to a closed polygon.

'Regularity' is considered to be the direction of motion of a point over a closed polygon (figure 4a). The angles satisfied by regularities are defined as follows:

$$for S^1 : \varphi^1 = \frac{\pi}{2},\ for S^8 : \varphi^8 = 3\frac{\pi}{2},\ for S^2 : \varphi^2 = 0,\ for\ \bar{S}^7 : \varphi^7 = \pi$$

$$for S^3 : \varphi^3 = (0, \frac{\pi}{2}),\ for S^6 : \varphi^6 = (\pi, 3\frac{\pi}{2}),\ for S^4 : \varphi^4 = (3\frac{\pi}{2}, 2\pi),$$

$$for\ S^5 : \varphi^5 = (\frac{\pi}{2}, \pi) \qquad (1)$$

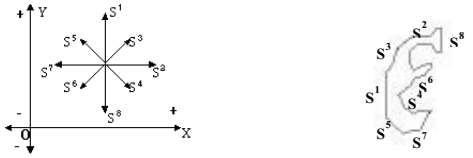
Regularity represents just one edge of a closed polygon, which approximates the letter border by a given accuracy, while consistency is defined as consecutive connected regularities. As an example, the border of the letter "E" is approximated with an accuracy of 0.5 by the closed polygon shown in figure 4b. The polygon is described by the consistency presented in equation (2). "*Cl*" closes the consistency.

$$S_1^5 \rightarrow S_2^1 \rightarrow S_i^3 \underset{i=3}{\overset{5}{\rightarrow}} S_6^2 \rightarrow S_7^3 \rightarrow S_8^2 \rightarrow S_9^8 \rightarrow S_{10}^7 \rightarrow$$

$$S_{11}^5 \rightarrow S_{12}^7 \rightarrow S_{13}^6 \rightarrow S_{14}^8 \rightarrow S_{15}^4 \rightarrow S_i^3 \underset{i=16}{\overset{17}{\rightarrow}} S_{18}^8 \rightarrow S_i^6$$

$$\underset{i=19}{\overset{22}{\rightarrow}} S_i^4 \underset{i=23}{\overset{24}{\rightarrow}} S_{25}^3 \rightarrow S_{26}^2 \rightarrow S_i^6 \underset{i=27}{\overset{28}{\rightarrow}} S_{29}^7 \rightarrow S_{30}^6 \rightarrow Cl \qquad (2)$$

Vertices that link different regularities are called "essential points of the letter's border " [7N] [8N]. Vertices that link regularities of one and the same type are called 'non-essential'. A vertex is used as a starting point of a polygon description that is accomplished by employing consistencies. In our case this is the polygon vertex with coordinates (x_{min}, y_{min}).



(a) (b)

Figure 4 - a) The plane points satisfy 8 regularities; b) Letter border, the regularities of its edges satisfy.

Further on, each consistency falls apart to four finite numerical sequences:

Rg - sequence containing regularities that build the consistency

Rp - sequence containing repetition indicators and showing how many times each regularity from *Rg* consecutively participates in the consistency;

An - angles that the regularities of *Rg* conclude with the axis Ox^+;

Le - lengths of the regularities belonging to *Rg*.

As a result, the consistency shown in equation (2) falls into four numerical sequences:

Rg = 5,1,3,2,3,2,8,7,5,7,6,8,4,3,8,6,4,3,2,6,7,6;

Rp = 1,1,3,1,1,1,1,1,1,1,1,1,1,2,1,4,2,1,1,2,1,1;

An = $\varphi_1,....,\varphi_{30}$;

Le = $l_1,....,l_{30}$ (3)

It follows from the definition of the regularities S^1, S^2, S^7, S^8 that *Rp* = 1. Thus, the geometrical model of each closed polygon (contour) approximating a letter border, when the letter is represented by a single object (which is not always the situation as figure 5 shows), may be presented as four finite numerical sequences. Employing the idea of finite numerical sequences, the model of a string containing Renaissance letters is developed as an aspect graph tree with six levels (figure 5). The following parameters are used there:

*O*k - string of letters;

Cki- letter i that belongs to the string k. If the letter contains only one object (for example "i" and "j" contain 2 objects), then Cki denote consistency;

It may happen in some cases that some of the letters are broken to two or more objects. In this case C_{ji} denotes the following vector $C_{ji} = (C_{ji}^1, \ldots, C_{ji}^u)$ (4)

where C_{ji}^v, for v=1,…,u, is a consistency and u is the number of the objects that present the corresponding letter. The consistencies in the upper vector are ranked according to the position of the object in the letter and using the following description rule: objects description is performed from the left corner of the letter to the right one and from bottom to top.

Using the presented model, the problem for matching strings, which contain the Renaissance letters, is reduced to matching aspect graph trees. It allows a significant decrease of calculation complexity of the algorithms. Only strings (words) with "almost" the same size of the rectangular hull of the keyword are considered for matching.

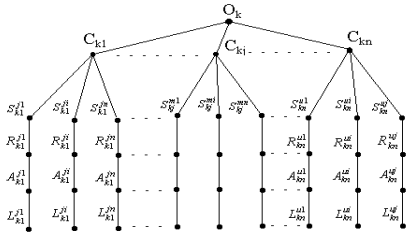


Figure 5 - The string's model as an aspect graph tree.

Observing the graph structure it may be noted that the second level is a sequence of vectors and four nodes are consecutively connected with each vector element. A number may be considered with each graph node by using the following mappings: $C_{ki} \rightarrow i$, $C^{v}_{ji} \rightarrow v$, $S^{mj}_{ki} \rightarrow mj$, $R^{mj}_{ki} \rightarrow$ the number of the consecutive repetitions of S^{mj}_{ki}, $A^{mj}_{ki} \rightarrow$ the angle of S^{mj}_{ki}, $L^{mj}_{ki} \rightarrow$ the length of S^{mj}_{ki}. This way the string model shown in figure 5 may be presented by set of finite numerical sequences (FNS).

It follows that the problem for matching graph trees may be restricted to matching finite numerical sequences. A set of rules is given hereafter in order to solve this problem.

By definition S^i and S^j are called different if $i \neq j$. They are called almost similar if $i=j$, but $|\varphi^i - \varphi^j| \leq d^1$, $|l^i - l^j| \leq d^2$, for $d^1 \neq 0$ and $d^2 \neq 0$. When $d^1 = 0$, the regularities are similar, and when $d^1 = 0$, $d^2 = 0$ they are equal to one another. The terms are denoted as follows: almost similarity by "$\approx$", similarity by "$\cong$" and equality by "$\equiv$". Consider FNS – a_1, a_2, and two elements: $e_{1i} \in a_1$, $e_{2j} \in a_2$. The numbers i and j give the position of the element in FNS.

Consider the elements e_{1i} and e_{2j} almost similar if: $/i - j/ \leq d^3 \wedge /e_{1i} - e_{2j}/ \leq d^4$, where $d^3 \neq 0$ and $d^4 \neq 0$. When $d^3 \neq 0$ but $d^4 = 0$ the elements are similar, and when $d^3 = d^4 = 0$ they are equal to one another. If FNS a_1 and a_2 satisfy the inequality $/\max(/a_1/,/a_2/) - m/ \leq d^5$ they are called almost similar when m is the number of the almost similar elements; similar - when m is the number of the similar elements; equal - when m is the number of equal elements between the sequences. d^5 denotes a threshold. It follows from the above definition that two consistencies c_i and c_j are:

(i) almost similar ($c_i \approx c_j$) if $Rg_i \approx Rg_j$, $Rp_i \approx Rp_j$, $An_i \approx An_j$, $Ln_i \approx Ln_j$ (5)

(ii) are similar ($c_i \cong c_j$) if $Rg_i \cong Rg_j$, $Rp_i \approx Rp_j$, $An_i \cong An_j$, $Ln_i \approx Ln_j$ (6)

(iii) equal ($c_i \equiv c_j$) if $Rg_i \equiv Rg_j$, $Rp_i \equiv Rp_j$, $An_i \equiv An_j$, $Ln_i \equiv Ln_j$ (7)

and the following inequalities are satisfied: $L_1 - LN \leq \frac{L_1}{d}$, $L_2 - LN \leq \frac{L_2}{d}$. The notations used have the following meanings: d – threshold; L_i for i=1,2 - total length of the corresponding consistency; LN - total length of the equal parts of the FNS - Rg. The sequence Le (see equation (3)) is the one used to calculate L. Moreover, to calculate LN the sequences Le and the result of the comparison between sequences Rg are used.

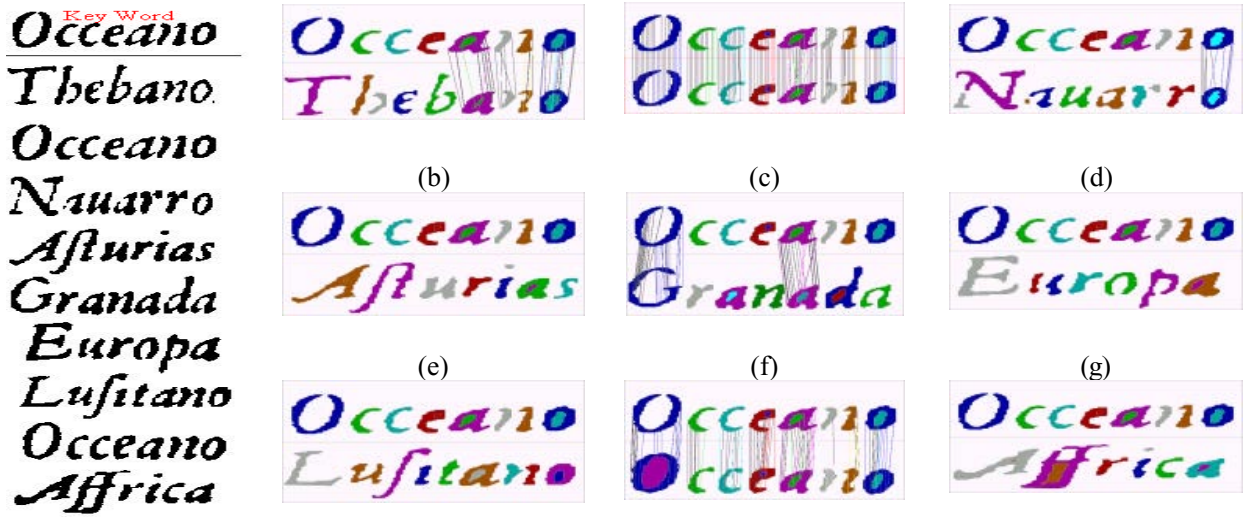


Figure 6 – (a)List of words whose rectangular hulls have the same size as the keyword 'Occeano' (first word on the top) with different degrees of matching: (b) only three letters are similar; (c) all the letters coincide; (d) Only the 'o' is found similar; (e) No similar objects were found; (f) A disadvantage is presented here, where the letters "O" and "G" are considered similar, once the "O" is broken and has a consistency similar to the consistency of "G"; (g and h) all pairs of corresponding letters are different; (i) complete degree of similarity; (j) No similar objects were detected

It should be noted that the matching result depends on the thresholds $d^1,...,d^5$. The user may choose the values, depending on his knowledge about the considered Renaissance letters.

In the case of broken letter, more than one closed polygon (objects) describing it. The notion "morphological similarity" and the approach presented in [Sirakov] are applied here, in order to define similar objects that belong to one and the same letter. Remember that the problem for matching Renaissance words is reduced to matching aspect graph tree, which we restricted to processing and matching of finite numerical sequences. It lead to significant decreasing of calculation complexity of the matching algorithms, which is very important when large number of Renaissance words have to be matched in quasi real time.

4. Results

The novel search-based and FNS approaches were tested against a set of keywords. A librarian, expert on the Renaissance period, selected 10 relevant keywords from the Pierre Belon's book *"Les observations de plusieurs singularités et choses mémorables trouvées en Grèce, en Asie, en Judée" (1554).* The available input consisted on 451 pages/binary images (1092x1622 pixels) that were previously de-skewed.

In what concerns the novel search-based approach, for each one of the keywords a rank list was generated, with all the images similar to the keyword appearing first in the list. The critical factor of success of the proposed method relies in two points. First, the ability of bypass the extraction and selection of features usually found in statistical pattern recognition [Jain00]. Second, the power of jumping above the problems associated with character segmentation in noisy documents, namely broken and touching characters. The developed system and the approach used are also being used as a test bed for future projects, given the quality of the results produced by the proposed method. In figure 7 are presented the 'Arabie' keyword and the first 29 more similar words that were found, by decreasing order of similarity, which correspond to all 'Arabie' words in the text, together with he first 11 mismatches.

In what concerns the FNS approach, two types of output are provided: one constituted by images the other one by a text file. Each image contains the result of matching between the keyword and the corresponding one from the list, being the similar letters painted in one and the same colour and the corresponding similar parts of the border connected with straight elements. On the other hand, the text file contains the FNS describing the border of each letter, the essential border points and the result of matching. In figure 6 are presented a list of words selected as candidates to be compared with the keyword 'Occeano', and the results of matching with the FNS approach.

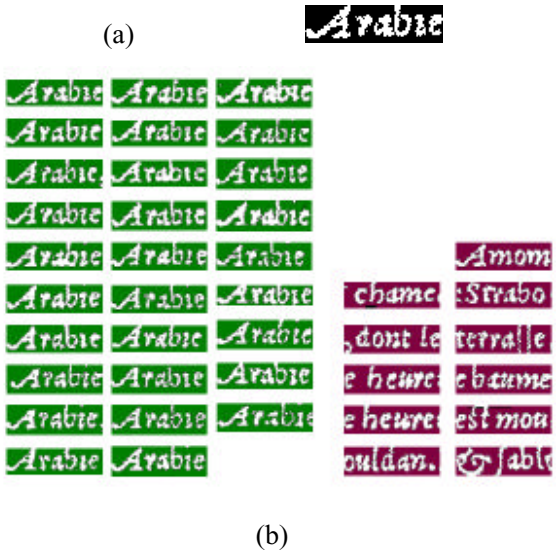


Figure 7 – (a) Selected keyword and (b) Similar words appear at top positions (left) and dissimilar words at the following rank positions (right)

Two strings (words) are considered as equal if and only if they have the same number of letters and the corresponding letters are similar. The notion "corresponding" means that only letters situated on one and the same place in the strings are subject of matching (first with first, second with second etc).

The global degree of matching for both approaches using the 10 selected keywords ranging from 90% to 100 %, which can be considered highly satisfying.

5. Comparisons of Techniques

Within this paper two novel approaches were compared, in order to perform the matching of words from books of the Renaissance.

The SBA approach allows searching and comparing similar images in noisy documents. Its stronger characteristic is to bypass the extraction and selection of features usually found in statistical pattern recognition and to overcome problems associated with character segmentation in noisy documents, namely broken and touching characters.

Similarly the FNS also successfully recognized words of Renaissance books. The description of the shape of objects (letters) by means of finite numerical sequences allows the definition of a shape similarity measure independent from size and orientation features.

A comparison for the classification of both approaches is presented in figure 8, were, both similarity measures are plotted against each other. Correct words are plotted with a green circle and incorrect words are plotted with a red triangle.

From this chart several remarks can be drawn. Firstly

the highly success rate of recognition of both approaches. Effectively is clearly seen that higher rates are obtained by the correct words. Secondly the clear two areas defined by points representing matched and unmatched words.

Thirdly the fact that some disagreements exist. Indeed a few candidates are misclassified by one of the methods.

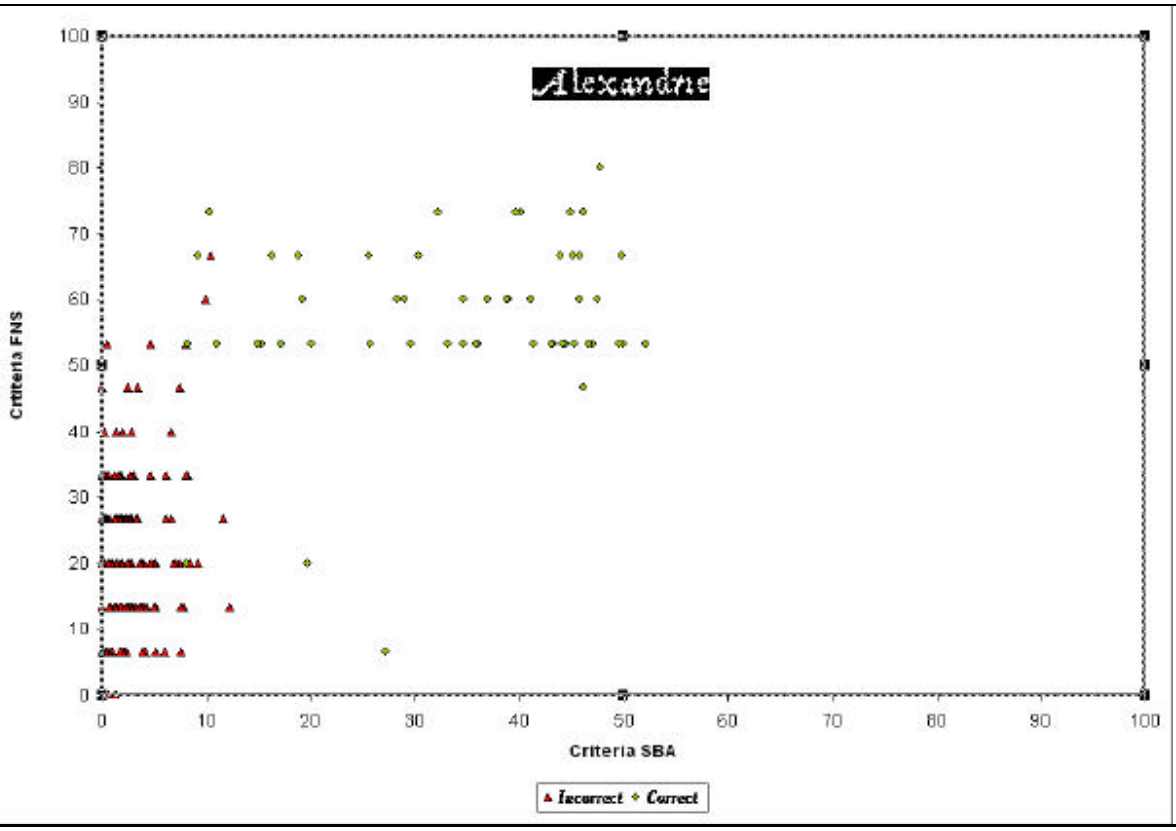


Figure 8 Comparison between SBA anf NFS approaches

The table in figure 9 presents some examples of misclassification. Firstly correct or almost correct words, which were discarded by FNS.

	SBA	FNS	
A	19.76	20	
A	27.18	6.66	
N	10.40	66.66	
N	9.88	60	
	8.06	20	

Figure 9

Effectively the excessive proximity in one case, and a clear division but with one character left, did not allow the correct matching. More interesting are cases N. Although a higher FNS match score, both words are clearly incorrect. These words correspond to the two highest points (red triangle) in the graph of figure 8.

Finally, a misprinted word which, probably due to the 'r' and 'i' connection that has been lost by both methods. Results presented suggest that a refinement of both methods will be useful despite their successful rate. However, from figure 8, it can be suggested a complementary analysis of doubtful candidates and thus to be submitted to a more careful analysis.

6- Conclusions

Within paper two different approaches for matching words from Renaissance period books are compared. The first methodology is a novel search-based approach, while the second one is based on FNS-Finite Numerical Sequences. Both methodologies were illustrated with several examples. The comparison of the results obtained with both methodologies were presented, being also discussed the contribution that both of these methodologies can give to word

recognition tasks. Both techniques performed with high degree of success. It can be seen that the SBA approach gives a more detailed match measure.

Acknowledgements

This project is supported by project DEBORA funded by the European Union-DGXIII (Telematics programme, LB-5608/A-DEBORA 25325/0) and "Programa de Financiamento Plurianual de unidades de I&D (POCTI), do Quadro Comunitário de Apoio III".

References

[Alder] Alder M.D., Principles of Pattern Recognition: Statistical, Neural Net and Syntactic methods of getting robots to see and hear. http://maths.uwa.edu.au/ ~mike/

[Bouchaffra99] Bouchaffra D., Govindaraju V., Srihari S.N.,1999. Recognition of Strings using Non-Stationary Markovian Models: An Application to Zip Code Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition, June 23-25, 1999, Fort Collins, Colorado, USA.

[Cha99] Cha S.H., Shihand Y.C., Srihari S.N., 1999, Approximate Stroke Sequence String Mathing Algorithm for Character Recognition and Analysis, Proc. ICDAR'99 - 5th International Conference on Document Analysis and Recognition, Bangalore, India, Sept 1999.

[Gray84] Gray R.M., 1984, Vector Quantification, IEEE ASSP Magazine.

[Guillevic97] Guillevic D., Suen C.Y., 1997, HMM Word Recognition Engine, Proc. 4th ICDAR.

[Heute96] Heute L., Moreau J.V., Paquet T., Lecourtier Y., Olivier C., 1996. Combining of Structural and Statistikal Features for the Recognition of Hand-written Characters. (13th ICPR, TU Vienna, Austria, August 25-29. 1996) Pattern Recognition and Signal Analysis, IEEE Computer Society, Vol. II, 1996, pp.210-215.

[Jain00] Jain K.A., Duin R.P.W., Mao J., 2000, Statistical Pattern Recognition: a Review, IEEE TPAMI.

[Kuo94] Kuo S.S., Agazzi O.E., 1994, Keyword Spotting in Poorly Printed Documents Using Pseudo 2-D Hidden Markov Models, IEEE TPAMI.

[Mitchell97] Mitchell T.M., 1997, *Machine Learning*, McGraw-Hill.

[Muge00] Muge F., Granado I., Mengucci M., Pina P., Ramos V., Sirakov N., Pinto J.R.C., Marcolino A., Ramalho M., Vieira P., Amaral A.M., 2000, Automatic feature extraction and recognition for digital access of books of the Renaissance, in *Borbinha J. & Baker Th. (eds.), Research and Advanced Technology for Digital Libraries*, *Lecture Notes in Computer Science - vol. 1923*, 1-13, Springer, Berlin.

[Rabiner86] Rabiner L.R, Juang B.H., 1986, An Introduction to Hidden Markov Models, IEEE ASSP Magazine.

[Rabiner89] Rabiner L.R., 1989, A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition, Proceedings of the IEEE.

[Rigoll96] Rigoll G., Kosmada A., Rottland J., Neukirchen C., 1996, A Comparison Between Continuous and Discrete Density Hidden Markov Models for Cursive Hand-written Recognition, in Proc. 13th ICPR, TU Vienna, Austria, Pattern Recognition and Signal Analysis, IEEE Computer Society, Vol. II, 205-209.

[Russ99] Russ J.C., 1999, *The image processing-handbook*, Springer & CRC, Heidelberg.

[Russell95] Russell S., Norvig P., 1995, *Artificial Intelligence: a Modern Approach*, Prentice-Hall.

[Serra82] Serra J., 1982, *Image Analysis and Mathematical Morphology*, Academic Press, London.

[Sirakov96] Sirakov N.M., 1996,Automatic Reconstruction of 3D Branching Objects, in Proc. 13th ICPR, TU Vienna, Austria, Pattern Recognition and Signal Analysis, IEEE Computer Society, Vol. II, 620-624.

[Sirakov94] Sirakov N.M., Muge F., Comparing 2D Borders Using Regular Structures, Proc. RecPad'94, Lisbon, Portugal, 169 -176.

[Sirakov] Sirakov N.M., Recognition of shape from a finite series of plane figures, in Shape in Pictures, Lie O., Toet A. Heijmans H.J.A.M., Foster D., Meer P. (eds.), Springer Verlag, the Netherlands, 453-462.

[Soille99] Soille P., 1999. *Morphological Image Analysis – Principles and Applications*, Springer, Berlin.

Sessão 5

ANIMAÇÃO E MULTIMÉDIA

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

ARENA and WOXBOT: Some Steps Towards the Animation of Cognitive Characters Acting in a Virtual World

Fábio Roberto Miranda João Eduardo Kögler Junior
SENAC College of Computer Science & Technology
Rua Galvão Bueno, 430 São Paulo SP Brasil
{fabio.rmiranda,kogler}@cei.sp.senac.br

Márcio Lobo Netto Emílio Del Moral Hernandez
Polytechnic School – University of São Paulo
Avenida Prof. Luciano Gualberto, Travessa 3, 158, São Paulo SP Brasil
{lobonett,emilio}@lsi.usp.br

Abstract
This paper reports some of the results of the WOXBOT / ARENA project on artificial life. This project to build virtual worlds was set initially aimed at the graphic simulation of an arena where WoxBots, small mobile robots, can perform requested tasks while behaving according to their own motivation and reasoning. Each robot is an intelligent agent that perceives the virtual environment via a simulated vision system and reacts moving away from or approaching to the object it sees. The conception and specification of the robots and the environment are being done very carefully to create an open distributed object architecture that can serve as a test bed freely available and ready to use for testing theories in some computational areas such as evolutionary computation, artificial life, pattern recognition, artificial intelligence, cognitive neurosciences and distributed objects architectures. Furthermore, it is a first step towards building a cognitive animated character.

Keywords
Artificial Life, Computer Cognitive Animation, Evolutionary Computation, Genetic Algorithms, Neural Networks.

1. INTRODUCTION

Artificial life is a term originally intended to mean the simulation of macroscopic behavioral aspects of living beings using microscopically simple components [Bonabeau95]. However, the term spanned to designate a wide variety of simulated living creatures, including virtual characters whose behavior emerges from hierarchically and functionally specialized complex structures like an animal's body [Terzopoulos98]. Both kinds of simulations are very interesting from the computational point of view: they offer very attractive means to computationally model complex behavior, a subject that has gained a special relevance under both the theoretical and the applied standpoints.

Artificial life worlds are computational simulations such as virtual spaces where animated characters interact with the environment and with other virtual beings of the same or of distinct categories. Different levels of sophistication can be found in these virtual creatures, from unicellular life with minimalist models to complex animals with detailed biomechanical models. However, all of them display behavior that emerges from the dynamics of a complex system (for instance, systems with learning, reasoning, ontologies, cognitive processes, etc).

New tendencies point to behavioral animation, where characters have some degree of autonomy to decide their actions. What we are aiming at is one of the most recent approaches to animation, cognitive animation [Funge99]. In this approach, artificial intelligence techniques are mixed with computer graphics to generate a computer animated character capable of acting in response to a storyboard or screenplay and displaying naturalism when doing other tasks not specified explicitly. Following this track to cognitive animation, we propose to exploit the capability of several methodologies and techniques that employ adaptive algorithms, evolutionary programming and artificial intelligence techniques.

Results obtained in a project set to build an artificial environment for animated virtual creatures (ARENA) are reported in this paper. Although it was conceived to allow more than one robot or virtual creature to be easily introduced to co-exist in the same environment, its first version presented here is a single-robot virtual world.

Our goal is to obtain virtual creatures capable of performing specified tasks in their environment by exploring certain strategies and adapting those whenever necessary. This scenario can be useful for many purposes: for behavior modeling, as a laboratory of learning algorithms, for research on societies of virtual characters, in the study of collective dynamics of populations, etc. The ARENA robot – WoxBot (Wide Open Extensible Robot) – has a vision system consisting of a simulated camera and a neural network that classify the visual patterns to provide input to a motor system controlled by a deterministic finite state machine (FSM). This FSM is an automaton obtained from an optimization procedure implemented with a genetic algorithm.

As an example of application, a given research project can be targeting visual recognition methods, so it will be using WoxBot as a subject employing the methods under test, and will take ARENA as a laboratory for evaluation experiments. Another research project instead, can target autonomous robot traveling strategies. In this case, the ARENA floor can be the field of traveling, and WoxBots will be simulating the autonomous vehicles. Yet the WoxBot could be re-shaped to have the aspect of a given car or truck model and the ARENA could be configured to resemble a street, in a project intended to analyze vehicle traffic conditions.

The ARENA is implemented as a distributed object environment and can run on single processor platform as well as can take advantage of parallelism or multithreading in high performance computer architectures. In this later case it could be used to handle highly sophisticated and complex applications, such as the simulation of a street with many cars. It could also be used to evaluate multi-agent distributed computational models using the WoxBots as the prototypical agents.

The organization of this paper is as follows: the next section gives an overview of the ARENA project, its requirements and choices for solutions. Sections 3 to 6 review several conceptual aspects involved in the various project components and give guidelines to be used in the implementation. Section 7 discusses particular implementation issues, such as the choice of Java and Java3D as programming language and 3D application programming interface. Section 8 addresses the modeling of the character and the environment and presents the results obtained. Section 9 foresees the further steps to be taken.

2. PROJECT OVERVIEW

Our long-term research line in artificial life is aimed at the development of complex virtual worlds with realistic creatures able to exhibit sophisticated behaviors with reasoning, learning and cognition. To achieve it, one is required to attend to the following aspects:

- Account to the wide variety of mathematical and computational models usually required to represent the complex aspects of living behavior;
- Provide an efficient environment and platform with enough computational power to represent in detail the behavior of the simulated creatures in real time, including the ability to support live interactivity with persons;
- Specify and build a system that embodies constant evolution and improvement in its constructs at the same time that enables or, moreover, promotes the reuse of well-succeeded results and implementations.

Our choice was to fulfill these requirements building an application development environment on top of an architecture of distributed communicating objects mapped on a microcomputer cluster [Bernal99]. Additionally, the ARENA implementation also exploits parallelism via multithreading, so the program can run both on multi and single processor platforms.

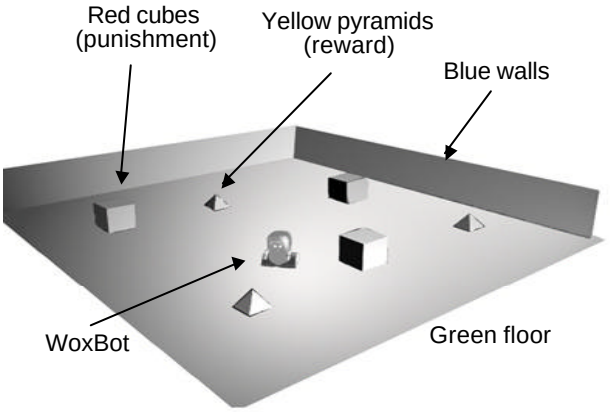


Figure 1: ARENA (Artificial Environment for Animats) – a virtual world for the development of artificial life projects. The character at the center is a robot that avoids the cubes and seeks the pyramids. A contact with a cube reduces the robot life, while the pyramids extends it. The robot goal is to prolong life as long as it can. An evolutionary computing scheme selects robots that perform better. The environment can support several robots.

The virtual space of the ARENA (figure 1) has a floor and walls encompassing the limits of the virtual world. Inside this scenario one can place objects of several kinds and functionalities such as obstacles, barriers, traps, shelters, energy or food sources etc. One or many characters can be introduced in this environment. In the first stage of the project the sole character will be the WoxBot (figure 2), whose task is to keep itself alive as long as it can.

Certain conceptual ingredients play specific roles on composing the artificial life scenario and thus should be present in the implementation: sensing and perception, knowledge use and evolution. In the following sections we examine these aspects.

3. ARTIFICIAL LIFE

Artificial life has been associated to computer science in different ways, from cellular automata theory [Adami98], [Bentley99] to computer graphics animation [Terzopoulos98], [Phillips88], [Badler92]. Some researchers have been involved with this field looking for models that would describe how real life began and evolved. In fact, they were looking for a universal life concept, which should be independent of the medium on which it existed [Adami98]. Other scientists were looking for physical models to give natural appearance to their characters. Very interesting results have been achieved through this approach [Terzopoulos98]. Although very different in purpose, these work have been related to evolution and natural selection concepts. In many of them, evolutionary computing schemes such as genetic algorithms have been an important tool to assist the conduction of transformations in the genotype of virtual creatures, allowing them to change from one generation to another. Mutation and combination (by reproduction) provide efficient ways to modify characteristics of a creature, as in real life. Selection plays a role choosing those that, by some criteria, are the best suited, and therefore are allowed to survive and reproduce themselves.

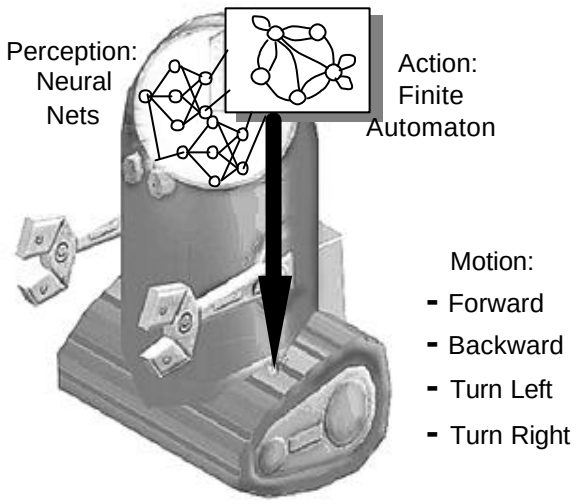


Figure 2: WoxBot (Wide Open Extensible Robot) – a virtual character that gets input from vision and reacts moving according to the decision taken by the finite state automaton.

Genetic algorithms are computational procedures capable of finding the optimal solution in particularly hard problems [Fogel95], [Holland92], [Koza92], [Mirchell97], [Beer90]. Each point of the optimization domain is represented by a binary string that is seen as "genetic information", in the sense that this string can be mutated, by flipping bits, as well as two such strings can be mixed by a crossover operation, in a way comparable to the actual biological reproduction. These strategies provide an efficient way to obtain global optimization in cases where it is very difficult, or not practical, to formalize the optimization problem on an analytical framework.

In this work the evolutionary computation is employed to generate the computer animated character control. Each WoxBot (Figure 2) is controlled by a finite state automaton, which evolves through generations, based on genetic mutations and crossover. Descendant robots can arise with some innovative features that may be better or worse than those from previous generations. A character being better or worse is a relative concept. In this project it is defined in a way correspondent to how well suited a character is to the environment. Those that by some criteria achieve higher grades (for instance keeping greater energy, living more time and performing their tasks faster) have higher chances to be selected for reproduction, and therefore to transfer the genotypes and, consequently, their features.

The use of knowledge in the artificial life environment can be done under two aspects: (i) it can be present in the environment and in the conception of the creatures and; (ii) it can be used by the creatures themselves when performing their actions. In both cases, the effective and rational use of the knowledge (about the world, the actions, the tasks) leads to what is called intelligent behavior. Intelligence can also be considered a property emergent from evolutionary systems [Fogel95]. Thus, the evolving characters can be modeled as intelligent agents performing tasks in the virtual environment.

4. INTELLIGENT AGENTS

Intelligent agents may be understood as computational entities that behave with autonomy in order to manipulate the information associated to its knowledge. This autonomy must regard the agent design, the environment, the goals and motivations [Russel95], [Beer90]. They have the capability of reasoning about what is going on in the environment where they are running, or of responding to some query conducted by external agents or persons.

In addition to artificial life features, our robots also exhibit features that allow their classification as intelligent agents. In this first implementation, while there is only one WoxBot in the Arena, they do not have communication, just sensing and mobility skills. And they have to decide which action they should take at every moment. A finite state machine has been evolved from random ones for this purpose. The robot should look for energy sources, while avoiding traps that make it weaker.

Based on visual information, as explained further, the robot plans its way to get closer to the energy sources. A finite state machine that can be a result of evolution through generations determines the actions of the plans.

Next versions of our robot will be able to communicate in an environment with other robots and they will implement also sociability skills, which are important features to promote a better adaptability from a creature to its environment, as in real life.

5. PATTERN RECOGNITION

One of the tasks that WoxBot has to perform in this first project is to be aware of nutrients (yellow pyramids) and hurting entities (red cubes) that are present in ARENA (figure 1). The ARENA scenario and robot vision are simulated by means of JAVA3D. The visual information is gathered from the 3D surrounding scene by projecting it to a view port with 3 color channels, namely R (red), G (green) and B (blue). Figure 3 depicts a typical input, which at the present state is taken in low resolution without any loss of accuracy, since no textures are yet used in our 3D virtual world simulation. To differentiate and spatially locate these entities, two specialized neural networks interpret the visual information received by the robot, one of them targeting nutrients and the other targeting hurting entities. These networks are named here ANN-I and ANN-II.

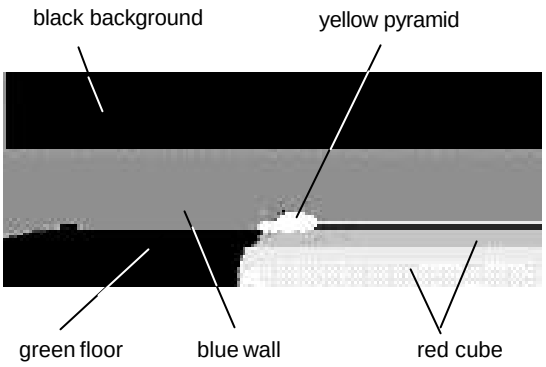


Figure 3: A typical image of the ARENA environment as seen by the WoxBot vision. The actual image has the depicted colors with shading resulting of the illumination. This image is the input of the neural nets that perform perception. It is also possible to add textures to the 3D scene and take yet more complex images.

The network ANN-I is trained for the identification of yellow pyramids as well as for a general evaluation (classification) of the position occupied by the principal yellow pyramid in the robot's visual field. To help in this task, a simple filter for the yellow color is implemented as a pre-processing stage. This is done by combining the primary sensory channels RGB so to obtain a gray scale where yellow is mapped into high values of gray and the remaining colors are mapped into low values of gray.

The monochromatic image obtained in this way is then used as the input to the ANN-I, the one that targets nutrients. After training, this network is able to activate one of 4 outputs (see Table I): output 1, indicating that no yellow pyramid is present in the visual field, output 2, indicating that the principal yellow prism is on the robot's left, output 3, when the principal yellow prism is on the center of the visual field, and output 4, when the principal yellow prism is on the right. In this way, ANN-I makes the robot aware of the presence and location of nutrients.

Code	scene situation
0	no object ahead
1	target object at left
2	target object at center
3	target object at right

Table 1: ANNI & II observers

In the same lines as ANN-I, the second neural network, ANN-II, performs a similar function for the identification and position evaluation of the red cubes representing the hurting entities. For this second network, however, the gray images that are sensed by the neural network inputs are obtained by using a different filter, which combines the channels RGB so to favor the red color.

The architecture chosen for ANN-I and ANN-II is the standard multi layer perceptron with one single hidden layer of moderate size and learning through the error back propagation algorithm.

Among the observations that we did during the training of the neural networks we want to mention that the detection and position classification of visual targets was facilitated by the use of samples (pyramids and cubes) of similar sizes. In other words, ANN-I and ANN-II don't have good abilities to implement scale invariance. That was in fact expected, since the complexity of the network is relatively limited, and its architecture does not have any specialized organization to implement scale invariance. This indicates that some kind of size invariance mechanism could be added in the pre-processing stage. Another possibility is to use a larger and more complex neural architecture so to make possible the interpretation of targets positioned at a broader range of distances.

An important issue to consider in the context of the neural networks of WoxBot is adaptability. In these initial experiments with the robot, the training of ANN-I and ANN-II was done in an isolated phase, which was performed previously to the exploration of ARENA. In other words, only after WoxBot was able to do the differentiation between nutrients and hurting entities, he started his exploration in ARENA, and then, no further adaptation of the visual recognition system was allowed.

In future phases of this project, we want the recognition system to be able to deal with unpredicted changing conditions such as change of illumination, and we will include the change of the features of the nutrients targeted by WoxBot (shape and color for example). Of course, this depends strongly on the ability of the visual recognition system to adapt, since the target recognition prototypes change with new experiences and new needs of the robot, as its life and exploration of ARENA evolves. At that point of the project, the intrinsic adaptive nature of neural networks will be crucial, and it will be fully explored.

6. ACTION, BEHAVIOR AND EVOLUTION

The WoxBot character is an intelligent agent with a simulated visual sensor to pick images of the environment from its point of observation. The two neural networks inside the agent analyze these images classifying the visual patterns. These networks outputs are tokens fed into the agent control system, which is a finite state machine (FSM). The inputs to the FSM generated by the combination of the ANN-I and ANN-II outputs are shown in Table 2.

input code to FSM	Semantics
0000	no YP and no RC
0001	no YP but RC at left
0010	no YP but RC at center
0011	no YP but RC at right
0100	YP at left but no RC
0101	YP at left and RC at left
0110	YP at left and RC at center
0111	YP at left and RC at right
1000	YP at center but no RC
1001	YP at center and RC at left
1010	YP at center and RC at center
1011	YP at center and RC at right
1100	YP at right but no RC
1101	YP at right and RC at left
1110	YP at right and RC at center
1111	YP at right and RC at right

Table 2: Input codes to the FSM and their meanings. YP = Yellow Pyramid and RC = Red Cube

Based on the above codes, the FSM chooses one action in its repertory: going ahead, turning right, turning left or going backwards. Table 3 shows the action encoding used to build the FSM representation as a genetic code.

code	Action
00	turn left
01	go straight ahead
10	turn right
11	go backwards

Table 3: Movement encoding

The motion control is performed by the FSM automaton, which is designed without specifying in advance what interstate transitions should occur. It is initially set with a random structure that is improved based on evolutionary computation concepts. Only the maximum number of allowed FSM states is specified. The FSM is represented by a string of bits coding its states, inputs and actions. This string is named the WoxBot chromosome. For each FSM state there is a chromosome section. All these sections have the same structure: for each of the 16 possible inputs shown on Table 2, there is an entry on the chromosome state section, composed by two fields: the first one is the number of the next state after the transition caused by the corresponding input; the second field is the code of the action, following Table 3, taken upon the given input. Figure 4 gives an example of a FSM and Figure 5 is the corresponding chromosome.

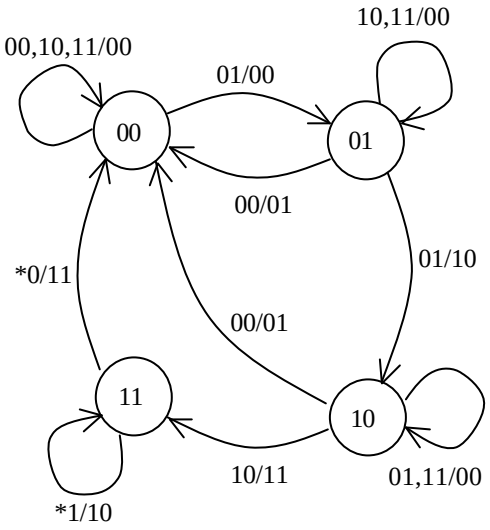


Figure 4: A sample Finite State Machine (FSM) used by WoxBot to control its reactions to the visual stimuli

At the start of the evolutionary process, a population of WoxBots is generated by sampling chromosomes at random from a uniform distribution. This population constitutes the first generation. Each member of the current generation is put into the ARENA, and it is assigned a performance value proportional to the duration of its life. A contact with red cubes shortens the life of WoxBots, while contact with yellow pyramids extends it.

State	In 00	In 01	In 10	In 11
00	00 00	01 00	00 00	00 00
01	00 01	10 10	01 00	01 00
10	00 01	10 00	11 11	10 00
11	00 11	11 10	00 11	11 10

Figure 5: Chromosome for the FSM of figure 4. There are 4 states. Each state has 4 inputs. Each Input has 2 fields, the first is the next state after the transition and the second is the action code.

After each generation of robots in the ARENA, the fittest ones are selected to be the parents of the next generation and their chromosomes are grouped in pairs. These chromosomes then undergo a mutation followed by crossover of each pair. This constitutes part of the next generation; the remnants are completed, in a heuristic method of adding variability, by drawing them from a uniform probability distribution, in the same fashion as it was done for the first generation. This procedure follows, giving thus more generations, until there is a stabilization of the fitness value of the population. Figure 6 shows schematically the whole evolutionary process.

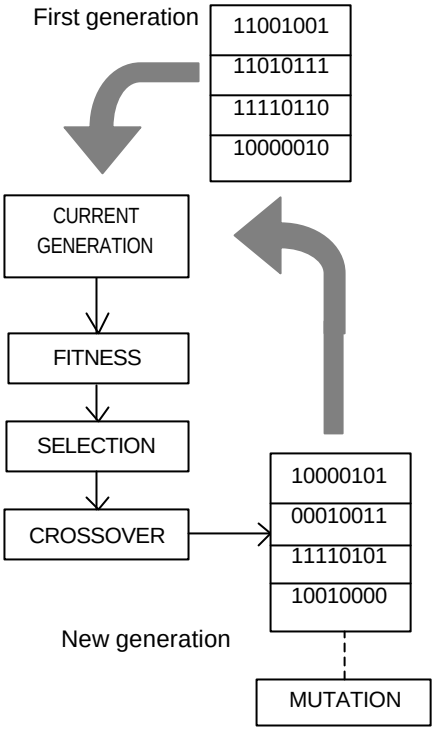


Figure 6: The evolutionary process. The first generation (top) undergoes simulation that results in fitness values. A crossover of the chromosomes of best individuals (WoxBots) yields a new generation, subject to mutation before simulation takes place again.

7. IMPLEMENTATION ISSUES

Before building an application intended for virtual world simulation, one key decision to be made is whether it will be a real time application or not. Many 3D animation packages today, like Maya™ or Softimage™, have an integrated Application Programming Interface (API), allowing algorithmic animation of characters. This approach, yet useful for some kinds of simulations, has the drawback of few or none capabilities for user interaction. These packages are also expensive, narrowing the scope of people that could later experiment with our work.

Having decided for real-time capability, the next issue is the choice of the API. Some requirements we decided to fulfill with ARENA included: portability through many platforms, share ability and open architecture, and easy of use for others. Given these issues, our choice was made for the Java Programming Language for development and Java3D as 3D graphics package. Java is a worldwide spread and freely available language strongly object-oriented. Its three-dimensional extension, Java3D, is available on a variety of platforms (Windows, Linux, Irix and others), is also freely available, and can be run through web browsers with the proper plug-in. The most powerful feature of Java3D is its scene-graph oriented approach that shifts the focus of 3D programming from vertexes to hierarchies of scene objects and events, shortening development time.

Additional advantages include the possibility of importing content authored in 3D modeling and animation packages, use of underlying OpenGL or DirectX accelerated hardware, automatic use of threads for executions, taking advantage of parallel hardware, and the availability of free Java Integrated Development Environments (IDEs), like Java Forté, from Sun Microsystems and JBuilder, from Inprise.

7.1 Application structure

Java3D organizes the objects represented in its scenes in a special kind of tree called *scene graph* [Sowizral98]. Such representation has the advantage of hierarchically organizing elements, what eases the programmer burden of implementing kinematically linked objects, such as a forearm linked to an arm, where a movement of the second implies a movement of the first. Another advantage relates to parallelism (branches of the tree potentially can be processed in parallel), and optimizations related to execution culling (whole branches of the hierarchy that aren't changed in certain ways can be optimized and made faster to process).

Some main objects where used at the ARENA, and are seen at figure 7. The objects VirtualUniverse and Locale operate as gateways to the 3D rendering engine. BranchGroup objects grant others access to the Locale and act as a grouping tool. TransformGroups represent grouped objects subject to transformations (translation, scaling and rotations).

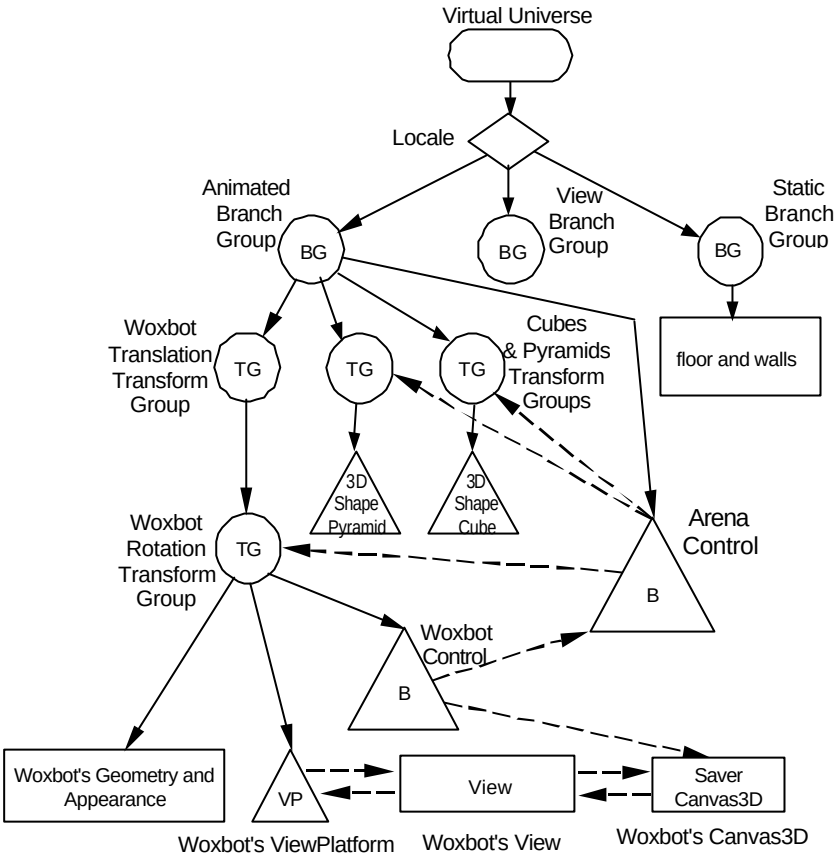


Figure 7: Java3D Scene Graph of ARENA with WoxBot. All objects linked to a VirtualUniverse through a Locale are taken into account by the rendering engine. BranchGroup (BG) objects serve as objects for grouping others. TransformGroups (TG) represents spatial transformations such as scaling, rotation and translation.

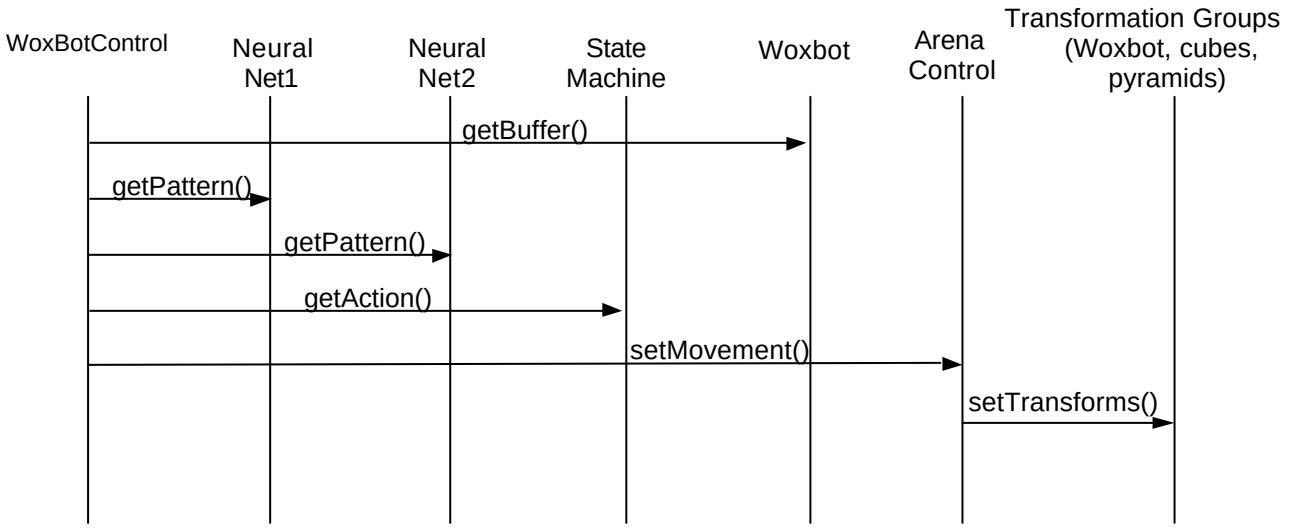


Figure 8: Sequence diagram of ARENA and WoxBot

The Java3D scene graph structure [Sowizral98] of the ARENA application is seen in figure 7. This application is composed of one Locale, with three BranchGroups directly attached to it. There is one BranchGroup for the user view of the scene (View BranchGroup), one BranchGroup holding the 3D objects in the scene that cannot move (floor and walls) and one BranchGroup holding the animated scene objects. The reason for this division is that Java3D performs a series of optimizations based on what a given BranchGroup can or cannot do, by holding all the 3D objects that cannot move in one BranchGroup, we can increase the efficiency of the optimizations. The Animated BranchGroup is where there is most to be observed in this application. The non-robot 3D objects (boxes and pyramids) have their own TransformGroup, the robot has one TransformGroup for rotation and another for translation. The WoxBot Rotation TransformGroup is the lastTransformGroup before the robot 3D mesh data, below it is also the ViewPlatform of the camera that renders the robot view (the WoxBot's View). The Animated BranchGroup also holds two control objects that are responsible for the dynamic aspect of the application: WoxBotControl and ArenaControl. WoxBotControl models the behavior of the robot, keeping track of visual data (accessing the WoxBot's Canvas3D), presents the data to the neural networks for pattern recognition and feeds the patterns into the state machine that decides robot's actions. Once robot's actions are defined, they're told the ArenaControl, that only lets valid actions take place (the robot cannot cross through walls or red cubes, for instance.

The diagram in figure 8 depicts in a simplified way the dynamic behavior of this program. An arrow from one bar to another means the first bar's object is requesting the second object to perform an operation. Return values are not represented. At extreme left, WoxBotControl can be seen. This object is responsible for most of the intricacies of the simulation. From time to time, this object polls the WoxBot Canvas3D for visual data. Then it processes the visual data and feeds the two neural networks, collecting pattern classification from them. Pattern data is then fed into state machine, which returns the action the robot should take. This desire of action is communicated to the Arena Control, that applies the results of this action to the scene. ArenaControl detects collision between the robot and other scene objects, decrementing robot's time of life when a collision is against a harmful object or incrementing it when collision is against yellow pyramids. In the latter case, the pyramids should disappear for a given amount of time, this behavior also is triggered by the ArenaControl.

8. SIMULATION AND RESULTS

The genetic algorithm was used to evolve a population of 16 individuals along 30 generations. The first generation was randomly created, every subsequent generation had the genotype of half of its members randomly generated and the other half created by a crossover of genotypes of the two most fit individuals of the previous generation, followed by a mutation. The crossover happened at a random point between the beginning and the end of the parents' chromosomes, exchanging complementary segments, so that resulting and original chromosomes have the same length. Mutation rate used in this simulation was 0.06, and it was a bit mutation rate, each bit copied from parents had a chance of 0.06 of being mutated.

A state machine with 4 states was used, which corresponds to a genotype of 258 bits (4 times 64 bits, the length of the description of each state transition / output relations, plus additional 2 bits to indicate the initial state of the machine).

The members of the generations are tested at the simulation environment one after another. In the simulation discussed in this paper each individual was given a time of 60 seconds, that could be extended or decreased on basis of the individual's behavior. Each time an individual collided against an yellow pyramid it received extra 4 seconds, that pyramid was then considered "eaten", and a new pyramid would appear at its place only after 4.5 seconds, this distribution of values was made to avoid seen behaviors where individuals has the tendency to stay at the same place indefinitely just waiting for the energy, this way they will end up dying once waiting for energy spends more energy that is gained when the waited energy arrives. The collision against a red cube caused the individual to loose 5 seconds, cubes also are considered "eaten" after the individual passes over it, and at the simulation discussed here those cubes took 3 seconds to reappear. By varying parameters such as values of the rewards and punishments related to passing over to the objects, and the time it takes the objects to reappear, we expect to see a range of strategies evolving from daring to conservative.

Figure 9 depicts the fitness plot along the generations. Some oscillation of the tendency is due to the low number of members in the populations, but the overall tendency to increasing and stabilization can be appreciated. After a long term running with bigger populations, the fitness will probably stabilize.

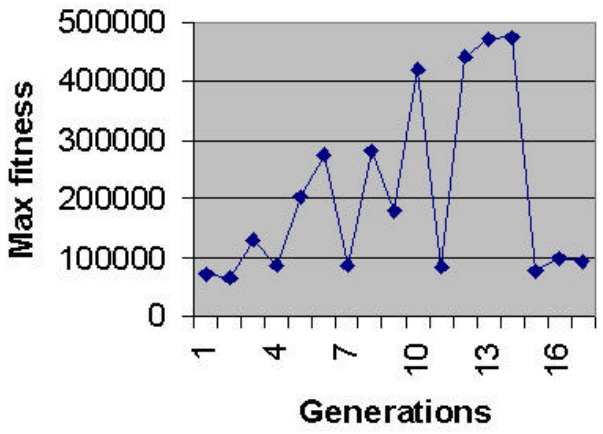


Figure 9: Fitness of the WoxBots along the evolutionary process of a population of 16 individuals.

At the present stage, the WoxBot still does not memorize the previous actions, so it actually cannot take advantage of learning. The number of states of the FSM is also low – a maximum of only 4 states is allowed. These limitations were adopted just to provide a reasonable chance for understanding the behavior of the machine by examining its structure.

Figure 10 presents a snapshot of the environment built in Java 3D to develop the ARENA / WoxBot project. Figure 11 shows the chromosome of the best-fitted individual in the above experimental run.

9. CONCLUSION AND FUTURE WORK

The first stage of the project was to build the ARENA with a single WoxBot to test the main idea and to orient the specification of the further steps. There are two main directions to follow simultaneously from now. One is to proceed with the ARENA development introducing more complexity in the environment, in the tasks and using two or more WoxBots simultaneously. The other is to improve the WoxBot to build new and more sophisticated characters.

The Jack [Adami98] project can be an inspirational experience for us, where different levels of research development and application and incremental development were successfully explored.

Concerning the actual stage of the simulations, WoxBot will have some improvements, namely to allow much more states in the FSM and to provide a memory for a finite number of previous actions. Also the ARENA will undergo some improvements concerning its size, the distribution of objects, the presence of textures and to allowance of simultaneous WoxBots. Also, concerning the WoxBots, given the use of textures and objects other than pyramids and cubes, improvements on its visual system will also have to be provided.

8. REFERENCES

[Bonabeau95] E.W. Bonabeau and G. Therulaz, Why do we need artificial life?. *Artificial Life an Overview*, edited by C.G. Langton, MIT Press, 1995.

[Terzopoulos98] D.Terzopoulos (org.), Artificial Life for Graphics, Animation, Multimedia and Virtual Reality. *SIGGRAPH 98 Course Notes 22*, 1998.

[Bernal99] Volnys Bernal *et al*, PAD Cluster: An Open Modular, and Low Cost High Performance Computing System. *Proceedings of 11th Symposium on Computer Architecture and High Performance Computing* Natal, RN.Brazil, Sep 29th-Oct 2nd , 1999, 215--222.

[Adami98] Adami, *Introduction to Artificial Life*, Springer Verlag, 1998.

[Phillips88] C.Phillips and N.I. Badler, Jack: A toolkit for manipulating articulated figures, *in ACM/SIGGRAPH Symposium on User Interface Software*, Banff, Canada, Oct. 1988.

[Badler92] N.I. Badler, C.B. Philips and B.L. Webber, *Simulating Humans: Computer Graphics, Animation and Control*, Oxford University Press, 1992.

[Bentley99] P. J. Bentley (ed.), *Evolutionary Design by Computers*, Morgan Kaufmann, 1999.

[Fogel95] D.B. Fogel, *Evolutionary Computation - Toward a new philosophy of machine intelligence*, IEEE Press, 1995.

[Holland92] J.H. Holland, *Adaptation in Natural and Artificial Systems*, MIT Press, 1992.

[Koza92] J. Koza, *Genetic Programming*, MIT Press, 1992.

[Russel95] S. Russell and P. Norvig, *Artificial Intelligence – A Modern Approach*, Prentice-Hall, 1995.

[Mirchell97] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, 1997.

[Beer90] R.D.Beer, H.J.Chiel and L.S.Sterling, A biological perspective on autonomous agent design, *in Designing Autonomous Agents*, P. Maes (ed.), MIT Press, 1990.

[Sowizral98] Henry Sowizral (org), "Inroduction to Programming in Java3D", SIGGRAPH 98, Course Notes 37,1998.

[Funge99] J. Funge, X.Tu, D. Terzopoulos. Cognitive Modeling: Knowledge, Reasoning and Planning for Intelligent Characters. *SIGGRAPH 99 Proceedings*, 1999.

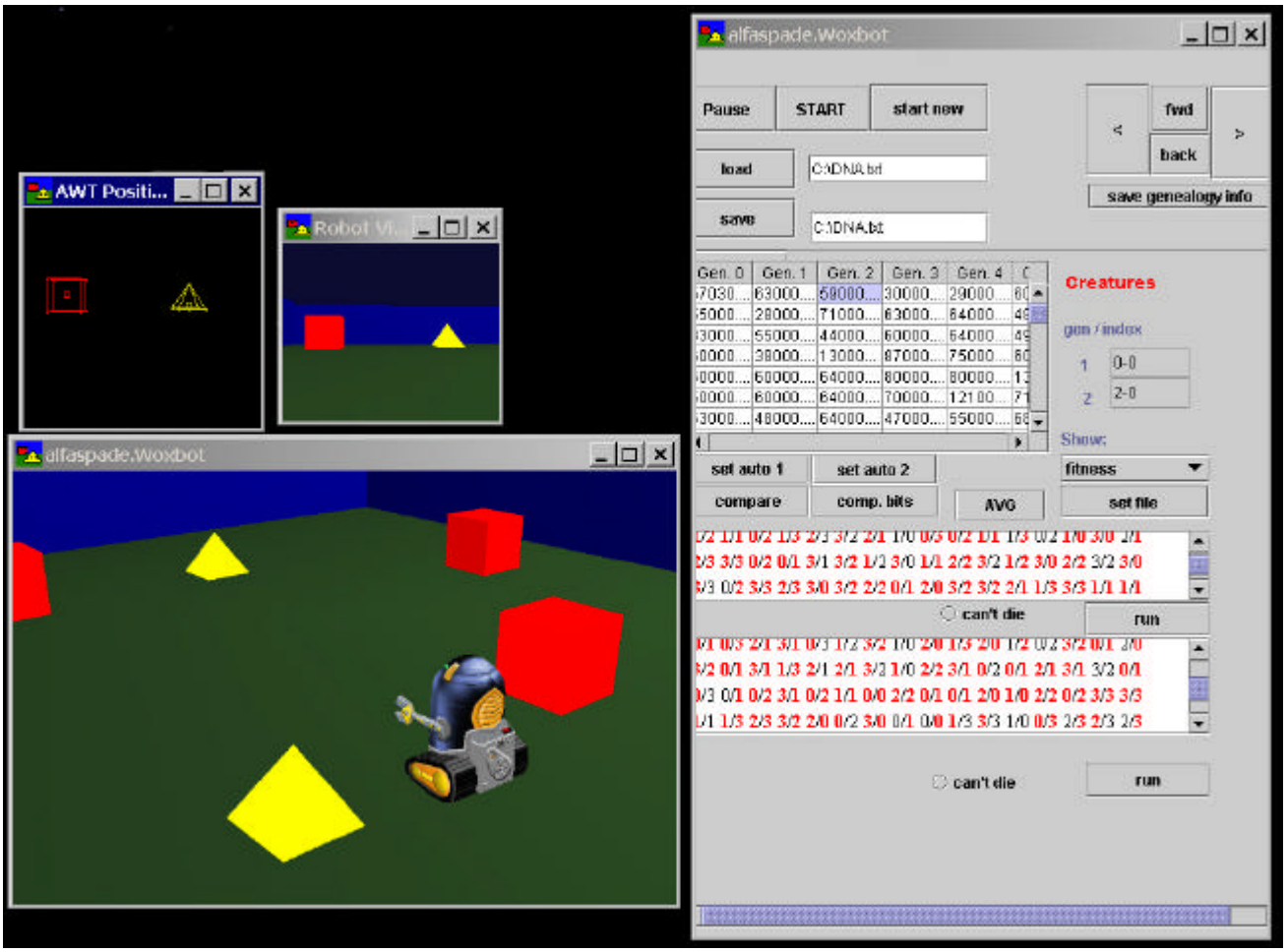


Figure 10: The Development desktop environment built with Java/Java3D. It allows the visualization of the evolutionary process live action, data logging for the population parameters, chromosomes, etc. It enables one to load a previously evolved individual and watch its performance again.

Initial state:3

3/2 3/0 0/3 1/0 0/3 3/3 2/1 3/3 2/3 3/0 3/2 3/1 2/0 3/1 3/1 0/2
2/2 0/3 2/3 3/0 0/3 0/1 3/0 0/3 1/0 2/2 1/0 1/1 0/2 3/2 1/0 1/3
1/0 2/3 2/2 0/3 3/1 1/0 2/0 1/0 1/3 1/0 0/0 1/1 3/2 0/0 3/2 0/0
0/1 2/0 2/3 3/3 3/2 3/1 3/3 2/1 0/0 1/0 1/0 0/0 0/2 1/3 1/1 3/2

Figure 11: Chromosome of the best fitted individual of the experiments run with the specifications provided in the text.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Estratégias de Desenvolvimento de Jogos Multimédia

Pedro Faria Lopes
ISCTE / ADETTI
Av. Forças Armadas, 1600 Lisboa
Pedro.Lopes@iscte.pt

Maria Vasconcelos Moreira
ADETTI
Av. Forças Armadas, 1600 Lisboa
Maria.Moreira@iscte.pt

Hugo Pereira
ADETTI
Av. Forças Armadas, 1600 Lisboa
Hugo.Pereira@iscte.pt

Sumário
Com a crescente necessidade de desenvolver projectos multimédia de maior qualidade, em tempos de concretização menores, há que definir critérios que permitam alcançar estes objectivos. Neste trabalho descrevem-se alguns dos critérios encontrados, tanto pelo estudo e análise de projectos já existentes, como pela nossa experiência na concretização de dois projectos, um já concluído e outro em desenvolvimento, na área dos jogos multimédia.

Palavras-chave
Multimédia, vídeo digital, design, animação por computador, tecnologia, técnica.

1. INTRODUÇÃO

Da análise e estudo de um conjunto significativo de produtos multimédia em diferentes áreas de aplicação (enciclopédias, *marketing* empresarial, diferentes tipos e estilos de jogos, material didáctico e outras) [FPublishing98, FPublishing 99, Domain99, IBM95, CinéLive99, Origin97, Sierra96, Gabriel96, Gabriel98] verifica-se que, numa grande maioria dos casos, o nível de qualidade alcançado não é coerente com o nível tecnológico disponível, ie, os diferentes componentes podem apresentar características de base com qualidade (boa definição de imagem, fotografias suportadas em mais de 16 milhões de cores, som com altas frequências de amostragem) mas a sua inter-relação e gestão é realizada de tal forma que o produto final se apresenta invariavelmente falho de coerência e globalmente com baixa qualidade.

Se se efectuar uma análise em retrospectiva pode concluir-se que se passa hoje na área do Multimédia o que, há uma década atrás, se verificava na área de Animação por Computador: à facilidade de acesso e melhor qualidade das ferramentas de suporte à criação e desenvolvimento de filmes em animação por computador, não passou a corresponder uma melhoria significativa dos conteúdos dos filmes. Pelo contrário, John Lasseter, realizador da Pixar e reconhecido internacionalmente depois do sucesso da primeira longa metragem em animação por computador, *Toy Story* [Lasseter95], refere explicitamente em 1987 [Lasseter87] que à melhoria e facilidade de utilização das ferramentas, com a possibilidade de melhoria da qualidade da animação, corresponde também um acréscimo de maior quantidade e pior qualidade de animação por computador: aceder e dominar as ferramentas é confundido com o domínio da tecnologia e da técnica.

Hoje em dia o acesso à tecnologia Multimédia é generalizado. Também o acesso à escrita, com a tecnologia tradicional do papel e do lápis, é generalizada desde há muito. No entanto poucos são os que pretendem vender o resultado da sua escrita. Relativamente ao acesso a câmaras de vídeo, os equipamentos são cada vez mais baratos e acessíveis. No entanto poucos são os que fazem ou vendem filmes, havendo claramente a distinção entre filmagens amadoras e o circuito profissional do audiovisual. Dados estes dois exemplos, escrita e audiovisual, faça-se a comparação com o que se passa em Multimédia. É hoje prática corrente que o acesso e domínio de alguma tecnologia multimédia significa ser-se capaz de desenvolver conteúdos multimédia; tal como há uma década se praticava em animação por computador. É esta, em nossa opinião, a razão da má qualidade da generalidade dos produtos multimédia existentes. Na realidade, acesso à tecnologia não deve ser confundido com o domínio da técnica, que implica estudar e desenvolver metodologias e modelos: é com técnica e metodologia que a qualidade dos conteúdos, quaisquer que sejam, pode ser construída, seja na escrita, no audiovisual, em animação por computador ou em multimédia.

1.1 Algumas questões em análise

Da análise dos produtos atrás referidos concluiu-se um conjunto de ilações que se podem resumir nos seguintes pontos [Williams99, Lopes99, Lopes01a, Lopes01b]:

- o vídeo digital em multimédia é aplicado mais como um gadget do que com propósito;
- os diferentes elementos multimédia são agrupados em operações de "juntar", sem preocupações de correlação entre os diferentes componentes;

- utilizam-se textos longos sendo frequente a necessidade de seleccionar um cursor deslizante já que o espaço disponível não é suficiente para fazer aparecer o texto completo;
- o texto utilizado nem sempre é apresentado da forma mais legível, seja por má escolha do tipo de letra, seja por ser apresentado com efeitos visuais que dificultam a leitura (letras texturadas, efeitos de transparência, etc.);
- as imagens, textos, gráficos ou fotografias são geradas sem a preocupação de criar um conjunto coerente e legível de cores;
- utilizam-se sons, animações e vídeos sem evidente valor acrescentado.

Em dois exemplos ao acaso, meramente como factor ilustrativo, encontrou-se o seguinte:

- CD "museu": músicas de ambiente agressivas e sons de interacção muito altos quando, em termos das edições tradicionais em catálogo, os conteúdos são referência de equilíbrio, bom gosto e informação bem tratada;
- CD para PMEs: vídeo de apresentação com um executivo, de gravata e penteado, filmado com o Marquês de Pombal em fundo, em hora de ponta e bastante vento.

Pode resumir-se, sem grande escala de exagero, que o domínio da tecnologia se sobrepõe aos factores de bom senso impondo "folclore de inadequações" [Lopes01a].

Desta análise salvaguardam-se excepções, por exemplo *O Dicionário Mágico* [Cibel96], entre outros. Destaque particular ainda para *Eve* [Gabriel96] e *Cerimony of Innocence* [Gabriel97] onde é bastante claro que a base tecnológica é posta ao serviço do conteúdo, ie, as soluções são encontradas para servir uma narrativa e uma mecânica de jogo que envolvam e cativem o jogador.

Nas próximas secções descrevemos alguns dos pontos mais relevantes no estabelecimento de uma estratégia de desenvolvimento de multimédia de qualidade, no contexto de investigação e desenvolvimento que temos vindo a realizar, não só no estudo de material de terceiros, como também validando os princípios estabelecidos na realização de projectos concretos, caso dos jogos *Eco-Pontos* [CML00] e *Eco-Media* [Lopes01c].

2. MULTIMÉDIA E NARRATIVA

Como primeiro ponto saliente-se que, por comparação com outras áreas de expressão (romances, teatro, cinema, ...), um produto multimédia deve começar por desenvolver-se em torno de uma estória. Qualquer que ela seja. Quer se trate de um produto de marketing, de apresentação empresarial, publicitário, jogo ou qualquer outro. A estrutura deve ter um início, um meio e um fim, bem definidos e claros: uma apresentação, um desenvolvimento e uma conclusão.

Como segundo ponto saliente-se que todos os elementos (sons, textos, gráficos, imagens, animações, vídeos) devem ser realizados função da estória: nenhum deve existir sem que o seu peso relativo seja estudado e claramente evidenciadas as suas funções como uma parte do todo, em termos de valor de conteúdo parcial e global.

Em terceiro lugar saliente-se que deve ser criado um documento de referência onde tudo o que diz respeito ao projecto é descrito no máximo detalhe possível. Este documento será a base de trabalho e deve ser "congelado" a partir do momento em que descreva o projecto na sua globalidade e cada uma das componentes em detalhe: depois de "congelado" não devem ser admissíveis alterações de conteúdo, a não ser de pormenor e pontuais, que não comprometam o projecto na globalidade.

Com base no documento de referência deve estabelecer-se as áreas de conhecimento envolvidas (composição musical, sonoplastia, grafismo, design, interfaces, psicologia, animação, vídeo e programação), determinar o grau de importância de cada uma, identificar os intervenientes e estabelecer uma escala e cadeia de autoridade sujeita ao gestor de projecto.

Esta aproximação tem sido seguida no nosso grupo de investigação, onde os conhecimentos em multimédia, ligados a áreas como desenvolvimento de estórias, ciências da educação, psicologia, animação, vídeo e som, têm sido orientadas no sentido de criar ambientes de jogos para aprendizagem que sejam apelativos para as audiências alvo através de ferramentas úteis e utilizáveis [CML97, Lopes01c].

3. DESIGN E MULTIMÉDIA

No contexto desta secção deve entender-se design como uma forma alargada de conceptualização, não apenas como uma abordagem clássica de design. Por outras palavras, concepção numa escala multimédia, tendo em conta os componentes múltiplos que fazem e constroem um produto multimédia.

Nas secções seguintes apresentam-se alguns exemplos do design visto nesta escala multidimensional, ou multicomponente, ie, onde cada elemento (imagem, vídeo, som, ...) é sempre analisado em função das implicações intrínsecas e extrínsecas, relativamente a si e aos elementos com que potencialmente se relaciona.

3.1 Grafismo e animação

Na Figura 1 está representado o primeiro estudo gráfico realizado no âmbito do projecto *Eco-Pontos*, um jogo multimédia para o ensino de comportamentos ecologicamente correctos, realizado para a Câmara Municipal de Lisboa [Lopes99].

Da análise da imagem pode verificar-se a sua estrutura gráfica, baseada em elementos desenhos com um traço de contorno substantivamente largo. Este é um tipo de grafismo fora do comum, apelativo pelas suas características visuais. Se considerado apenas isoladamente, por exemplo se se destinasse a um livro impresso, seria

um grafismo a preservar. No entanto, no contexto do projecto, teve que sofrer modificações no sentido de se coadunar com os outros componentes, em particular a componente de animação.

Pela descrição do documento de referência, alguns elementos do jogo seriam animados. Manter um grafismo como o proposto na Figura 1 seria posteriormente encontrar fortes problemas na animação: a qualidade e presença do traço de contorno é tão importante que teria que ser considerado como mais um elemento da animação; a animação teria que se preocupar não só com os movimentos específicos de cada parte de cada objecto animado como também com a animação explícita do próprio traço. Este tipo de animação seria difícil de concretizar, morosa e dispendiosa em termos financeiros.

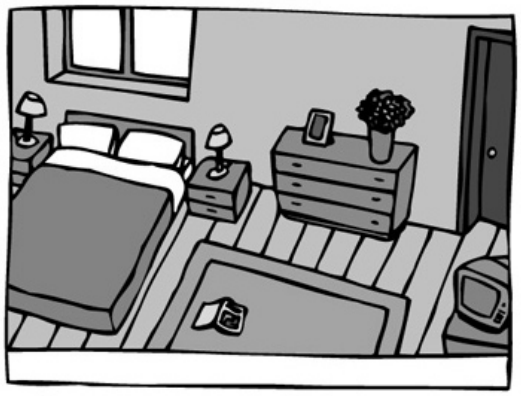


Figura 1. Estudo preliminar, "quarto dos pais".

Figura 2. Resultado final, "quarto dos pais".

Seria inadmissível criar os objectos a animar com um traço mais fino para facilitar a animação já que a coerência global do grafismo se perderia. Pelo que a solução foi reduzir a importância relativa do traço, resultando na imagem apresentada na Figura 2. Este estilo de design foi "congelado" e adoptado em todo o projecto.

Este é um exemplo das implicações "dinâmicas" das diferentes áreas. Caso não existisse a componente de animação no projecto, o design original, mais interessante, poderia ter sido preservado. Mas para manter o projecto equilibrado sobre todos os pontos de vista, ie, graficamente coerente, economicamente viável e controlado em termos de evolução temporal, o grafismo em termos de evolução temporal, o grafismo original teve que ser adaptado.

3.2 Área de imagem e som

Na Figura 3 representa-se a imagem da sala de jantar no jogo *Eco-Pontos*. Um dos elementos principais da sala é a televisão onde podem ser mostrados vídeos associados a cada uma das cassetes vídeo que estão no chão da sala.

O design da televisão teve em linha de conta múltiplos factores. Para mostrar os vídeos seria necessário aumentar a televisão? Com que implicações em termos de equilíbrio do grafismo e das acções paralelas que o jogo tinha que suportar dinamicamente?

Figura 3. Imagem da sala.

O documento de referência indicava que, enquanto decorre um vídeo, o jogador podia interagir com o cenário, manipular os objectos, inclusivamente sair do cenário. Aumentar a televisão implicava criar um cenário novo com todos os componentes restantes aumentados, complicando a gestão dos objectos manipuláveis (jornal, revistas, garrafa, latas, cassetes) e das restantes acções.

Figura 4. "Vidrões da Estrela" na televisão.

Em termos da área de visibilidade para os vídeos jogou-se a dois níveis para manter o cenário de base não alterado: criar uma televisão um pouco maior que o habitual, sem entrar em exagero, de modo a maximizar a área disponível para a imagem dos vídeos; jogar com a componente identificativa da faixa áudio dos vídeos que seriam passados. Os vídeos são uma série de animações do Vasco Santana a defender a reciclagem e os comportamentos ecologicamente correctos; fazem parte de uma

campanha televisiva que a CML desenvolveu e que teve um forte impacte na população. A memória auditiva serve aqui como factor identificativo da mensagem não sendo portanto necessário introduzir todas as complicações adicionais do aumento da televisão. Pela Figura 4 não é evidente que o vídeo a passar é o "Vidrões da Estrela" mas com a componente auditiva há uma identificação correcta e imediata.

Figura 5. Vídeo de imagem real inserido em jogo.

No caso de os vídeo serem de outra natureza, por exemplo de imagem real, a correlação deste aspectos teria que ser feita a outro nível para manter o mesmo tamanho relativo da televisão na sala: em vez de jogar com planos de imagem "aberta" (ditos planos gerais) pode jogar-se com planos de imagem "ampliada" (ditos grandes planos) de modo a aumentar a caracterização e identificação visual, tal como se exemplifica na Figura 4 onde se inseriu um vídeo de imagem real durante uma apresentação ao vivo [Lopes01a] do jogo *Eco-Pontos*.

3.3 Espaço de texto

É fundamental minimizar a utilização de texto já que é pouco legível em ecrã. Utiliza-se em que situações? E em que disposição relativa?

Figura 6. Texto inserido em ecrã navegável.

Se há espaço para imagens, animações e sons, deve deixar-se que esses elementos "falem" por si: "uma imagem vale mais que 1000 palavras", se a deixarmos falar; se for "afogada" em texto perde a eficácia. Um texto é usável em complemento se, ou quando, a imagem não é suficiente. E deve estar restrito numa zona bem identificada e delimitada, devendo ser autónoma e completamente visível, sem necessidade de "navegação" dentro do texto.

A Figura 6 mostra um dos ecrãs do "ABC do Ambiente", um conjunto de fichas do jogo *Eco-Pontos* com informação adicional sobre o conteúdo do jogo. A área de texto foi previamente definida de modo a privilegiar a área gráfica. Todo o texto, em todas as fichas, foi sintetizado de modo a caber na área de texto, sempre com o mesmo tipo e tamanho de letra.

3.4 Coerência visual e de modelo

Ao longo do jogo há alguns equipamentos (sofá, televisão, frigorífico, máquina da roupa) que se avariam para lá do recuperável. A estes esquipamentos a CML atribui o nome de *monstros* e o procedimento correcto é telefonar para a CML para chamar a camioneta que os recolherá gratuitamente para evitar que fiquem na rua a fazer lixo. Na Figura 7 representa-se o sofá que se avaria.

Em diversas instâncias da literatura informativa da CML estes monstros estão representados antropomorficamente, com um aspecto caricatural. No contexto do jogo nunca acontece uma abordagem caricatural: sendo graficamente "desenhado", tal não é incoerente ou incompatível com um modelo realista, sendo que todas as acções são também de carácter realista.

Figura 7. Sofá estraga-se, nasce um "monstro".

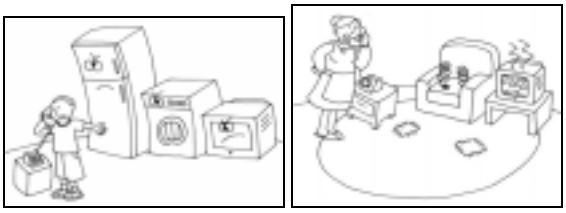


Figura 8. Ensaio de monstros para ABC do Ambiente.

Para as fichas do ABC do Ambiente relativas à descrição dos monstros, fizeram-se dois ensaios, um caricatural, outro realista (Figura 8). Verificou-se que utilizar o elemento em caricatura era incoerente com toda a realidade do jogo onde nunca se apela à caricatura. Pelo que se manteve a coerência gráfica e de modelo realista, seleccionando a imagem onde os "monstros" são objectos estragados e não aparelhos antropomórficos.

3.5 Coerência técnica

Ao longo de todo o projecto houve a preocupação de manter um rigor técnico elevado no que respeita à realidade que o jogo suporta: pequenos detalhes aparentes foram pensados e/ou corrigidos de modo a que a mensagem global dos comportamentos ecologicamente correctos saísse reforçada.

Na Figura 9 mostra-se um estudo para a Ficha 1 do ABC do Ambiente. Neste estudo aparecem duas pessoas a transportar compras, uma com uma cesta, outra com sacos de plástico, representando desperdício. A idade é a de pessoas não jovens.

Figura 9. Estudo para Ficha 1 do ABC do Ambiente.

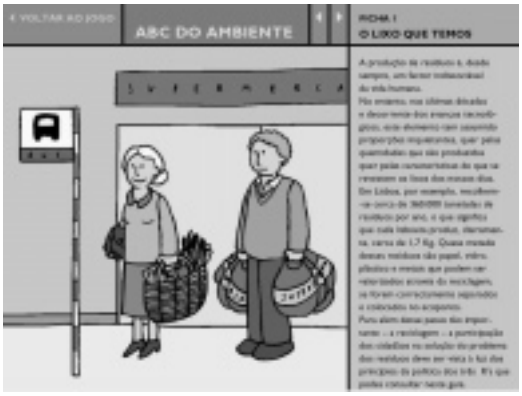


Figura 10. Coerência técnica na informação visual.

A Ficha final, Figura 10, foi modificada de modo a retratar um jovem que transporta os sacos de plástico já que, do ponto de vista da CML, tecnicamente verifica-se que são os jovens que denotam mais comportamentos anti-ecológicos e como tal, em rigor, a imagem da Figura 9 era tecnicamente menos correcta.

4. CONCLUSÕES

O desenvolvimento de multimédia de qualidade obriga a uma abordagem integrada e equilibrada das diferentes componentes que compoem um projecto. A integração de componentes e elementos não pode ser vista como uma fase de "juntar", tem que ser coerentemente vista com apelo ao interrogar do peso e importância relativa de cada uma das partes.

Desenvolver multimédia nesta perspectiva implica um trabalho de estudo e interrogação do valor acrescentado dos componentes de modo a garantir que a soma das partes produz um resultado melhor e superior à simples soma aritmética dos valores individuais.

Pela análise de trabalhos já produzidos e pela nossa própria experiência, enumerámos alguns tópicos de contributo para a criação de referências de boa prática em multimédia digital de qualidade. Os resultados alcançados mostram que é possível e praticável criar multimédia de qualidade.

A apresentação ao vivo dos projectos reforça o conteúdo deste trabalho, onde as componentes descritivas e ilustradas são, naturalmente, menos fortes que uma exploração real dos projectos.

O projecto *Eco-Pontos* foi financiado pela Câmara Municipal de Lisboa ao abrigo de um protocolo de colaboração entre a ADETTI e a CML.

O projecto *Eco-Media* é co-financiado pela Fundação para a Ciência e Tecnologia (FCT), no âmbito do programa SAPIENS 99, e pelo Programa Operacional (POSI), do Quadro Comunitário de Apoio III (2000-2006), comparticipado pelo fundo comunitário europeu FEDER, projecto número POSI/1999/CHS/34676.

5. AGRADECIMENTOS

À Maria Ferrand e ao Alexandre Matos por terem aceite envolver-se no projecto *Eco-Pontos* e por todo o empenho, profissionalismo e entusiasmo que trouxeram.

A todos os participantes, individuais e empresariais, no desenvolvimento do jogo *Eco-Pontos*, encontrando-se listada na ficha técnica do jogo a responsabilidade de cada um.

À Susana Lopes e à Sofia Pinto pela participação no projecto *Eco-Media*.

6. REFERÊNCIAS

[Adobe00] Adobe Premiere, http://www.adobe.com, 2000.

[Cibel96] *O Dicionário Mágico*, jogo multimédia, Cibel Lda, 1996.

[CinéLive99] CinéLive Magazine et CD-ROM, CinéLive, April issue, 1999.

[CML00] ***Eco-Pontos***, jogo multimédia, Departamento de Higiene Urbana e Resíduos Sólidos, Divisão de Sensibilização e Educação Sanitária, Câmara Municipal de Lisboa, Portugal; Produtor executivo e director: P.F. Lopes, Dezembro 2000.

[CML97] "ECO VASCO & Companhia (uma aventura ecológica)", Departamento de Higiene Urbana e Resíduos Sólidos, Divisão de Sensibilização e Educação Sanitária, CML, 1997.

[Compesi90] R. J. Compesi, R. E. Sherriffs, *Small Format Television Production*, 2nd Edition, 1990.

[Domain99] Games Domain website 1999 (Available on July, 1999 at http://www.gamesdomain.com).

[FPublishing98] PC Format, magazine issues from March to June 1998, Future Publishing, 1998.

[FPublishing99] PC Game, Future Publishing, 1999 (Available on July, 1999 at http://www.pcgame.com).

[Gabriel96] P. Gabriel, *Eve*, computer game, Real World, 1996.

[Gabriel98] P. Gabriel, *Cerimony of Innocense*, computer game, Real World, 1998.

[Gonçalves00] A.P. Gonçalves, *Planning and Development of a CD-ROM for Teaching Multimedia*, MSc Thesis, Instituto Superior de Ciências do Trabalho e da Empresa, 2000.

[Hecker00] C. Hecker, "Physics in Computer Games", COMMUNICATIONS of the ACM, 43 (7), July 2000.

[Hori00] H. Hori, TMPGEnc, ver beta12a, (0.11.20.98), (as available on May, 2000 at http://www.yks.ne.jp/~hori/TMPGEnc.html), 2000.

[IBM95] *ThinkPad MPEG Sampler*, CD-ROM, Interactive Media, IBM, 1995.

[Lasseter87] J. Lasseter, "Principles of Traditional Animation Applied to 3D Computer Animation", *Computer Graphics*, Vol. 21, no. 4, July 1987, pp. 35-44.

[Lasseter95] J. Lasseter, *Toy Story*, Pixar e Walt Disney, filme de animação por computador de longa metragem, 1995.

[Lopes99] P.F Lopes, M.V. Moreira, M.L. Duarte, "EcoVasco, an Ecological Multimedia Adventure ", EUROGRAPHICS Multimedia 99 Workshop, Milan, Italy, September 1999.

[Lopes00] P.F. Lopes, A. Muge, M.V. Moreira, "Taco a Taco", 9º Encontro Português de Computação Gráfica, Marinha Grande, Fevereiro de 2000.

[Lopes01a] P.F. Lopes, "Multimédia: rápido, divertido, fácil e barato", Seminário de Multimédia e Computação Gráfica XXI - O Futuro, organização do GPCG e APDC, Multimédia XXI, Feira Internacional de Lisboa, 4 de Maio, 2001 (comunicação convidada).

[Lopes01b] P.F. Lopes, M.V. Moreira, A.P. Gonçalves, "A New Computer Game Approach for Multimedia Digital Video Reuse", International Conference on Media Futures, Florence, Italy, 8-9 May, 2001.

[Lopes01c] P.F. Lopes, M.V. Moreira, et al., *EcoMedia*, jogo multimédia, projecto co-financiado pelo programa Sapiens99 da Fundação para a Cienência e Tecnologia, projecto nº POSI/1999/CHS/34676.

[MacroMedia00] Director 8, MacroMedia, http://www.macromedia.com, 2000.

[MainConcept00-11] MainConcept, http://www.mainconcept.de, 2000.

[Optibase98] MPEG MovieMaker, Real-time high quality MPEG-1 encoding board, Optibase, 1998.

[Origin97] ***Wing Commander, The Prophecy***, computer game, Origin, 1997.

[Rice96] Rice, F.P., *Child and Adolescent Development*, Prentice Hall, September 1996.

[Sierra96] ***Phantasmagoria: a puzzle of flesh***, computer game, Sierra; Executive producer: K. Williams, 1996.

[Williams99] J. Williams, A. Lock, C. Burnett, "Digital Video for Multimedia: Considerations for Capture, Use and Delivery", Multimedia Resources Unit, University of Bristol, 1996 (Available on May, 1999 http://www.man.ac.uk/MVC/SIMA/digital_video/title.html).

Sessão 6

AMBIENTES VIRTUAIS

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

A Realidade Virtual como Ferramenta de Treino para Montagem de Cablagens Eléctricas

Luís Grave Cristina Escaleira António F. Silva
Centro de Computação Gráfica
Coimbra / Guimarães
{Luis.Grave, Cristina.Escaleira, ASilva}@ccg.pt

Adérito Fernandes Marcos
Centro de Computação Gráfica/Universidade do Minho, Dep. de Sistemas de Informação
Coimbra / Guimarães
Aderito.Marcos@ccg.pt / marcos@dsi.uminho.pt

Sumário

A realidade virtual (RV) é aplicada em diferentes áreas de interesse, entre as quais se insere a indústria automóvel. A capacidade de RV em criar simulações realísticas de dados, objectos e ambientes, associado à possibilidade de interagir e manipular de uma forma intuitiva com os mesmos, tem sido um chamariz neste domínio, abrindo oportunidades de aplicação em diferentes sectores dos quais a prototipagem, a simulação de situações de risco, a visualização e o treino de processos fazem parte.

Este artigo foca em especial uma aplicação de RV direccionada para o ensino e treino de operadores de montagem na indústria de cablagens eléctricas para a indústria automóvel.

O artigo começa por introduzir o tema de uma forma genérica apresentando alguns conceitos sobre ambientes virtuais, ensino e treino que servirá de apoio à descrição do problema e dos objectivos. Depois é apresentado o sistema de realidade virtual começando por referir os requisitos necessários para a sua elaboração e expondo em seguida a estrutura, a modelação geométrica e a configuração do mesmo. O artigo continua, apresentando as metáforas de interacção utilizadas, seguido de uma exposição sobre o método adoptado para testar e validar a ferramenta em ambiente laboratorial e fabril. Neste ponto é apresentado uma análise dos resultados obtidos. Finalmente, são tecidas as conclusões e indicado novos caminhos a percorrer nesta área de conhecimento.

Palavras-chave

Ambientes virtuais, treino, ensino, metáforas de interacção, realidade virtual.

1. INTRODUÇÃO

A necessidade das empresas introduzirem soluções inovadoras que permitam melhorar o desempenho das suas operações é cada vez maior dada a competitividade a que estas estão sujeitas. A utilização eficaz das potencialidades que a tecnologia oferece constitui um objectivo base em cada empresa, independentemente do sector de actividade a que pertence. Actualmente a indústria automóvel preocupa-se com o ensino e treino no processo de produção, por forma a minorar os tempos de aprendizagem associados ao mesmo.

Desta forma, as empresas investem em pesquisa e investigação de ferramentas tecnológicas capazes de optimizar e simplificar o ensino e treino dos processos de produção e consequentemente diminuir o seu tempo e custos associados. Neste cenário, a realidade virtual surge como uma das várias apostas tecnológicas identificadas pela indústria automóvel.

A indústria de cablagens eléctricas para automóveis sente particularmente este problema devido à grande dependência das suas linhas de montagem da mão-de-obra qualificada [Grave 2000]. Com efeito, cada linha de montagem de uma cablagem necessita, para cada turno, de cerca de trinta operários especializados. Esta necessidade agrava-se devido à formação que cada operário tem forçosamente de receber antes de ser integrado nas linhas de montagem. A grande diversidade de cablagens em produção simultânea numa fábrica impossibilita a existência de simuladores físicos das linhas de montagem, o que obriga os formadores a colocar os operários nas linhas de montagem depois de terem recebido uma formação quase exclusivamente teórica.

A presença, na linha de montagem, destes operários diminui, quer a velocidade de montagem da linha, quer a qualidade do produto final.

Neste contexto, a realidade virtual aparece como uma possível solução para resolver os vários problemas de formação, nomeadamente:

- Facilidade em modelar novas cablagens, ou conjuntos de cablagens;
- Não desperdício de material;
- Possibilidade de treino repetido de uma operação, ou um conjunto de operações com o intuito de melhorar o desempenho;
- Por último, e talvez mais importante, cobre os principais aspectos da formação prática: localização dos objectos (memória espacial), gestos inerentes à montagem (memória gestual) [Mine 1997] e sequência de operações a realizar.

A memória espacial consiste em relembrar onde um objecto se encontrava e a memória gestual em recordar todos os gestos e esforço gasto nos mesmos para produzir uma determinada acção ou sequência de acções.

A realidade virtual, através de técnicas de imersão e manipulação directa de objectos, consegue transmitir ao utilizador a sensação de um espaço físico (virtual) que o rodeia e com o qual ele interage, contribuindo para a memorização do seu futuro posto de trabalho [Sá 1999].

No entanto, a grande maioria de aplicações existentes que usam tecnologias de realidade virtual, estão orientadas para serem utilizadas por especialistas que dispendem muito tempo na aprendizagem da própria ferramenta. O grande desafio na concepção e desenvolvimento da ferramenta aqui descrita foi encontrar uma solução intuitiva e de fácil utilização, direccionada para operários sem experiência em RV, para que, num curto espaço de tempo, utilizassem o sistema para treino das cablagens eléctricas onde iriam ser integrados. Desta forma, houve a necessidade de dedicar uma particular atenção às metáforas de interacção por forma a conseguir uma linguagem gestual aproximada da realidade. Este foi um dos principais problemas que se teve de enfrentar, conjuntamente com a complexidade da cablagem a simular.

2. DESCRIÇÃO DO PROBLEMA

O sistema de realidade virtual foi desenvolvido no âmbito de um projecto mobilizador do PEDIP II onde o parceiro industrial foi o cliente. O problema apresentado pela indústria de cablagens automóveis incidiu sobre o sistema de ensino e treino do processo produtivo vigente nas unidades fabris da mesma.

As razões que motivaram um estudo de efectividade de auxiliares de ensino e treino na linha de montagem fundamentam-se sobretudo nos seguintes aspectos:

- Elevado número de formandos/ano que passam pelo processo de ensino e treino nas diversas instalações fabris da indústria de cablagens automóveis.
- Custo elevado da criação de postos de trabalho apenas para tarefas de ensino e treino.
- As dificuldades sentidas pelos responsáveis da formação em transmitir aos formandos os conhecimentos necessários para a execução das tarefas que lhes são destinadas.
- Risco elevado ao nível do controlo de qualidade final de que são constituídas as tarefas efectuadas na linha de montagem. Estando estas constrangidas em termos temporais, são de extrema importância para a qualidade final do produto. Com efeito num muito curto período de tempo (cerca de quinze minutos) é feita a montagem completa de uma cablagem.
- A existência de fábricas em que da formação teórica se passa logo para a linha, implicando que o andamento global da linha seja atrasado visto que, tem de permitir que os novos trabalhadores se adaptem ao posto de trabalho.

Estes factores críticos conduziram ao estudo e desenvolvimento de um sistema de Realidade Virtual que permitisse o ensino e treino das sequências de montagem fora das linhas de produção, levando consequentemente a uma significativa redução de custos e a uma avaliação prévia das capacidades dos futuros operadores. O sistema é útil especialmente para unidades fabris sem centro de treino que normalmente usam as linhas de montagem para este efeito, com os prejuízos para a produção daí decorrentes.

Outra vantagem associada ao uso do ambiente de ensino e treino virtual foi permitir testar, fora da linha de montagem, determinadas tarefas consideradas críticas para a produção. Apesar de não ter sido possível, com base nos tempos do simulador, aferir tempos totais de produção, foi contudo exequível a produção de estimativas aproximadas. Deste modo, foi colocada ao dispor das unidades fabris uma ferramenta que lhes permitiu avaliar determinadas tarefas, através da elaboração de estatísticas que, para além de melhor identificarem os pontos críticos, ajudou aos responsáveis pela engenharia dos processos a planificarem novas estratégias de produção.

2.1 Objectivo

O ambiente de treino virtual teve como objectivo representar o mais fidedignamente possível um posto de trabalho para montagem de cablagens eléctricas. Por isso foi necessário contextualizar a disposição e funcionamento das linha de montagem , bem como as sequências de montagem em ambiente fabril.

2.1.1 Linhas de montagem

Na secção de montagem das cablagens, estão dispostas diversas linhas de montagem circulares, como exemplificado na fig. 1. Cada linha de montagem consiste numa estrutura metálica sobre a qual estão fixos conjuntos de painéis verticais, que circulam a uma velocidade pré-estabelecida, de modo a satisfazer a cadência de produção pretendida.

Em cada painel estão, por sua vez, dispostos suportes estrategicamente localizados sobre os quais vão sendo

colocados os componentes que constituem a cablagem. Ao longo de cada linha de montagem, estão colocados diversos operários em posições pré-definidas – um por painel – designados operadores de linha. Cada operador de linha é responsável por um conjunto de sub-tarefas, ou operações elementares, do processo de montagem. Entre estas referem-se a título de exemplo a colocação de cabos sobre o painel, cabos estes previamente preparados na secção de corte e cravação, a fixação de caixas e outros componentes, o encaixe de terminais nos conectores correspondentes, a colocação de tubagens e clips e o enfitamento de fios.

Fig. 1 Linha de montagem típica.

2.1.2 Sequências de montagem

Numa cadeia de produção, uma cablagem eléctrica é produzida através de várias sequências encadeadas, cada uma equivalendo a um conjunto de operações simples. Por exemplo: colocação de peças no painel, ligação de fios eléctricos (simples, duplos ou triplos) entre as várias peças do painel, realização de enfitamento de um conjunto de fios, colocação de epis, detecção e marcação de eventuais erros, etc.

Cada operador desempenha várias sequências por posto de trabalho e utiliza um estendal para o aprovisionamento dos objectos que precisa.

Numa hipotética reprodução virtual de todas as sequências e, por conseguinte, de todas as tarefas que os operadores realizam em ambiente fabril, surgiram alguns desafios de implementação, nomeadamente:

- simulação do enfitamento de fios;
- percepção de toque dos objectos virtuais;
- simulação do encaixe de um fio no alvéolo de uma peça (movimento e som associado);
- simulação de marcação de erros (através de "fitas autocolantes" próprias);
- validação da parte eléctrica da cablagem;
- representação do estendal.

2.2 Vantagens da Ferramenta

O uso do sistema de Realidade Virtual como uma ferramenta de ensino e treino na montagem de cablagens eléctricas apresentou as seguintes vantagens:

- A não existência de desperdício material;
- A possibilidade de repetição de sequências;
- A independência entre postos de trabalho, isto é, a realização de um posto não implica a realização prévia dos anteriores;
- Ensino passo-a-passo de todas as operações em cada sequência;
- A identificação automática de operações correctamente realizadas;
- A possibilidade de obtenção dos tempos realizados por cada passo e tempo global de execução de uma sequência.
- A necessidade de conhecimentos informáticos ser praticamente nula;
- Apoio ao formador na avaliação de competências e apetências.

Concluindo, o uso de um sistema deste género permite, de uma forma rápida e completa, a aprendizagem e treino da memorização de sequências de tarefas, a memorização espacial e gestual associada à resolução do posto de trabalho virtual.

3. O SISTEMA

3.1 Requisitos do Sistema de Realidade Virtual

Os principais factores seleccionados para o desenvolvimento de um Sistema e para a aquisição de material foram de ordem económica, tecnológica e humana. O Sistema de Realidade Virtual teria de ter um custo o mais adequado possível ao problema que pretendíamos resolver para poder ser utilizado numa sala de formação e facilmente ser replicado noutras salas de formação, ou seja, um custo que comporte a escalabilidade. Estando a tecnologia de Realidade Virtual num estado de evolução rápida e estabilização, ainda é raro encontrar sistemas completos de ensino e treino utilizando estas tecnologias devido às próprias limitações bem como pelos custos elevados que ela ainda implica. O terceiro factor relaciona-se com o nível de usabilidade por parte do utilizador final. Este é talvez o factor mais importante porque implica a aceitação do sistema por parte dos utilizadores que o vão usar, nomeadamente os formandos e formadores, cuja reacção à solução encontrada determina o seu sucesso. Sem uma boa aceitação da solução desenvolvida, e consequentemente da tecnologia adoptada, dificilmente faria sentido realizar esforços adicionais de aperfeiçoamento. A usabilidade é o factor mais difícil de controlar à priori, uma vez que só mediante a experiência de formação concreta com o grupo alvo é possível obter respostas e concluir orientações.

Estes factores influenciaram a forma como a visualização e interacção com o painel de montagem no Ambiente

Virtual foi feita. A escolha reverteu sobre o uso de capacetes de Realidade Virtual (HMD da expressão inglesa Head-Mounted Displays) em detrimento de uma projecção plana sobre um grande ecrã. Esta decisão teve em linha de conta as seguintes considerações:

- necessidade que o operário experimente um efeito de imersão completa no ambiente de treino Virtual, no menor período de tempo, por forma a se atingir rapidamente o máximo nível de concentração nas tarefas de treino em detrimento dos problemas de desajustamento provenientes de uma incompleta abstracção do mundo real;
- formando deve-se concentrar unicamente nas tarefas a serem desempenhadas e não na tecnologia;
- formando deve ter uma boa visibilidade quer sobre o painel quer sobre o estendal onde se encontram as peças e os cabos que lhe permitirão montar a cablagem;
- formando deve realizar, no ambiente virtual, o treino das várias acções recorrendo aos gestos e movimentos corporais de forma muito similar aos reais que irá efectuar na linha de montagem, sendo que estes gestos e movimentos também são parte integrante da aprendizagem;
- espaço necessário na solução usando HMD é bastante mais reduzido;

Neste contexto é ainda digno de registo que o sistema foi desenvolvido com uma única luva como interface de interacção, que só pode ser utilizado por destros ou ambidestros.

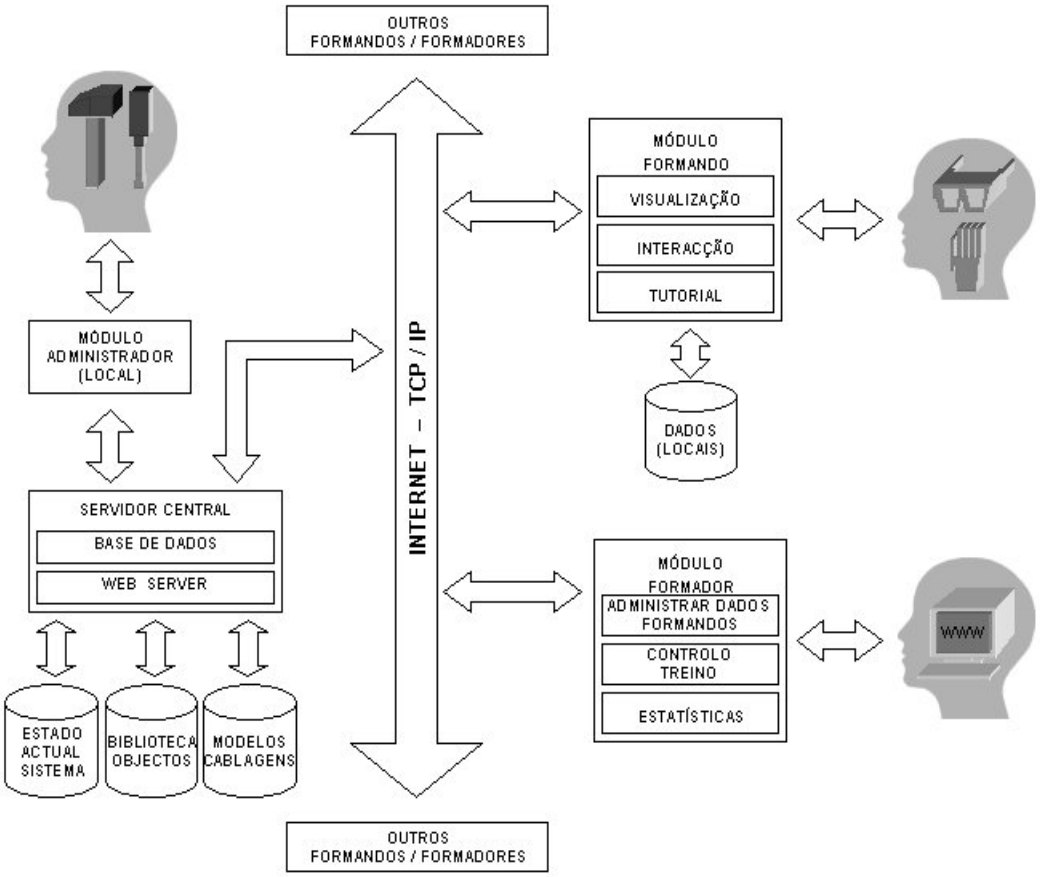


Fig. 2 Arquitectura

Por análise dos vários objectivos a atingir no processo de treino, ficou claro e aceite na generalidade, que o objectivo central é treinar os formandos na memorização da sequência correcta de montagem, os respectivos gestos e movimentos corporais, no reconhecimento de objectos (peças, contra-peças, etc.) assim como na habituação à nova configuração de montagem. O paradigma de interacção a adoptar tem assim forçosamente que facilitar esta memorização, já que a forma como a sequência correcta deve ser feita não é considerada fundamental, mas sim a sequência em si. Para o formando ter uma completa noção da forma como as operações deveriam ser efectuadas, seria de todo conveniente utilizar duas luvas. Devido a restrições orçamentais, a hipótese de utilizar duas luvas foi posta de parte.

No que diz respeito aos interfaces gráficos igual cuidado foi tido em relação aos factores humanos, quer por um lado, para os formadores disporem de uma ferramenta de fácil utilização que lhes permitisse elaborar e conduzir a sua formação, quer por outro lado, a interface do

formando que é constituída pelo AV e pela maneira com que o formando interage com esse AV.

3.2 Arquitectura

A especificação do projecto adopta uma arquitectura conceptual, cuja esquematização pode ser encontrada na fig. 2 usando a configuração de servidor-cliente. O servidor é o cérebro de todo o sistema, fornecendo os serviços requisitados pelos restantes módulos. Engloba a base de dados que contém os modelos dos objectos usados, a configuração das cablagens e ainda o estado actual do sistema para cada um dos treinos a decorrer. O servidor é o responsável por manter a consistência do sistema, acoplando ainda um servidor web (servidor de páginas na World Wide Web) responsável pela conexão via TCP/IP com os módulos formador e formando.

3.2.1 Módulos cliente

Existem três módulos cliente para os utilizadores tipo do sistema: administrador, formador e formando.

O **módulo administrador** é local relativamente ao servidor e disponibiliza as ferramentas necessárias para a concepção e configuração das cablagens a serem usadas nas acções de formação. O acesso a este módulo é restrito a quadros especializados da empresa na área de prototipagem e restruturação de cablagens.

O **módulo formador** fornece os meios para controlo do processo de treino. Permite a administração de dados de formandos, monitorização do treino em tempo real e obtenção de estatísticas. Uma grande vantagem neste módulo é a flexibilidade de acessibilidade e visualização de dados por parte dos formadores, tornada possível pela utilização de um browser web como interface com o utilizador.

O **módulo formando** é constituído por três componentes fundamentais: o motor gráfico, responsável pela geração e projecção das imagens; o controlo dos dispositivos de interacção; e o tutorial. Devido à quantidade de dados que o AV necessita, é mantida uma base de dados local que, no arranque do sistema, recebe os dados do servidor. Durante o processo de treino, os dados obtidos são introduzidos e actualizados na base de dados local que se encontra sincronizada com a base de dados central do servidor. Este módulo é activado remotamente pelo formador através da utilização de CGI's.

O sistema permite replicar os módulos formador e formando, oferecendo a possibilidade de se encontrarem no mesmo momento vários utilizadores em diversas acções de formação.

3.2.2 Componentes do sistema de realidade virtual

O sistema é composto por elementos que permitem a visualização, interacção e a gestão de todos os eventos que ocorrem durante a execução da aplicação.

Os componentes físicos do sistema são:

- Estação gráfica *SGI* (custo aproximado 60.000 €) ;
- Capacete de realidade virtual (Head-Mounted Display): *Kaiser Electroptics XL50* (custo aproximado 17.500 €), *Virtual Research V8* (custo aproximado 12.000 €);
- Dispositivo de interacção: *Virtual Technologies CyberGlove* (custo aproximado 13.000 €);
- Sistema de localização: *Ascension Tech Flock of Birds* (custo aproximado 7.000 €);

O software utilizado para gerar o interface do formando é uma plataforma para criação de sistemas de realidade virtual denominada *VirtualDesign2*. Para criar o interface com o formador recorreu-se a uma aplicação em ambiente web. São utilizadas para o efeito páginas dinâmicas em *PHP* suportadas por um servidor *Apache* e um servidor de base de dados *MySql*.

Fig. 3- Dispositivos de interacção

3.3 Modelos Geométricos e Configuração do Sistema

Um dos objectivos deste projecto foi reproduzir virtualmente, à escala real, o posto de trabalho de um operador para efeitos de treino e formação de operadores.

Um posto real é constituído por um painel onde o operador realiza uma determinada sequência de operações. Existe um estendal de aprovisionamento na retaguarda que contém um conjunto de fios, peças e outros objectos que serão colocados no painel.

O ambiente virtual do posto de trabalho consiste num ambiente imersivo que utiliza uma luva associada a um sistema de localização de posição para interacção com o AV. Este representa um posto de trabalho onde, mediante uma selecção prévia do formador, o formando é confrontado com as diferentes disposições de objectos que poderá encontrar no seu posto de trabalho real. Nessa situação deverá realizar as várias tarefas da sequência em questão com o auxílio dos dispositivos de interacção.

3.3.1 Modelação geométrica

No sistema fabril, o painel é um suporte físico concebido para a construção de uma dada cablagem eléctrica, onde se encontram acoplados objectos denominados contra-peças que irão servir de base para o encaixe da ou das peças correspondentes. Além destes, existem outros objectos acoplados ao painel, denominados forquetas, que servem de suporte aos fios condutores que ligam as várias peças da cablagem eléctrica.

A reprodução virtual do painel é feita respeitando as suas dimensões reais. As peças, contra-peças, forquetas e leds são modelados com detalhe à escala real, uma vez que um

dos requisitos da formação virtual é identificar correctamente o posto de trabalho.

A cablagem exemplo que está a ser utilizada contém cerca de 100 peças e 28 contra-peças. A maioria dos objectos apresentam uma geometria (irregular) construída a partir de polígonos quadrangulares que, após serem triangularizados, duplicam o seu número. Em média cada objecto é constituído por cerca de 850 triângulos, com excepção de alguns maiores que têm sensivelmente 4900 triângulos. O elevado número de polígonos gera um sistema potencialmente complexo que influencia o número de imagens geradas por segundo e exige maior utilização de recursos por parte do sistema. Para acelerar o processo de geração de imagens e obter *frame rates* entre 15 a 20 fps foi necessário proceder à diminuição do número de polígonos a visualizar.

Os testes conduzidos permitiram concluir que uma optimização da geometria é suficiente para obter a performance requerida. Foi possível observar que tanto a nível de interacção como de visualização foram obtidos bons resultados em termos de número de imagens geradas por segundo.

Os objectos que simulam os fios eléctricos não carecem de ser modelados uma vez que são gerados de forma dinâmica em tempo real aquando da sua colocação no painel.

Devido à constante alteração das linhas de montagem existentes e à introdução de novas linhas de montagem em empresas de construção de cablagens eléctricas, surgiu a necessidade de criar uma biblioteca capaz de armazenar as representações dos objectos utilizados. Como vantagem pode ser referida a possibilidade de reutilização de peças nos casos em que o novo painel contem peças idênticas, evitando o esforço de realizar uma nova série de modelos geométricos. Ao evitar este esforço, deixa de ser necessário contratar mão de obra especializada capaz de modelar os objectos e integrá-los na aplicação e investir em software específico, o que se traduz em redução de custos de manutenção da aplicação de tempo despendido.

3.3.2 Configuração do sistema

Em cada linha de montagem existe um determinado número de postos de trabalho cuja sequência de operações é diferente de posto para posto, embora sejam usadas peças em comum.

De forma a facilitar a definição destas sequências de operações, o módulo do administrador oferece funcionalidades de configuração do sistema, através do recurso a um editor de configurações. Desta forma permite-se a configuração por engenheiros da produção sem necessidade de possuírem conhecimentos informáticos aprofundados. Tarefas como a alteração da localização de uma peça no painel de montagem resumem-se a alterar as coordenadas numéricas da mesma.

4. DESENVOLVIMENTO DE INTERFACES

O desenvolvimento das interfaces do sistema de realidade virtual mereceu uma atenção redobrada. Para tal foram realizados estudos específicos sobre metáforas de interacção existentes para conceber o interface para o formando. Constatou-se que apesar dos diversos trabalhos de investigação realizados nesse domínio, os resultados ficam aquém das expectativas. Para além disso, foram levadas a cabo pesquisas sobre ambientes colaborativos distribuídos geograficamente para conceber o interface com o formador [Korves 1998].

Deste modo, o controlo da formação está dividido em dois tipos de interfaces dirigidos aos dois principais intervenientes. O interface para o formador é essencialmente bidimensional, constituído por janelas, um paradigma de interacção já bem conhecido. O interface para o formando é imersivo, menos conhecido pelo utilizador comum, podendo vir a ser a forma de interacção mais transparente e sobretudo mais intuitiva com um AV.

4.1 Interface com o Formador

Para dar resposta às necessidades identificadas, foi desenvolvida uma interface que permite ao formador realizar o seu treino com um determinado formando, mesmo que os dois se encontram em locais geograficamente distintos.

Deste modo, o interface com o formador serve para controlo do treino, actualização da base de dados com informação sobre os formandos, resultados e desempenho na realização dos treinos. Permite ainda pesquisas, estatísticas e análises sobre dados já recolhidos.

O interface com o formador é constituído por várias janelas que permitem inserir ou eliminar formandos da base de dados, fazer uma análise dos treinos já efectuados e compará-los a outros. É também possível obter estatísticas conjuntas de todos os formandos e chegar a conclusões sobre dificuldades em efectuar uma determinada operação.

O formador, sem sair do mesmo interface, pode controlar e monitorizar o treino virtual. O controlo contempla as seguintes operações: iniciar, interromper, terminar a simulação, especificar o exercício e o seu grau de dificuldade e permitir o uso do tutorial. A monitorização consiste na visualização dos tempos e da performance do formando numa determinada altura, bem como uma representação da disposição dos objectos no painel durante o treino. O formador terá ainda à sua disposição funções de ajuda ao formando, sob a forma de activação do tutorial.

Para alguém que nunca tenha experimentado um Sistema de Realidade Virtual (SRV), o tempo que um operador demora a concluir uma determinada tarefa não corresponde directamente ao tempo que demoraria numa situação real. Enquanto que um operador experiente, em ambiente fabril, demora por volta de 2 a 3 minutos aproximadamente a realizar uma tarefa, no AV ele

demora entre 5 a 10min, dependendo do tempo de habituação ao sistema.

No entanto, o tempo relativo entre as diferentes performances desempenhadas pelos diferentes operadores pode ser interessante como forma de apoio ao formador na avaliação de aptidões.

O interface com o formador permite controlar e receber os dados resultantes da formação. Este foi desenvolvido com tecnologias Web com o recurso a HTML, CGI´s e linguagens de script que permitem construir e aceder à base de dados, por forma a assegurar a portabilidade da interface. As principais vantagens que a plataforma para a interface apresenta são:

- formador só necessita, como software, de um browser HTML, que pode ser utilizado em qualquer sistema operativo;
- formador pode estar em qualquer lugar com acesso à Internet, constituindo assim uma forma de ensino e treino à distância;
- facilidade e flexibilidade na alteração do design do interface com o formador;
- a criação e manutenção do software de aceso à base de dados fica mais transparente, modular e extensível.

4.2 Interface com o Formando

O formando é imerso num Ambiente Virtual (AV) de treino que simula o posto de trabalho com realismo. Esta imersão é conseguida através de uma construção virtual de um painel e de todos os objectos perfeitamente reconhecíveis (peças, contra peças, bases, forquetas…). Para tal foi necessário utilizar uma estação gráfica suficientemente poderosa que assegurasse um número suficiente de imagens por segundo e um conjunto de metáforas de interacção intuitivas e representativas da realidade.

Um aspecto importante na concepção do interface com o formando foi o sistema de visualização. Factores como, a qualidade de imagem, a facilidade de deslocação do utilizador no AV, a imersão total do utilizador e a facilidade de interacção com a luva virtual foram decisivos na escolha do mesmo.

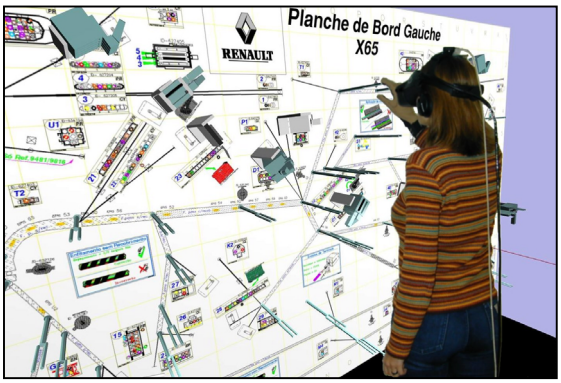


Fig. 4 Ambiente de treino virtual

A solução final recorreu ao uso de um capacete de realidade virtual (HMD). As vantagens associadas a este tipo de dispositivos de visualização são:

- imersão total no mundo virtual
- presença virtual
- visualização panorâmica
- mobilidade no espaço

A utilização da CyberGlove (luva de interacção) como dispositivo de interacção não revelou dificuldades de maior. A luva permitiu a manipulação de objectos e a comunicação com o SRV através de gestos.

4.3 Metáforas de Interacção

O estudo de metáforas utilizadas baseou-se na pesquisa das já existentes e devidamente documentadas em artigos da área [Poupyrev 1997]. Com base nos resultados obtidos da investigação foram criadas as metáforas para o sistema de realidade virtual. A preocupação central na concepção das mesmas foi encontrar a melhor forma de representar as acções reais no sistema de realidade virtual através de gestos simples e intuitivos, uma vez que o público alvo da ferramenta eram operadores sem experiência em informática.

Um factor importante na escolha das metáforas foi a restrição ao uso de uma mão (devido ao uso de uma só luva), por terem que se representar acções que no ambiente de trabalho real requerem o uso de ambas as mãos.

Dos vários desafios a representar, um dos mais interessantes foi a representação de fios. Devido às suas reduzidas dimensões, e de modo a permitir a sua manipulação, os fios foram representados simbolicamente por um objecto. Utilizou-se uma esfera para representar o fio, e verificou-se que os utilizadores conseguiam seleccionar o fio e manipulá-lo com relativa facilidade. Esta esfera representa o conceito de *novelo* de fio que o utilizador vai distribuindo pelo painel.

Fig. 5 - Aprovisionamento e colocação do fio

No entanto houve desafios/acções que não foram possíveis de representar devido ao actual estado de evolução das tecnologias, nomeadamente o cálculo do

posicionamento de movimentos circulares, não regulares e rápidos, como acontece no enfitamento de cablagens, ou a modelação de acções de montagem rigorosa ao nível de movimentos milimétricos.

5. TESTE E VALIDAÇÃO DO SISTEMA

Foi elaborado um questionário para avaliar o grau de imersão e de interactividade proporcionado pelo sistema de treino. As questões foram directamente colocadas aos utilizadores, e permitiram avaliar sobretudo as sensações subjectivas, isto é, as respostas descreveram como cada utilizador se sentiu e teve percepção dos dispositivos de Realidade Virtual e do próprio Ambiente Virtual.

O questionário teve como objectivo observar o conforto e a aceitação dos dispositivos, o grau de envolvimento e imersão no Ambiente Virtual, assim como a viabilidade de um Ambiente Virtual deste género como forma de treino e aprendizagem.

Paralelamente ao questionário, foram avaliados de uma forma mais objectiva, os resultados destas diferentes sessões de treino em termos de medição de tempos. O questionário foi constituído por três partes que foram respondidas em fases distintas: antes da sessão de treino, depois da sessão de treino, e um questionário especificamente para pessoas que têm ou já tiveram contacto com a linha de montagem de cablagens eléctricas. As três partes em que o questionário se dividiu são as seguintes:

- qual o estado do sujeito antes da experiência;
- qual o estado do sujeito após a experiência, como considerou ele o seu desempenho e qual o grau de imersão sentido;
- finalmente, algumas questões que seriam respondidas unicamente por indivíduos experientes nas linhas de montagem e que visava saber qual o nível de transferência de conhecimento que eles sentiram poder obter.

Os utilizadores foram inseridos num Ambiente Virtual sempre muito semelhante e com objectivos de desempenho também eles muito idênticos, de maneira a se poder utilizar os resultados do questionário com bastante confiança e se poderem tecer algumas conclusões sobre as sensações de um utilizador em ambientes deste género.

Foram também analisados os resultados provenientes directamente do sistema, divididos entre pessoas que já tinham tido formação real como grupo de controlo e pessoas que tiveram o primeiro contacto com o painel de montagem através do Sistema de Realidade Virtual. Assim, desta maneira foi possível ter dados concretos relativamente a transferência de conhecimento que o Sistema era capaz de promover.

O posto de trabalho da experiência de teste, foi modelado com base em postos reais actualmente existentes nas linhas de montagem.

5.1 Análise dos Resultados

A validação da aplicação de Realidade Virtual foi realizada em conjunto com os parceiros industriais. Eles disponibilizaram vários operadores para testarem a aplicação, que foram seleccionados segundo a maior ou menor experiência em linhas de montagens.

No âmbito da validação e teste foram registados diversos indicadores, desde a performance do utilizador, o tempo médio despendido a efectuar as operações, o feedback verbal de cada utilizador, ou até o tipo de gestos e movimentos realizados.

Associado a este registo empírico foi utilizado o questionário anteriormente referido, para registar numa perspectiva mais técnica o impacto que esta nova tecnologia teve sobre cada utilizador.

Os resultados obtidos foram analisados segundo três dimensões distintas para classificar o sistema sob um ponto de vista tecnológico, técnicas de interacção e o processo de ensino. Do tratamento efectuado ao questionário obtiveram-se as seguintes conclusões:

Imersão:

1. Bom nível de realismo da cena
2. Muito bom nível de imersão
3. Bom nível de ausência de náuseas e desorientação
4. Nível satisfatório no uso do capacete
5. O único factor de distracção foi o HMD, pois foi considerado incómodo devido ao seu peso. Normalmente, após um treino de 30 min, o utilizador apresentava algum cansaço, apontando o HMD como o responsável desse desconforto.

Os resultados obtidos podem ser considerados bastante positivos, uma vez que os objectivos que se pretendiam alcançar, nomeadamente a imersão completa e o realismo do mundo virtual em que eles foram inseridos, foram atingidos e a aceitabilidade da tecnologia por parte do utilizadores foi muito satisfatória.

No entanto alguns utilizadores sentiram algum mal estar no final da experiência. Este dado, apoiado por estudos na área, [Pierce 1997] indica que os indivíduos já com pré-disposição para enjoar em situações normais tais como viagens de carro, avião e principalmente de barco, podem sentir algum mal estar perfeitamente normal, mas não impeditivo no uso destas tecnologias.

Interacção:

1. A liberdade de controlo da interacção foi satisfatória
2. Muito Bom nível de envolvimento
3. Bom Grau de satisfação no uso da luva
4. Nível satisfatório na noção de sobreposição entre a mão real e virtual .

O uso da luva como dispositivo de interacção não introduziu obstrução na forma de interagir com o sistema uma vez que os utilizadores a encararam como a forma

mais natural de interacção com o painel na resolução dos postos de trabalho virtuais.

Treino:

1. No geral, foi verificado um bom desempenho na realização das tarefas
2. Identificação muito fácil das peças
3. Identificação muito fácil do painel
4. Apenas metade dos inquiridos sentiram mais facilidade no Ambiente Virtual do que no real
5. Nível de imersão na linha de montagem foi considerado satisfatório.

Como seria de esperar, o ambiente virtual não substitui totalmente a realidade mas consegue-se atingir níveis de abstracção tais que permitem a concentração exclusiva no exercício.

Análise global:

- A sensação de imersão sentida foi quase completa;
- Houve facilidade de identificação do painel e das peças virtuais que compõem o painel;
- O nível de desempenho sentido foi bastante satisfatório;
- A manipulação dos objectos foi sentida como muito realística;
- Bom grau de envolvimento uma vez que os utilizadores se preocuparam unicamente com a resolução dos exercícios sem terem que pensar como efectuar a manipulação;
- Correspondência da mão virtual com a mão real aceitável embora não haja uma sobreposição completa. Os utilizadores conseguiram controlar com facilidade a mão virtual;
- Os gestos de interacção foram classificados pelos utilizadores como fáceis de memorizar mesmo não sendo uma representação fiel dos gestos que eles realizam no posto de trabalho real;
- A luva virtual como dispositivo de interacção foi bem aceite pelos utilizadores do sistema e considerada confortável e de fácil utilização;

No final das sessões de treino com o sistema de Realidade Virtual estimou-se que cada utilizador precisou entre:

- 5 min a 15 min para a habituação inicial aos dispositivos do sistema (luva e capacete) ;
- 1 a 8 horas para memorização do posto de trabalho, dependente da complexidade do posto e da capacidade de aprendizagem do operador;
- 15 a 25 minutos para executar um posto de trabalho completo, havendo uma redução gradual de tempos que estabilizavam após haver uma memorização da sequência de operações.

A comunicação constante entre os programadores e os vários utilizadores do sistema trouxe melhoramentos à aplicação, quer na detecção de erros subjacentes ao sistema, quer na correcção da disposição dos objectos no painel.

5.2 Formação com treino em AV vs. formação convencional

A formação com treino em AV por si só não é suficiente para suprimir o arranque lento de produção de cablagens, cada vez que se inicia uma nova linha de montagem. Dependendo do grau de complexidade da cablagem a produzir, por vezes a produção nos primeiros dias é quase nula. No entanto, usando o AV de treino como um complemento na formação teórica, pode em certa medida, diminuir o referido atraso na produção.

Assumindo que cada operário tem 15 dias de formação por ano (valor médio), um ganho de 20%, obtido pelo uso do sistema de treino virtual, resultaria em 3 dias ganhos. Num cenário de 900 operários em formação por ano e um ganho de 25 € por dia, haveria um ganho de 25 € x 3 dias x 900 operários = 67.500 € por ano. Os dados anteriores foram fornecidos pelo parceiro industrial.

6. CONCLUSÕES

Apesar da juventude das tecnologias de Realidade Virtual com aplicações industriais, pode-se concluir que os resultados obtidos excederam as expectativas quer em termos sociais (aceitação, conforto e motivação) quer em termos industriais (a aprendizagem efectuada e redução de custos). Foi possível observar na prática que grande parte dos objectivos propostos no início do projecto foram implementados e testados com sucesso.

Com este projecto é possível ensinar de uma forma exacta sequências de montagem de cablagens eléctricas, através de interfaces para os formandos e para os formadores, fáceis de utilizar mesmo sem qualquer conhecimento prévio de informática.

O formador tem acesso a uma ferramenta que além de ser um suporte de ensino e treino, é também uma ferramenta de apoio na avaliação das competências e apetências dos operadores da linha de montagem.

O projecto permitiu ainda, não ultrapassando os custos estabelecidos em termos de equipamento e utilizando hardware de Realidade Virtual que já começa a ser standard, conceber uma aplicação que tem sérias aplicações reais realizando assim algumas promessas que esta tecnologia tem tido como expectativas.

Como nem sempre tudo só tem aspectos positivos, houve pontos que por motivos simplesmente tecnológicos actuais ainda não teve resultados tão satisfatórios. As pessoas que efectuaram o treino tiveram algum desconforto na utilização dos capacetes de Realidade Virtual e a falta de sensação táctil e de force-feedback também não permitiu uma imersão e envolvimento mais completo. A existência de uma segunda luva permitiria ter uma interacção mais natural. Ainda a nível do hardware utilizado, as luvas requerem um cuidado particular e devido à sua sensibilidade requerem uma

calibração constante e diferente para cada utilizador [Pierce1997] [Fuhrmann. 2000].

A experiência adquirida ao logo deste trabalho permite concluir que a utilização das tecnologias de realidade virtual como suporte ao ensino e treino constitui uma área pouco explorada pela indústria, mas com muito potencial para optimizar os processos de produção das mesma.

Dentro desta área de conhecimento de ensino e treino, diferentes trabalhos de investigação podem ser realizados, nomeadamente em sub-áreas mais específicas como sejam a montagem e desmontagem de produtos, ergonomia, e prototipagem virtual.

7. AGRADECIMENTOS

Os autores gostariam de agradecer ao PEDIP II pela possibilidade financeira em apoiar projectos nesta área emergente e ao Centro de Computação Gráfica que através de Bolsas de Investigação e Desenvolvimento permitiu, em parte, a realização deste projecto.

8. REFERÊNCIAS

[Cruz-Neira 1993] Cruz-Neira, C., Sandin, D. J., DeFanti, T. A.; *Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE*; Proceedings of ACM SIGGRAPH'93; USA, 1993.

[Foley. 1997] James Foley, Andries van Dam, Steven Feiner and John Hughes, *Computer Graphics – Principles and Practice*, 2nd edition, Addison Wesley, 1997

[Fuhrmann. 2000] Fuhrmann, A., Schmalstieg, D., Purgathofer, W.; *Pratical Calibration Procedures for Augmented Reality*; Eurographics Virtual Environments 2000; Amsterdam; June 2000.

[Grave 2000] Luís Grave, António F. Silva, Cristina Escaleira, Cistina Silva e Adérito Marcos, *Trabalho Cooperativo em Ambientes Virtuais de Treino de Montagem de Cablagens Eléctricas*, COOPMedia'2000 - 1º Workshop de Sistemas Multimédia Cooperativos e Distribuídos, Auditório da Comissão de Coordenação da Região Centro, Coimbra, 2000

[Hoxie 1998] Hoxie, S., Irizarry, G., Lubetsky, B., Wetzel, D; *Developments in Standards for Networked Virtual Reality*; IEEE Computer Graphics and Applications, March/April 1998.

[Kalawsky 1993] Roy S. Kalawsky, *The Science of Virtual Reality and Virtual Environments*, Addison-Wesley, 1993

[Kenny 1994] Patrick J. Kenny and Tim Saito, *Results of a Survey on the Use of Virtual Environment Technology in Training NASA Flight Controllers for the Hubble Space Telescope Servicing Mission*, NASA, EUA, 1994

[Korves 1998] Korves, B., Loftus; *The Application of Immersive Virtual Reality for Layout Planing of Manufacturing Cells*; Advances in Manufacturing Technology XII, Volume 12; RW Baines, Ed. UK, Professional Engineering Pub. Ltd., 1998; 303-308.

[Mine 1997] Mine, M. R., Brooks Jr., F. P., Sequin, C. H.; *Moving Objects in Space: Exploiting Proprioception in Virtual-Environment Interaction*; Proceedings of SIGGRAPH'97; Los Angeles; 1997.

[Pierce1997] Jeffrey S. Pierce, Randy Pausch, Christopher B. Sturgill and Kevin D. Chirstiansen, *Designing a Successful HMD-Based Experience*, Presence Vol. 8 Nº4, MIT Press, EUA, 1997

[Poupyrev 1997] Poupyrev, I., Weghorst, S., Billinghurst, M., Ichikawa, T. *A Framework and Testbed for Studying Manipulation Techniques for Immersive VR*; Proceedings of ACM VRST'97; Lausanne; 1997.

[Psotka 1995] Psotka J.; *Immersive Tutoring Systems: Virtual Reality and Education and Training*, U.S. Army Research Institute technical report, Instructional Science; 1995.

[Sá 1999] Sá, G. A., Zachmann, G.; *Virtual Reality as a Tool for Verification of Assembly and Maintenance Processes*; Computer & Graphics, Volume 23, N.º 3; (December 1999), 389-403.

Jogos de Guerra e Paz

Helder Batista
IST/INESC
Lisboa
hjpb@rnl.ist.utl.pt

Vasco Costa
IST/INESC
Lisboa
vasc@rnl.ist.utl.pt

João Pereira
IST/INESC
Lisboa
jap@inesc.pt

Sumário
A simulação de ambientes virtuais realistas de grande complexidade e interactividade é um problema complexo a vários níveis. Na parte de visualização as dificuldades prendem-se com as limitações do hardware para apresentar uma cena complexa a ritmos interactivos. Na parte das comunicações o desafio coloca-se na gestão de um número muito elevado de participantes mantendo a coerência do ambiente sem sacrificar o desempenho da simulação. Propomos de seguida soluções que procuram tornar possível a realização de simulações de elevado realismo em equipamentos comuns, de baixo custo, disponíveis actualmente no mercado.

Palavras-chave
Simulação, Visualização, HLA, Ambientes Virtuais, Distribuição, Nível de Detalhe Contínuo.

1. INTRODUÇÃO

O desenvolvimento de simulações em larga escala iniciou-se nos meios militares para facilitar o treino e simulação de situações de combate. Exemplo disto são esforços pioneiros como a rede SIMNET [Pope89] e o projecto NPSNET [Macedonia95a, Macedonia95b].

Para a sua realização eram, e ainda são hoje em dia, usados equipamentos de topo de gama em termos de capacidade gráfica e redes dedicadas - que obviamente estão ao alcance apenas de instituições com elevados recursos financeiros.

Vamos usar uma aplicação do tipo simulador de combate aéreo para explorar as soluções disponíveis para resolver os problemas específicos relacionados com este tipo de aplicações. As principais dificuldades a resolver são:

- A visualização de um terreno de grandes dimensões em tempo real.
- A visualização de um grande número de objectos presentes na cena.
- A gestão da participação de um grande número de utilizadores na simulação.

Recentemente foi estabelecido pelo Departamento de Defesa dos E.U.A. (DoD) um novo padrão para a arquitectura em rede de simulações. A *High Level Architecture* (*HLA*) [DMSO98]. Neste momento todas as simulações do DoD são realizadas sobre essa arquitectura e todas as aplicações antigas estão a ser convertidas para o novo sistema.

Neste artigo primeiro descrevemos muito sucintamente a arquitectura da aplicação. Em seguida mostramos em mais detalhe como é efectuada a visualização do mundo. Depois descrevemos em pormenor como foi feita a distribuição dos dados entre os vários utilizadores. Por fim apresentamos as nossas conclusões e ideias para trabalho futuro.

2. ARQUITECTURA DA APLICAÇÃO

A aplicação é composta por vários módulos: (**Figura 1**)

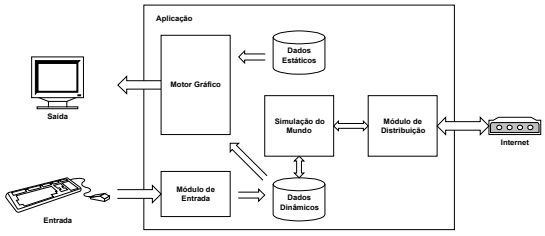


Figura 1 – Arquitectura da aplicação.

Motor Gráfico	responsável pela visualização do mundo e pela detecção de colisões entre os vários objectos.
Módulo de Entrada	interpreta os comandos do utilizador e executa as alterações necessárias no mundo.
Simulador do Mundo	responsável pela simulação de certos fenómenos do mundo como a física newtoniana dos objectos, meteorologia, etc.

Módulo de Distribuição responsável pela distribuição dos dados relevantes para manter a coerência do mundo visto pelos participantes na simulação.

Escolhemos separar os dados estáticos dos dinâmicos já que os dados estáticos são tipicamente mais volumosos. Assim a aplicação é instalada na máquina junto com os dados estáticos e estes não têm de ser transmitidos pela rede. Exemplos de dados estáticos incluem: as malhas de triângulos dos caças e do terreno, texturas, etc. Os dados dinâmicos incluem a posição dos caças, mísseis, etc.

3. VISUALIZAÇÃO

Para a visualização interactiva de cenas complexas o ideal é poder adaptar o nível de detalhe da cena de acordo com alguns parâmetros, como por exemplo, a distância de um objecto ao observador e o poder computacional da máquina.

Existem alguns métodos, designados por de nível de detalhe (*level of detail* ou LOD), que consistem em armazenar várias representações de um mesmo objecto com níveis de complexidade diferentes, usando o nível mais apropriado consoante desejamos mais ou menos detalhe.

Esta aproximação, embora de implementação simples, tem algumas desvantagens, pois conduz ao armazenamento de várias cópias do mesmo objecto com o consequente desperdício de memória.

A transição entre níveis de detalhe é muito brusca e portanto desagradável para o observador (um efeito vulgarmente designado na literatura por *popping*). A sua aplicação a terrenos não é viável devido às dimensões dos dados envolvidos.

Existem algoritmos específicos para a visualização de terrenos de grandes dimensões, tais como algoritmos baseados em árvores quaternárias [Lindstrom96], o ROAM [Duchaineau97] e outros [Garland97], mas não resolvem o problema para os modelos tridimensionais em geral. Um bom resumo do trabalho na área da simplificação de objectos é [Heckbert97].

O algoritmo desenvolvido por Hugues Hoppe, por nós designado como de malhas progressivas (*progressive meshes*) [Hoppe96], foi utilizado no projecto. Este algoritmo tem várias características que o tornam interessante para a realização deste tipo de aplicações:

- Nível de detalhe contínuo (CLOD), ou seja, os modelos podem ser alterados em apenas um vértice de cada vez, dessa forma tornando as mudanças no detalhe mais suaves.
- Pode ser aplicado de forma eficiente tanto a modelos tridimensionais comuns como ao caso específico dos terrenos que, neste tipo de simulação, é uma parte importante quer do ponto de vista do realismo quer do desempenho.
- Suporta refinamento selectivo, ou seja, o aumento ou diminuição do detalhe de zonas particulares do modelo - essencial no caso dos terrenos em que geralmente a área visível é muito inferior às dimensões totais do modelo.
- Subdivisão irregular da malha de polígonos - o que resulta em malhas com menor número de polígonos e com menor erro de aproximação.
- As fases computacionalmente complexas do algoritmo são efectuadas numa fase de pré-processamento o que permite um alto desempenho em tempo-real.
- Suporta a criação de *geomorphs*, ou seja, a interpolação das transformações efectuadas no modelo de forma a suavizar ainda mais os efeitos de *popping*.
- Suporta a preservação de características visuais importantes, como as fronteiras entre materiais.

3.1 Malhas Progressivas

Os detalhes deste algoritmo podem ser consultados em [Hoppe96, Hoppe97 e Hoppe98]. No entanto dado que as variações na implementação podem ter grande influência na qualidade da malha simplificada vamos abordar sucintamente os aspectos mais importantes da implementação.

O algoritmo é aplicado de igual forma tanto ao modelo do terreno como aos restantes modelos na simulação.

Neste algoritmo, um modelo tridimensional é transformado numa árvore binária em que cada nó contém um vértice da malha de polígonos.

No entanto o terreno dispensa o armazenamento de certos atributos como materiais, as normais e as respectivas considerações para a medição do erro das transformações. A árvore do terreno é também muito mais sensível em relação ao uso de memória já que tipicamente o modelo do terreno tem números muito elevados de polígonos (tipicamente mais de 1 milhão de polígonos).

A construção da árvore é efectuada durante a fase de pré-processamento, pelo que o tempo de construção não é crítico, logo é preferível usar uma medida de erro que permita obter malhas de boa qualidade.

Pode facilmente constatar-se que tipicamente não é necessário armazenar todos os vértices originais de um modelo para obter o mesmo nível de detalhe. Esse facto é explorado, sendo possível indicar um nível de erro que é desprezável, dessa forma eliminando vértices e ficando com um modelo que ocupa muito menos espaço que a malha completa sem um sacrifício perceptível da qualidade.

O factor crítico, para a qualidade das malhas de polígonos simplificadas, é a correcta medição da distorção visual introduzida pela eliminação de um vértice. No entanto, para a medição do erro introduzido tem de haver um compromisso entre tempo de execução e complexidade de implementação e fidelidade da medida. Na prática é mais correcto designar esta medição por heu-

rística já que é bastante complexo implementar uma medida que reflicta fielmente a distorção visual introduzida por uma transformação na malha.

A função por nós construída tem a seguinte forma:

$$erro = \sum_i \left[\left(\left\| N(g_i) - N(f_i) \right\| + \frac{\sum_j A(g_j)}{\sum_k A(f_k)} \right) \times A(g_i) \right]$$

$$\forall i : f_i \in F, g_i \in G$$

Onde:

F	é o conjunto das faces originais.
G	é o conjunto das faces modificadas pela transformação.
A(x)	é a área da face *x*.
N(x)	é a normal da face *x*.

Pode dividir-se a fórmula em três partes:

- $\left\| N(g_i) - N(f_i) \right\|$ - Medida da variação da orientação das faces afectadas: contribui para a preservação da aparência da superfície.
- $\frac{\sum_j A(g_j)}{\sum_k A(f_k)}$ - Medida da variação da área: contribui para a preservação do volume do modelo.
- $A(g_i)$ - Peso de cada face: determina que alterações em faces maiores introduzem mais erro

Para os modelos com vários materiais, a fórmula é idêntica, mas os conjuntos de faces são separados por material de forma a preservar as fronteiras de descontinuidade entre faces de materiais diferentes.

Pode ver-se que os resultados desta heurística são aceitáveis, obtendo-se representações com boa qualidade mesmo com um número de polígonos muito inferior ao original, tanto para o terreno como para modelos genéricos (ver **Figura A** em apêndice).

Dado que numa situação normal apenas está visível uma parte do terreno, é natural que apenas se introduza detalhe nessa zona, para isso é necessário usar técnicas de *view frustum culling*, ou seja, testar os vértices do terreno para determinar se estão no volume visível, e ajustar o detalhe da malha de acordo com esse resultado e proporcionalmente à distância do observador (os objectos mais distantes devem ter menos detalhe).

Neste momento não foram efectuadas optimizações neste teste que é uma parte importante para o desempenho. Poderia por exemplo, ser efectuada uma divisão do terreno em quadrículas, determinando-se rapidamente se um conjunto grande de vértices está dentro do volume de visualização, reduzindo consideravelmente o tempo gasto nestes testes [Assarsson99], [Assarsson00].

Para aumentar a performance, repartimos este trabalho por vários quadros (*frames*), reduzindo a carga, especialmente se usarmos um número de polígonos muito elevado. Esta técnica no entanto introduz um certo atraso no refinamento adequado da malha o que pode aumentar um pouco o efeito de *popping,* no entanto, ao permitir usar um número mais elevado de polígonos com mais *performance*, esse efeito é minimizado.

Para obter um ritmo constante, é usado um mecanismo de ajuste dinâmico do erro permitido, de forma a manter um número constante de polígonos na cena.

Em tempo real a lista de vértices e faces activas é a única estrutura actualizada e é muito menor que o número total de vértices do modelo. Desta forma uma das vantagens deste algoritmo é o seu desempenho ser independente do tamanho total do modelo.

A estimativa do valor do erro aceitável é determinada pelo teste que indica se o vértice está visível ou não e pela distância ao observador.

Mesmo com o *hardware* de hoje em dia, o suporte de terrenos exremamente grandes é problemático, pois não é possível mantê-lo completamente em memória.

Para o suporte de terrenos extremamente grandes a estrutura da malha progressiva está desenhada de forma que torna possível a divisão do terreno em zonas, que podem ser construídas separadamente, acelerando o processo de pré-processamento e que podem ser cobertas com mosaicos da textura maior (*texture tiles*), desta forma, ultrapassando as limitações das dimensões máximas da textura da placa gráfica, isto porque as zonas individuais do terreno mantêm a forma quadrada ou rectangular. Em tempo real, estas zonas podem ser carregadas ou retiradas de memória conforme o necessário.

Outra das optimizações efectuadas foi a implementação de *geomorphs*. Este mecanismo consiste na interpolação ao longo do tempo das alterações na malha, desta forma tornando o efeito de *popping* ainda mais difícil de detectar.

Como foi dito anteriormente, o objectivo é a manutenção de um ritmo elevado mesmo quando há muitos objectos visíveis na cena. Portanto o número de triângulos não pode aumentar linearmente com o número de objectos. Também não é conveniente mantê-lo constante pois iria reduzir o desempenho desnecessariamente quando há poucos objectos em cena e ter pouca qualidade com muitos objectos visíveis. Portanto usamos uma heurística para aumentar o número de triângulos de forma não linear. Por cada duplicação do número de objectos visíveis (neste caso os objectos são os caças) aumentamos o número de polígonos em 1/4. Esta aproximação é suficiente para manter uma qualidade gráfica aceitável, já que o detalhe dos caças é também ajustado conforme a sua distância ao observador. A qualidade gráfica global pode ser ajustada mudando o número de triângulos usa-

dos para desenhar um único caça (os triângulos *base*). Esta fórmula pode ser aproximada por:

$$\sqrt[3]{n} \times base; n = \text{número de objectos}$$

3.2 Resultados

Os testes foram efectuados num AMD Duron 800 MHz com uma placa gráfica NVIDIA GeForce 2 MX 32 MB que é neste momento um sistema de baixo custo. O protótipo corre sobre o sistema operativo Windows 2000 utilizando a biblioteca 3D OpenGL [ARB99]. A cena foi visualizada numa janela 800x400.

Os resultados de desempenho apresentados na **Figura 2** são para a navegação num terreno gerado a partir de um mapa de elevação de 2045x2045 pontos representando uma área de 120x120 km^2, que foi dividido em 16 zonas de 512x512 pontos (ver **Figura B** em apêndice).

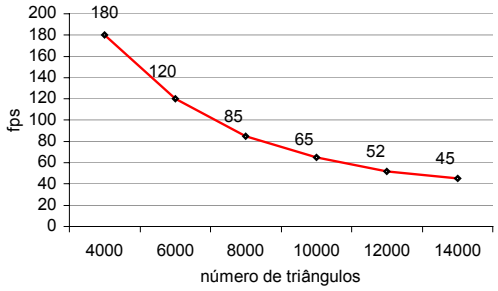


Figura 2 – Variação do desempenho consoante o número de triângulos no terreno.

Na **Figura 3** está um exemplo da progressão do número de polígonos com o aumento do número de caças, usando diferentes número de triângulos *base* (repare-se que a escala do número de caças é logarítmica).

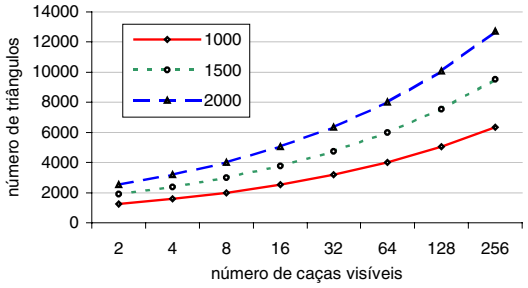


Figura 3 – Variação do número de triângulos consoante o número de caças visíveis.

Na **Figura 4** mostra-se o desempenho numa cena com um terreno com 6000 triângulos e vários números de caças visíveis, com 1500 triângulos *base* (ver **Figura C** em apêndice).

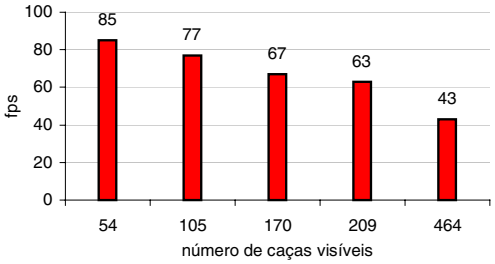


Figura 4 – Variação do desempenho consoante o número de caças visíveis.

4. DISTRIBUIÇÃO

Os problemas de comunicação podem ser resumidos a apenas dois: largura de banda e latência.

Nós desejamos suportar comunicações de longa distância em equipamento de consumo. Temos de suportar os modems de 56Kbps já que são o principal meio de ligação à Internet hoje em dia. As tecnologias de banda larga tais como o ADSL e os modems por cabo ainda não estão suficientemente divulgadas.

Estudos feitos em aplicações multimédia indicam que os seres humanos ficam desconfortáveis com inconsistências na ordem dos 100 ms. Estas inconsistências tornam-se intoleráveis quando na ordem dos 200 ms [Singhal97]. Em ligações transatlânticas sobre a Internet podemos esperar latências entre 200 e 400 ms.

Mesmo que a latência devido ao software e hardware possa ser reduzida estamos limitados pela velocidade da luz quando comunicamos sobre vastas distâncias.

De modo a que os seres humanos não se sintam perturbados é necessário esconder de algum modo a latência induzida pela rede. Também temos de conseguir suportar modems de 56Kbps ao simular centenas de objectos.

Jogos de computador tradicionais como o *Dogfight* e o *DOOM* utilizaram a abordagem ingénua de enviar a posição de todos os objectos simulados uma vez por quadro [Zyda99]. Apesar de esta abordagem funcionar sobre redes locais, devida à elevada largura de banda e baixa latência destas, sobre a Internet apenas a elevada latência é suficiente para arruinar a experiência do utilizador.

Os esforços pioneiros SIMNET e NPSNET aliviaram estes problemas com a introdução de técnicas como a estimativa (*dead-reckoning*) [Aronson97, Zyda99] e a difusão em grupo (*multicasting*) [Deering89].

4.1 Difusão em Grupo

A difusão em grupo (*multicasting*) é utilizada em comunicações entre grupos de máquinas. Cada máquina encontra-se registada num grupo de difusão que possui um dado endereço IP. Os encaminhadores (*routers*) da rede encarregam-se de entregar aos vários membros os pacotes endereçados ao grupo. Utilizando a difusão em grupo podemos reduzir o tráfego necessário para comunicar um evento a várias máquinas.

4.2 Estimativa

As técnicas de estimativa (*dead-reckoning*) trocam alguma consistência na simulação para conseguir fingir uma latência mais baixa e reduzir os requisitos de largura de banda. Elas conseguem atingir estes resultados utilizando predição. Os antigos marinheiros conseguiam empiricamente localizar com algum grau de precisão a sua posição num mapa dado o ponto de partida, a orientação do navio (utilizando uma bússola), a velocidade (medida em nós) e o tempo passado desde que saíram do porto. Utilizando técnicas semelhantes podemos prever a localização de outros objectos. No nosso caso dos caças e dos mísseis. Para isso basta lembrarmo-nos da física elementar:

$$\vec{x} = \vec{x_0} + \vec{v_0}t + \frac{1}{2}\vec{a}t^2$$

Esta equação como é sabido permite descrever o movimento de um corpo com aceleração constante. Sabendo os dados da posição inicial, a velocidade inicial, a aceleração e o tempo decorrido podemos saber onde nos encontramos num dado momento.

A utilização de um protocolo de estimativa permite maiores ganhos em objectos e eventos mais previsíveis. Quando um objecto é completamente controlado por um humano, o seu comportamento é eventualmente imprevisível. No entanto pode usar-se vária informação para prever mais fielmente o comportamento futuro de um objecto. Por exemplo, num caça há limites dinâmicos que têm de ser respeitados, por isso pode ser relativamente fácil prever a sua posição mesmo quando controlado por um ser humano.

4.3 HLA

A HLA [DMSO98] define o modo como deve ser realizada uma simulação desde a documentação dos dados até à implementação e forma de comunicação entre os utilizadores. A HLA define também uma biblioteca (libRTI) [DMSO00] que suporta a construção e operação de simulações distribuídas. Esta biblioteca está num nível de abstracção acima dos protocolos de comunicação usados e está disponível para várias linguagens de programação.

É definido um modelo de objectos e um modelo de interacção comum a todos os participantes (Federados) numa simulação (Federação), o que possibilita a troca de informação de forma uniforme.

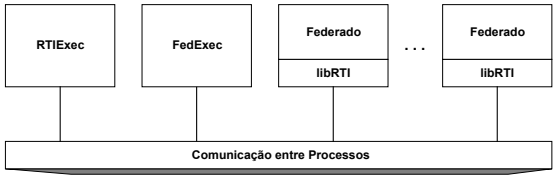


Figura 5 – Componentes da RTI.

A interface de comunicação do HLA (*Run-Time Interface* ou RTI) (ver **Figura 5**) faz com que uma simulação se apresente a um participante como uma entidade única, embora se encontre distribuída pelos vários participantes, não sendo necessária nenhuma componente centralizada. Aliás a centralização é desencorajada.

Para efectuar com eficiência a comunicação de mensagens entre vários participantes a RTI pode usar difusão em grupo.

4.4 Implementação

Decidimos aplicar carimbos temporais às actualizações da posição tal como no protocolo PHBDR [Singhal95, Singhal97]. Isto reduz o erro entre a posição real e a posição mostrada remotamente mas requer que os computadores tenham os relógios sincronizados. Felizmente é fácil de cumprir este requisito utilizando o *Network Time Protocol* (NTP) [Mills92]. O sistema operativo Windows 2000 possui de origem uma implementação do *Simple Network Time Protocol* (SNTP) [Mills96] que é adequada às nossas necessidades.

Para reduzir o erro é enviado um pacote sempre que a distância entre a posição real e a posição prevista exceda um determinado valor. Quanto mais elevado for o valor escolhido, menos pacotes serão enviados pela rede, mas valores elevados causam movimentos de correcção da trajectória mais bruscos e portanto mais desagradáveis.

Para suavizar este movimento movemos o objecto suavemente da posição anterior para a nova posição utilizando um algoritmo de convergência. Foi implementada uma forma de convergência linear para suavizar o movimento.

A informação da orientação do objecto é enviada junto com a informação da posição.

4.5 Resultados

Testamos o protocolo de estimativa por nós desenvolvido na trajectória de um caça representada na **Figura 6**.

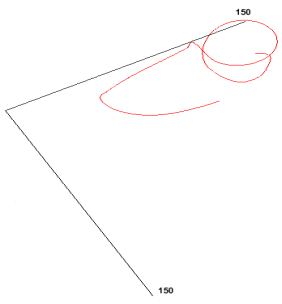


Figura 6 – Trajectória de um caça.

O caça esteve parado aproximadamente durante 1 segundo antes de iniciar o movimento.

O número de pacotes enviados utilizando a estimativa é uma ordem de magnitude inferior ao número de pacotes enviados com a abordagem ingénua como podemos ver na **Figura 7**. Todavia o erro na posição do caça vista remotamente é ligeiramente superior visto que apenas enviamos um pacote quando a posição actual e a posição prevista diferem mais que um certo valor que não é nulo. Este problema pode ser observado na **Figura 8.1**.

A convergência linear também aumenta o erro visto que o caça demora mais tempo a voltar à trajectória real como pode ser visto na **Figura 8.2**.

A introdução de latência como esperado também aumenta o erro. Todavia uma vez que o pacote com a actualização seja recebido o caça volta à trajectória real como pode ser visto na **Figura 8.3**. O erro máximo nesta trajectória é ligeiramente inferior a 7 metros.

Quando a latência é elevada o erro observado é proporcional à velocidade do caça e à latência da rede.

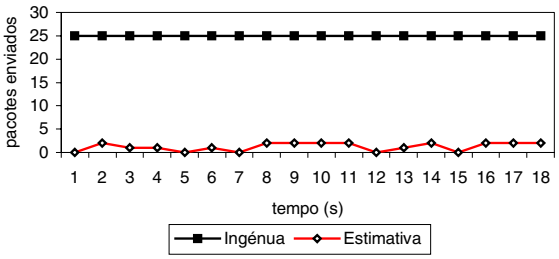


Figura 7 - Número de pacotes enviados utilizando a abordagem da estimativa vs. a abordagem ingénua.

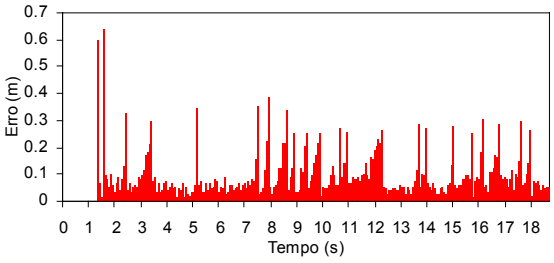


Figura 8.1 - Erro com latência de 0 ms e sem convergência.

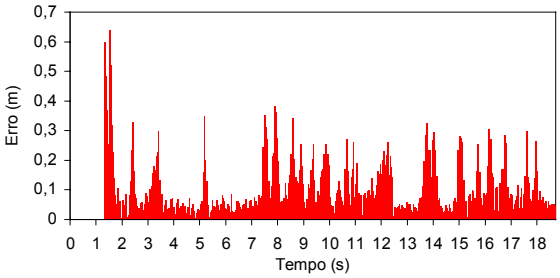


Figura 8.2 - Erro com latência de 0 ms e convergência linear.

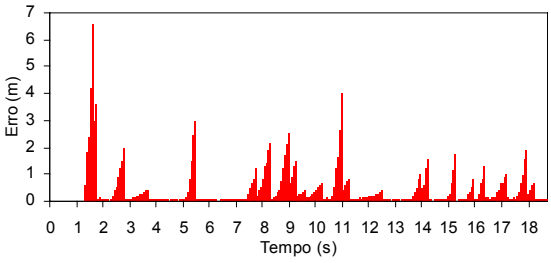


Figura 8.3 - Erro com latência aleatória de 200 a 400 ms e sem convergência.

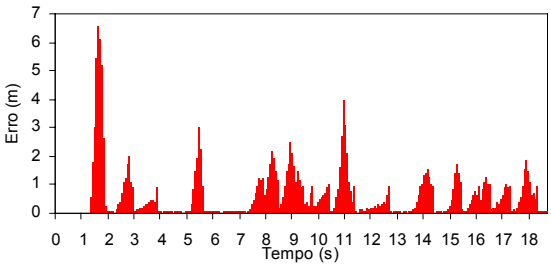


Figura 8.4 - Erro com latência aleatória de 200 a 400 ms e convergência linear.

5. CONCLUSÕES E TRABALHO FUTURO

Os testes demonstram que, utilizando os algoritmos apropriados, o *hardware* actual tem capacidade suficiente para suportar a simulação de ambientes virtuais de grandes dimensões com qualidade gráfica adequada.

A utilização da HLA facilita muito o desenvolvimento deste tipo de aplicações.

A utilização combinada de difusão em grupo e algoritmos de previsão, reduz muito a largura de banda requerida e consegue reduzir a latência percepcionada pelos utilizadores.

No entanto uma dificuldade persiste, a elevada latência das comunicações na Internet [Cheshire96] impõe limitações na fiabilidade que se consegue obter, já que mesmo usando algoritmos de previsão, é impossível esconder atrasos elevados na comunicação, pois as acções de utilizadores remotos são imprevisíveis mesmo estando constrangidas por limites físicos dos objectos simulados.

No futuro, é ainda necessário efectuar testes da aplicação com um número elevado de máquinas distribuídas geográficamente.

6. AGRADECIMENTOS

Agradecemos ao INESC os meios disponibilizados para a realização deste projecto.

7. REFERÊNCIAS

[ARB99] OpenGL Architecture Review Board. OpenGL® 1.2 Programming Guide, 3rd edition. Addison Wesley, 1999.

[Aronson97] Aronson, J. Dead-Reckoning: Latency Hiding for Networked Games. Gamasutra. September 1997.
http://www.gamasutra.com/features/19970919/aronson_01.htm

[Assarsson99] Assarsson, U. and Möller, T. Optimized View Frustum Culling Algorithms. Technical Report 99-3, Department of Computer Engineering, Chalmers University of Technology, March 1999.

[Assarsson00] Assarsson, U. and Möller, T. Optimized View Frustum Culling Algorithms for Bounding Boxes. Journal of Graphics Tools 5(1):9-22, 2000.

[Cheshire96] Cheshire, S. Latency and the Quest for Interactivity. November 1996.
http://www.stuartcheshire.org

[Deering89] Deering, S. E. Host extensions for IP multicasting. RFC1112, Aug-01-1989.

[DMSO98] Defense Modeling and Simulation Office. High Level Architecture Interface Specification, Version 1.3. U.S. Department of Defense, April 1998.
http://hla.dmso.mil

[DMSO00] Defense Modeling and Simulation Office. HLA RTI 1.3-Next Generation Programmer's Guide Version 3.2. U.S. Department of Defense, September 2000.

[Duchaineau97] Duchaineau, M. et. all. ROAMing Terrain: Real-time Optimally Adapting Meshes. IEEE Visualization 1997, 81-88.

[Garland97] Garland, M. and Heckbert, P. Fast Triangular Approximation of Terrains and Height Fields. Carnegie Mellon University, May 2 1997.

[Heckbert97] Heckbert, P. and Garland, M. Survey of Polygonal Surface Simplification Algorithms. Carnegie Mellon University, May 1 1997.

[Hoppe96] Hoppe, H. Progressive meshes. Proceedings of SIGGRAPH 1996, 99-108.

[Hoppe97] Hoppe, H. View-dependent refinement of progressive meshes. Proceedings of SIGGRAPH 1997, 189-198.

[Hoppe98] Hoppe, H. View-dependent level-of-detail control and it's application to terrain rendering. IEEE Visualization 1998, 35-42.

[Lindstrom96] Lindstrom, P. et. all. Real-time, continuous level of detail rendering of height fields. Proceedings of SIGGRAPH 1996, 109-118.

[Macedonia95a] Macedonia, M. A Network Software Architecture for Large Scale Virtual Environments. Doctor's Thesis, Naval Postgraduate School, June 1995.

[Macedonia95b] Macedonia, M., Brutzman D., Zyda, M. et. all. NPSNET: A Multi-Player 3D Virtual Environment over the Internet. Proceedings of the ACM 1995 Symposium on Interactive 3D Graphics, 9-12 April 1995.

[Mills92] Mills, D. Network Time Protocol (Version 3) specification, implementation and analysis. RFC1305. University of Delaware. March 1992.
http://www.eecis.udel.edu/~mills/ntp.htm

[Mills96] Mills, D. Simple Network Time Protocol (SNTP) Version 4 for IPv4, IPv6 and OSI. RFC2030. University of Delaware. October 1996.

[Pope89] Arthur R. Pope. The SIMNET network and protocols. Technical Report 7102, BBN Systems and Technologies, Cambridge, MA.

[Singhal95] Singhal, S. and Cheriton, D. Exploiting position history for efficient remote rendering in networked virtual reality. PRESENCE: Teleoperators and Virtual Environments 4(2):169-193, Spring 1995.

[Singhal97] Singhal, S. Effective Remote Modeling in Large-Scale Distributed Simulation and Visualization Environments. *Doctor's Thesis*, Stanford University, 1997.

[Zyda99] Singhal, S. and Zyda, M. Networked Virtual Environments: Design and Implementation. ACM Press and Addison Wesley, 1999.

8. APÊNDICE

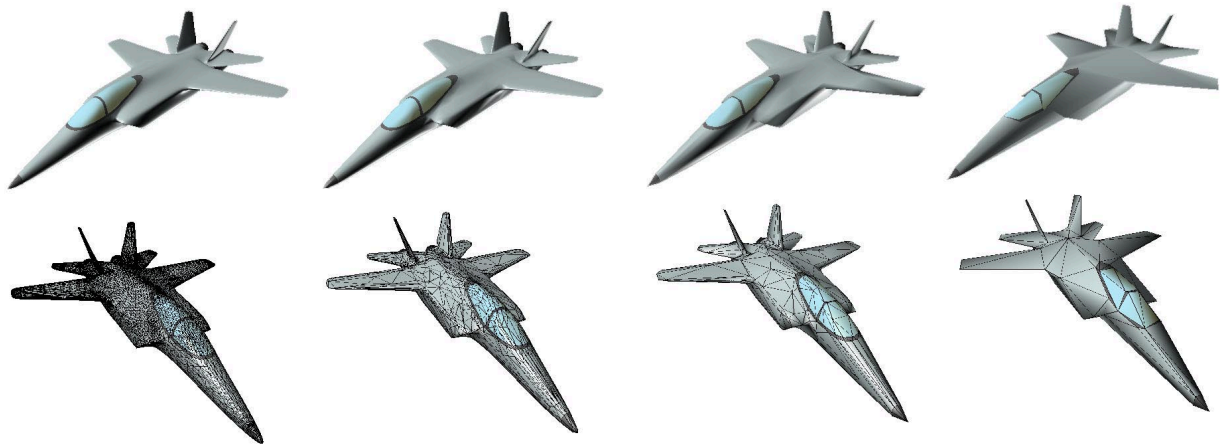


Figura A - Da esquerda para a direita: modelo original (30860 faces); simplificado até 3160 faces; simplificado até 800 faces; simplificado até 200 faces.

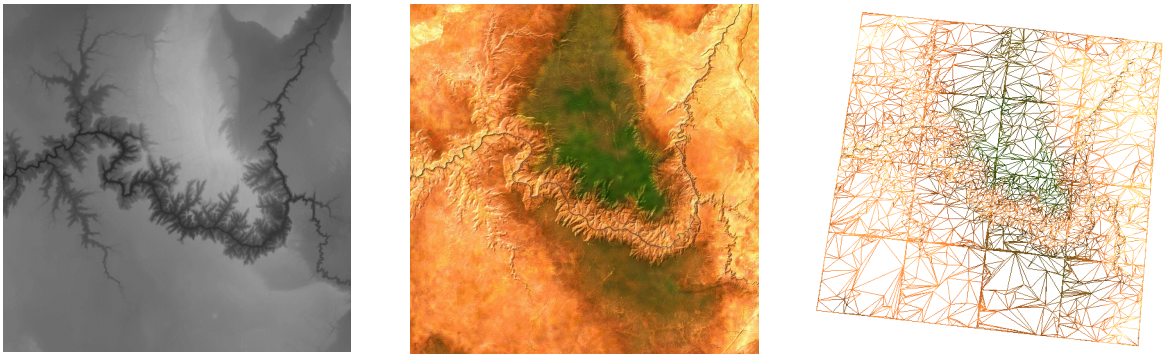


Figura B - Terreno com 2045x2045 pontos. Da esquerda para a direita: dados de elevação; textura; malha com 6000 triângulos.

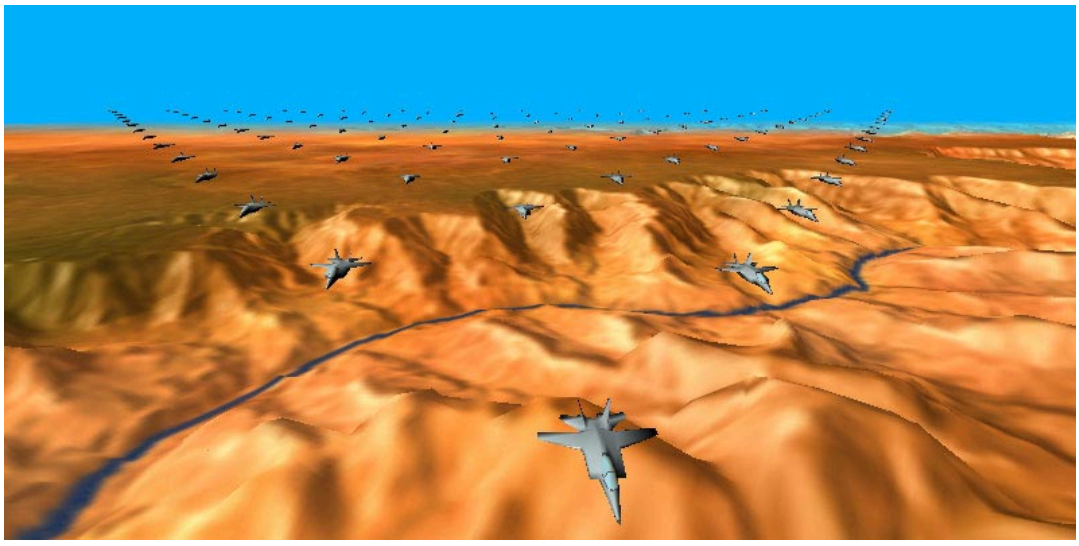


Figura C - Cena com 105 caças.

10º Encontro Português de Computação Gráfica
1-3 de Outubro 2001

Construção e gestão da complexidade de cenários urbanos 3D em ambientes virtuais imersivos

João Pimentel, Nuno Batista, Luís Goes, José Dionísio
Secção de Ambiente e Energia
Instituto Superior Técnico, Av. Rovisco Pais 1, 1049-001 Lisboa
jjdd@ist.utl.pt
http://visualis.ist.utl.pt

Sumário

Os cenários urbanos tridimensionais são, por excelência, uma das áreas que mais desafios coloca aos sistemas de visualização interactiva. Mais ainda, são muitas vezes, a base de trabalho para panóplias de aplicações, desde o planeamento urbanístico, sistemas de navegação automóvel, simuladores de catástrofes, impacte ambiental e meteorologia, turismo, educação, entre tantas outras. Todavia, todas estas aplicações têm um factor em comum: a diversidade e a quantidade de dados geométricos e imaginológicos. Neste trabalho, pretende-se fazer uma reflexão sobre várias vertentes da gestão da complexidade, apontam-se alguns dos maiores desafios em cada uma das fases de desenvolvimento e comentam-se as estratégias utilizadas. O ambiente de trabalho manipulado reporta ao tecido urbano da cidade de Lisboa, um exemplo vivo da dinâmica de uma cidade do ponto de vista de expansão do edificado, e ao seu não menos importante modelo digital de terreno, já que é conhecida pela cidade das "sete colinas". O modelo existente é um dos mais complexos de entre os conhecidos na comunidade científica.

Palavras-chave

Realidade Virtual, Cidades Virtuais, Visualização em Tempo Real, Interfaces Homem-Máquina

1. INTRODUÇÃO

Os cenários urbanos tridimensionais são, por excelência, uma das áreas que mais desafios coloca aos sistemas de visualização interactiva. A sua elevada complexidade geométrica, a liberdade dada ao utilizador de variar as perspectivas de visibilidade de grande pormenor para panorâmica, o acréscimo do foto-realismo a partir da utilização de imagens de alta qualidade, quer para as fachadas dos edifícios, zonas arborizadas ou coberto vegetal, até à fotografia aérea para texturização do modelo digital de terreno, colocam um sem número de constrangimentos à capacidade do *hardware* existente, só passíveis de serem ultrapassados com um planeamento prévio rigoroso dos elementos disponíveis e das capacidades a oferecer aos seus utilizadores. No presente caso, pretende-se a disponibilização ao grande público, num espaço físico apropriado para o efeito, do actual tecido urbano da cidade de Lisboa, e da sua dinâmica evolutiva, através da incorporação de novos projectos e intervenções urbanísticas.

Este cenário pelos requisitos apontados, materializa-se numa aplicação com projecção estereoscópica (no caso passiva) e uma interface gráfica suportada por periféricos de interacção para escolha das características dos objectos a manipular.

O âmbito deste trabalho não é a especificação do ambiente, a arquitectura de visualização e interacção, ou o seu aspecto visual final, mas tão somente a apresentação de um sub-conjunto das estratégias e alternativas que se foram colocando ao longo do desenvolvimento dos trabalhos. Todavia, muitas outras opções poderiam ter sido tomadas e vários outros

Figura 1 – Los Angeles Virtual

trabalhos relevantes foram já apresentados com motivações e objectivos idênticos (fig. 1), embora com níveis de complexidade relativamente distintos.

2. LISBOA 3D

O projecto Lisboa 3D apoia-se fundamentalmente em dois tipos de ferramentas: aquelas que auxiliam o tratamento de dados e construção dos modelos virtuais em pré-processamento, e as que são utilizadas para os manipular em tempo real. Em primeiro lugar, aborda-se a validação dos dados provenientes dos sistemas de informação geográfica; em seguida, como efectuar a exportação desses dados para o ambiente virtual em causa; posteriormente, explica-se como gerar o modelo virtual a partir da informação exportada; faremos também referência a algumas das optimizações utilizadas, com vista a uma simplificação na complexidade do modelo, e um consequente aumento na *performance* de visualização do mesmo, e finalmente, disserta-se acerca das interfaces associadas à manipulação dos modelos na cena.

2.1 O Desafio

O maior desafio é a gestão das bases de dados, no seu sentido mais lato. Entende-se por gestão, a especificação e implementação das várias estruturas de dados, das características geométricas ou não, dos vários objectos, e o seu comportamento dinâmico durante o atravessamento da árvore de *rendering*, permitindo o detalhe necessário para cada operação, inclusivé detecção de colisões, para operações de grande plano, ou técnicas de *occlusion culling* e *frustum culling* para navegação panorâmica ou a baixa altitude.

Na verdade, o modelo reporta a cerca de 120 Km^2 de àrea texturizada, com fotografia aèrea digital, e a um número de pontos altimétricos superior a 8 milhões, disponíveis em memória. O modelo de terreno está dividido em 252 quadrículas, cada com dimensões de 800mx500m e com texturas de resolução variável, tipicamente de 256x256 para zonas de grande edificado e 512x512 para praças principais ou zonas mais a descoberto. Para cada uma destas configurações base existem configurações intermédias que, dinamicamente, vão dando lugar às anteriores, consoante os requisitos da cadeia de visualização. O modelo comporta igualmente cerca de 200.000 edifícios (fig. 2), igualmente com fachadas texturizadas, em grupos de 6 a 8 fachadas típicas, divididas por 37 zonas da cidade de tipologia distinta. A configuração base suporta 6 *pixels* por metro de altura das fachadas.

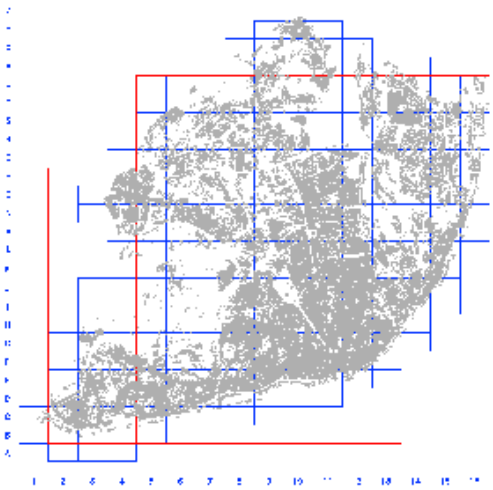


Figura 2 – Mapa de densidade de edificado

Finalmente, há a acrescentar as mais de 100.000 ocorrências de coberto vegetal localizado, essencialmente árvores, que se fizeram representar por algumas das texturas mais representativas das existentes da cidade. A sua contribuição para a carga do sistema reside fundamentalmente no seu elevado número e não na complexidade da textura.

Dado o cenário de partida, e para os requisitos e propósitos definidos no início do desenvolvimento da aplicação, pretende-se atingir o mínimo de 15*f/s* com todas as capacidades de interacção com o modelo, mantendo o elevado foto-realismo, coerência visual e temporal, em ambiente de visualização imersiva e interacção natural, tridimensional e em tempo real.

2.2 Aquisição e Validação de Dados

Como base para a construção do modelo "Lisboa Virtual", foram necessários diversos tipos de dados, criados e manipulados por equipas de arquitectos e engenheiros de várias especialidades. Desta heterogeneidade nasceu o primeiro problema: como tirar partido das ferramentas por eles utilizadas para filtrar, validar e exportar os dados que nos eram relevantes?

Para melhor esclarecer os passos a partir dos quais foi possível gerar "Lisboa Virtual", apresenta-se, de seguida, o esquema da sequência de tarefas, o qual será explicado ao longo das próximas secções (fig. 3).

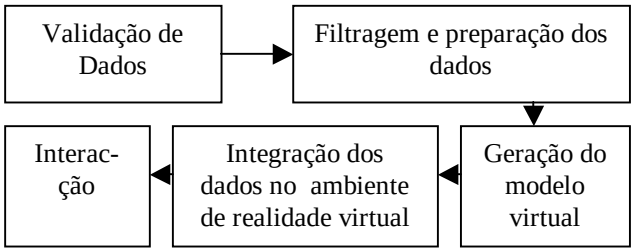


Figura 3 – Esquema de tarefas

Os dados originais oriundos dos SIG consistiam, basicamente, em conjuntos de mapas a duas dimensões com o posicionamento dos objectos urbanísticos, descrições muito limitadas da geometria dos objectos, e mapas de elevação do terreno sob a forma de pontos altimétricos. Numa primeira fase de avaliação, foram suprimidas outras primitivas como equipamentos sociais (candeeiros de rua) entre outros. A adição destes últimos só terá cabimento numa estratégia de pormenor local e numa fase posterior ao levantamento da tipologia da cidade. Para a consequente modelação tridimensional do

relevo do terreno, foram utilizados pontos altimétricos e curvas de nível, num pormenor equivalente a uma carta 1:1000.

Em relação ao processo de edificação, houve necessidade de recorrer a dados sobre:

- geometria das bases dos edifícios (*layout*)
- alturas dos edifícios relativamente ao nível do mar
- relevo do solo (malha de polígonos)

Existe uma relação clara de precedência entre o relevo do solo e os edifícios, que obriga a que o relevo já esteja construído, para situar correctamente em altura, a base de cada edifício.

Outro conjunto de dados utilizados no modelo, referem a localização exacta das árvores na cidade:

- conjunto de pontos 2D (cada um, indicativo da posição de uma árvore no modelo)
- relevo do solo (malha de polígonos)

2.2.1 Validação

A validação dos dados tem um papel fundamental no processo, pois a qualidade dos mesmos nem sempre é a melhor. Principalmente nos dados respeitantes ao relevo e aos edifícios, a correcção dos erros é um passo importante para a credibilidade dos futuros modelos. A correcção é feita manual e empiricamente[1], verificando a coerência entre diversos tipos de informação do SIG. Para correcção de erros no relevo, por exemplo, procede-se a uma análise de isosuperfície, e procuram-se declives muito acentuados, ou mudanças súbitas de cota como possíveis consequências de erros nos dados. Quando são identificados, procede-se à sua análise com sobreposição de ortofotomapas e procura-se nestes uma razão para os declives e alturas suspeitos. Se a razão existir, os dados são mantidos; caso contrário, os dados erróneos são eliminados ou alterados para os seus valores correctos.

2.3 Geração do Modelo Tridimensional

A partir dos dados validados e consoante o seu tipo, são-lhes associadas funções de geração tridimensional automática (ex. extrusão a partir de uma linha de contorno e da sua cota mais elevada), que têm como objectivo a modelação tridimensional dos objectos a que os dados se referem. Cabe à aplicação desenvolvida, caso não necessite de todo o detalhe proporcionado pelo ambiente SIG, proceder à simplificação dos dados, antes da aplicação das funções de modelação. As simplificações de dados permitem a obtenção de modelos de menor complexidade, o que aporta grandes reduções ao nível de armazenamento em memória e tempo de processamento e visualização.

[1] Agora isso é apenas necessário na fonte, pois o processo de importação de dados foi entretanto consideravelmente automatizado, permitindo ao mesmo tempo a correcção automática de pequenos erros nos dados.

2.3.1 Modelo digital de terreno (MDT)

A geração das malhas reparte-se entre as ferramentas SIG e os pré-processadores do ambiente de realidade virtual, sendo as primeiras, responsáveis pelas tarefas de tratamento de dados e geração de versões preliminares das malhas, e os segundos, pela atribuição das coordenadas de textura. A construção das malhas tridimensionais envolve uma maior complexidade se o relevo estiver dividido em quadrículas individuais, pois a geração dos modelos tridimensionais de cada uma delas deve ter em conta a exigência final de encaixe perfeito nas fronteiras de quadrículas adjacentes. Esta divisão tem, todavia, algumas vantagens aliciantes: permite uma utilização mais eficaz de técnicas de *culling*, uma vez que para uma determinada posição da câmera virtual, basta desenhar as partículas cujo volume envolvente intersecte o volume de visualização, evitando-se assim o desenho de toda a malha. A divisão aporta ainda vantagens tais como uma maior rapidez em testes de colisões e de *point-in-polygon*, devido ao decréscimo do número de polígonos a ser testado. Há ainda a acrescentar, as vantagens que advêm do uso de uma estrutura modular, a qual torna possível o carregamento de pequenas porções do modelo na memória, sendo bastante útil para o aumento da *performance* de visualização.

VRML foi o formato intermediário escolhido entre os dados gerados pelo SIG e o motor de realidade virtual. O uso de VRML neste processo justifica-se, devido à sua sintaxe relativamente simples e portabilidade entre as várias aplicações. Embora tenha sido necessário a criação de ferramentas de *parsing* que pudessem importar/exportar dados em VRML, o tempo de desenvolvimento das aplicações é suficientemente curto face aos ganhos de interface entre sistemas e pessoas. Desta forma é possível, em qualquer fase do desenvolvimento, partilhar os vários sub-modelos entre as várias equipas e especialistas que, independentemente da plataforma de trabalho que possuem, conseguem emitir pareceres sobre a qualidade do modelo geométrico.

As malhas de terreno, depois de exportadas, são submetidas a um processo de regularização. Efectua-se uma amostragem da malha irregular, com passo constante, ao longo de dois eixos, obtendo-se a altura do terreno nesse ponto, e criando desta forma uma nova malha. A regularidade de espaçamento dos pontos, resultado do varrimento da malha antiga (irregular), facilita a criação de *stripes*, e, numa fase mais avançada, promove a utilização de níveis de detalhe (*LOD*'s) dinâmicos. Estes tópicos serão abordados com maior detalhe na secção de optimizações.

2.3.2 Edificado 3D

Como já foi descrito anteriormente, os dados relevantes para a construção dos edíficios são um conjunto de polígonos representando a sua implantação no terreno e um conjunto de pontos representando a sua altimetria.

Para cada ponto altimétrico deve ser determinada a base que lhe corresponde, ficando estabelecida a relação ponto-base que permite a modelação do edífício por varrimento espacial da base até à sua altura. Para determinar a que polígono pertence um ponto, é disparado um raio tendo como direcção o eixo vertical e sentido de cima para baixo. A cada base intersectada, corresponderá um edífício, sendo a sua altura indicada pelo ponto.

Após a geração do edifício, é verificada a geometria da sua base. Se esta corresponder a uma forma rectangular, ou quadrangular, é gerado um telhado com duas ou quatro águas respectivamente. Todos os restantes casos (e devido ao facto da determinação das águas não ser trivial), são submetidos a uma função de *striping*, que recebe como dados um polígono, e cujo resultado é um grupo de *stripes* coplanares (sendo no melhor caso um único stripe por telhado); o aspecto final destes casos não triviais para o utilizador, resume-se ao de um terraço.

2.3.2.1 Determinação da altura correcta para a base

Fazer o varrimento espacial da base, até ao ponto de altitude correspondente, seguindo uma trajectória no eixo positivo vertical, pode originar um efeito de "floresta" subterrânea, devido à não planaridade do terreno.

As desvantagens que advêm deste efeito, prendem-se com a desvirtuação dos propósitos das técnicas de *culling* (teste de visibilidade de superfícies parcialmente ocultas), e na dificuldade do mapeamento de texturas sem distorção (o volume visível terá de apresentar portas e janelas consentâneos com as escalas dos modelos, para além de a porta dever estar na base visível da textura – fig. 4).

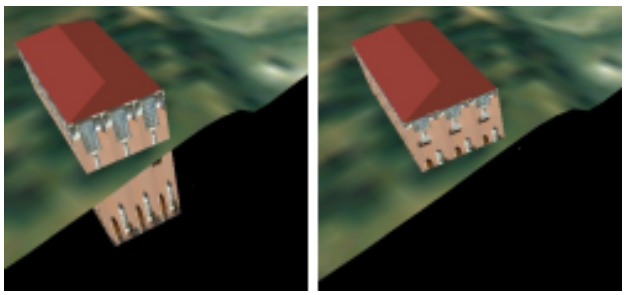


Figura 4

A determinação da altura da base de um edífício recorre à malha de relevo da partição onde a base se encontra e ao registo da menor altura entre os vértices da base. Como o solo pode ser irregular, para se evitar que as arestas fiquem no ar, deve ser subtraída uma margem de segurança à menor altura dos vértices.

2.3.2.2 Materiais e mapeamento de texturas

As bases poligonais exportadas pelo SIG apresentam, normalmente, grande detalhe. Consequentemente, após extrusão, obtêm-se modelos geométricos com um número de polígonos elevado e incomportável para o *hardware* existente. Para evitar a construção de modelos complexos devem ser aplicadas simplificações às bases antes de se proceder à extrusão. As optimizações utilizadas foram de dois tipos:

1. unificação de vértices nas bases: eliminação de arestas inferiores a um valor pré-determinado. Para tal unificam-se vértices que se encontrem entre si a uma distância menor que um dado valor;
2. remoção de arestas co-lineares: redução do número de arestas utilizado para aproximar bases de edífícios com formas curvílineas, através da remoção de arestas que sejam aproximadamente co-lineares.

A técnica de mapeamento de texturas implementada passa por duas fases: a escolha da textura e o mapeamento da mesma. A aplicação tem associado a cada textura um escalão de altura, determinado pela altura aproximada da fachada representada. Para escolher uma textura para um determinado edífício, é necessário classificar esse edífício segundo a mesma escala de altura. Essa classificação é feita através da sua altura ao solo (p. ex. prédios com menos de 3 metros pertencem ao escalão 1). Uma vez escolhido o escalão de altura, é aleatoriamente seleccionada uma textura desse escalão para ser atribuída ao edífício. No caso do modelo "Lisboa Virtual", o processo de selecção aleatório de uma textura, passa em primeiro lugar, por identificar a que zona de Lisboa pertence o edifício. Lisboa encontra-se dividida em "zonas características", ou seja, áreas onde as fachadas dos edifícios são similares (é de notar que estas zonas não coincidem necessariamente com as quadrículas individuais (de terreno) existente, pelo que as primeiras podem cobrir desde fracções de quadrículas até várias quadrículas inteiras do modelo). No final, a selecção de uma textura, passa por identificar a zona onde se situa o edifício, e a escolha aleatória de uma fachada, num grupo de fachadas específicas dessa zona, tendo em conta o escalão a que o edifício pertence. No mapeamento de texturas, a atribuição das coordenadas de textura é feita de forma a não repetir a textura verticalmente, e repetir horizontalmente o número de vezes que produza menos distorção. Para calcular o número de réplicas horizontais é calculado o rácio entre a altura do edífício e o número de *pixels* da textura e define-se esse número respeitando o mesmo rácio horizontalmente.

2.3.3 Cobertura vegetal

A informação do SIG utilizada para a modelação das árvores são os pontos de localização das árvores no plano horizontal, não existindo informação sobre a altura de implantação das árvores, diâmetro das suas copas, ou tipo de árvore.

Após a determinação da localização da árvore sobre o relevo é construído um modelo pseudo-tridimensional representativo de uma árvore, com altura e copa

aleatórios, mas sempre dentro de parâmetros pré-definidos para a zona e classe de árvore.

Os modelos escolhidos para representar as árvores são constituídos por duas faces, intersectando-se perpendicularmente, mapeadas com texturas indexadas e utilizando transparências. O objectivo foi o de alcançar uma representação de árvores com aspecto aparentemente tridimensional, mas com uma geometria bastante simples, técnica bastante vulgar em visualização em ambiente CAD.

3. NAVEGAÇÃO E INTERACÇÃO: UMA SOLUÇÃO OU UM DUPLO PROBLEMA?

Basicamente, a problemática nesta classe de complexidade, reside no facto de o mesmo modelo poder proporcionar ao utilizador o detalhe do edifício ou equipamento ao nível do solo e, sem alteração forçada de configuração do modelo, poder ter a perspectiva de avião no instante seguinte. Mais ainda, na perspectiva de detalhe, o mesmo utilizador deveria não apenas poder apenas desfrutar do foto-realismo elevado mas, igualmente, poder editar e alterar, em tempo real, qualquer estrutura existente. A duplicação em memória de vários modelos, cada um com particularidades específicas do nível de acesso do utilizador aos objectos 3D, ou a flexibilidade da estrutura de dados que os representa e a sua adaptabilidade aos vários níveis de acesso possíveis são a motivação para uma estratégia de optimizações geométricas sem perda de funcionalidades de interacção e manipulação directa.

O termo "qualidade aceitável" surge espontaneamente como correspondendo às necessidades primárias de uma sessão interactiva: conseguir manter uma taxa de refrescamento de desenho do modelo[2] [fps] pelo menos superior ou igual a 15. Escolhida a alternativa do controle dinâmico de geometria no *pipeline* de *rendering*, gerindo em tempo real a complexidade efectiva da porção do modelo que é efectivamente desenhada, desenvolveram-se alguns algoritmos para vistas de grande plano e vistas paronâmicas, respectivamente, LOD´s dinâmicos (níveis de detalhe dinâmico) e *occlusion culling*.

A divisão do espaço do modelo de forma rígida, com recurso a extensivo trabalho de pré-processamento *offline* [Aliaga et al.], também é uma técnica com o objectivo de reduzir a complexidade efectiva do modelo. Basicamente, a diferença entre ambas as alternativas, reside no facto de uma decidir o que é que pode ser visto num dado instante, indicando ao motor de visualização aquilo que ele pode efectivamente desenhar, e a outra, tendo em conta vários parâmetros dinâmicos, define qual a melhor versão da geometria a utilizar para o desenho do modelo.

[2] *Frame rate* (em inglês), da qual deriva a unidade: *(F)rames (p)er (s)econd*

As dificuldades principais associadas à primeira técnica consistem em decidir aquilo que efectivamente deve ser desenhado e como usar critérios de avaliação efectivos que permitam apoiar uma tal decisão em tempo real; em relação à segunda técnica, podemos apontar como principais dificuldades o facto de exigir replicação de informação em memória e de ser necessário que os vários graus de detalhe existentes mantenham a coerência junto das fronteiras das várias partes do modelo.

É nexte contexto que a decisão recaíu sobre a utilização de uso de *LOD*'s dinâmicos[3], onde a exigência de replicação de geometria em memória é substituída pelo cálculo da mesma à medida que esta é precisa, bem como de efectuar os necessários ajustes entre as fronteiras de LOD's de múltiplas quadriculas (fig. 5), com resoluções variadas[4].

Contudo, a aplicação prática de tal conceito levanta vários problemas:

- como organizar a informação geométrica de forma a que possa haver uma maneira prática e rápida de calcular a geometria necessária em cada instante?
- como fazer com que tal processo não influencie de maneira catastrófica o ciclo principal de *rendering*?
- como determinar qual o grau de precisão necessário para uma dada geometria, num dado instante?

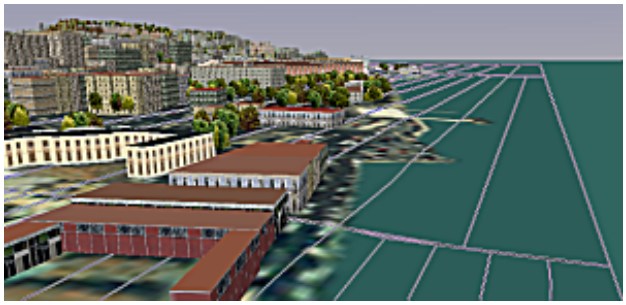


Figura 5

Em relação à primeira questão, no caso do modelo digital de terreno, foi utilizada uma malha regular, de retícula quadrada, e de sub-divisões da mesma para a obtenção dos vários níveis de detalhe - o uso de uma malha quadrada regular traz algumas vantagens sobre malhas irregulares: a determinação dos pontos de fronteira é consideravelmente mais simples e menos susceptível a falhas, e é garantida uma distribuição uniforme dos pontos da mesma. Se bem que existam

[3] Dinâmicos no sentido em que são calculados em *runtime*, antes do início de cada sessão da aplicação, podendo ser facilmente configuráveis através da alteração de parâmetros existentes num ficheiro de configuração, não implicando por isso uma geração *offline* consideravelmente demorada...

[4] [Schmalstieg et al.] introduzem o conceito de Smooth LODs, que é semelhante a esta abordagem.

certos pormenores que as malhas irregulares conseguem retratar com maior precisão, os ganhos obtidos com essa construção compensam amplamente tal perda, em particular a maior facilidade de manipulação e cálculo de detalhe da mesma[5].

A utilização de tal técnica em elementos como árvores e edifícios não faz grande sentido, pois este tipo de elementos é caracterizado por ser de natureza discreta, em oposição ao terreno, que é de natureza contínua. A discussão em relação ao que acabam por ser os vários níveis de detalhe em elementos deste tipo pode resumir-se no facto de poderem ser particionados, e na escolha das partições resultantes nos correspondentes níveis de detalhe respectivos.

A resposta à segunda questão é mais complicada: se bem que o uso de malhas regulares facilite o cálculo e ajuste das várias geometrias necessárias, e acabe por ser relativamente rápido em virtude da regularidade das mesmas, efectuar tais operações durante o ciclo principal de *rendering* acaba por penalizar fortemente o desempenho do sistema e de modo bastante visível para o utilizador. Uma aproximação possível ao problema é a paralelização dos vários procedimentos envolvidos. Apesar de, para um equipamento de mono-processador, os ganhos se limitarem a evitar que o ciclo de rendering pare por completo durante o cálculo, o uso desta técnica em máquinas com múltiplos processadores pode trazer ganhos efectivos como uma maior fluidez de navegação (traduzida numa maior taxa de desenho de imagens por segundo).

Todavia, nem todos os motores de *rendering* dos sistemas de Realidade Virtual contemplam esta variante - múltiplas sequências de execução (*execution threads*) em simultâneo, o que traz alguns efeitos secundários indesejáveis tais como: elementos desenhados que ainda se encontravam em construção, com a respectiva impressão de falha no desenho ou terminação prematura da aplicação de visualização, devida a erros causados por inconsistência nos dados ou por picos de acesso à memória, entre outras causas. Nesse caso, o melhor a fazer será dotar o sistema de mecanismos de sincronização que permitam realizar o trabalho necessário, sem no entanto apresentarem os efeitos secundários mencionados. A técnica de *time slicing* – a reserva de um certo intervalo de tempo no ciclo principal de rendering no qual os cálculos necessários possam ser efectuados, não por um, mas por vários processadores, pode trazer ainda desvantagens no sentido que pode não conseguir evitar congestionamento nos acessos à memória, implicando potenciais terminações inesperadas do sistema (a imagem fica "engasgada"). Po outro lado, a técnica de *double buffering*, (basicamente a gestão da réplica aligeirada da geometria envolvida nos cálculos, em que os cálculos são efectuados sobre a cópia não visível e os seus resultados são disponibilizados ao sistema, para que este os mostre, à medida que os mesmos vão sendo concluídos) também encerra em si a desvantagem inerente à replicação, a ocupação de espaço em memória. Todavia é possível que tanto a cópia como o original partilhem certas caracteristicas, tais como materiais comuns ou tabelas de pontos comuns, como forma de minimizar a carga do sistema.

A resposta à terceira questão acaba por ser semelhante à questão que o *occlusion culling* levanta:

- como avaliar e decidir o que realmente é necessário mostrar num dado instante ?

Uma possibilidade é a aplicação de heurísticas que determinam qual o nível de detalhe a aplicar a um dado elemento do modelo, em função, por exemplo, da distância do ponto de vista da câmera ao mesmo. A solução não é fácil e, em última análise, dependerá do modelo em causa e da liberdade de acção que o utilizador deverá ter.

4. OPTIMIZAÇÃO

A enorme quantidade de informação que constitui o modelo, conduz a elevados tempos de carregamento e a taxas de refrescamento da imagem que afectam a interactividade do modelo. Torna-se indispensável complementar o tratamento e validação dos dados de origem com uma simplificação da representação geométrica desses componentes, de forma a torná-la o mais compacta e estruturada possível.

Igualmente importante para a optimização do modelo, é a concepção de uma estrutura hierárquica dos seus componentes, que potencie um atravessamento para *renderização* o mais eficiente possível. Para esse efeito a organização hierárquica deve ter em conta o funcionamento dos motores de *renderização* a que se destinam. Nesta secção abordam-se as várias técnicas utilizadas para aumentar o desempenho e assegurar a componente "tempo real" do modelo.

4.1 Geometria do edificado

Uma das consequências do primeiro processo de extrusão e da necessidade de texturizar as faces verificou-se ser a partilha de posição entre vários pontos com características incompatíveis (atributos de cor, textura). Esta replicação de pontos com a mesma posição, tornava os ficheiros de geometria de uma dimensão elevada, sendo dispendiosos em termos de memória e de carregamento. Tornava-se imperativo partilhar pontos, tendo para isso que ser contornada a necessidade de informação distinta entre os pontos de iguais coordenadas. A solução encontrada para realizar um processo de extrusão em que, em vez de cada parede ser tratada independentemente, as paredes são tratadas

[5] Existem maneiras mais eficientes de organizar malhas de terrenos, sendo um exemplo *quad-trees*, que para o caso em questão – multiplas divisões da geometria do terreno já existentes – dificultam a sua aplicação prática, pois em geral partem do pressuposto de que a geometria é constituída por um só bloco...

como uma malha texturizada com pontos partilhados, foi a atribuição de coordenadas de textura a cada ponto, de forma a torná-las válidas para todas as faces que o possuem. Para esse efeito, durante a criação de uma face, a atribuição da coordenada de textura do novo ponto introduzido por essa face, é atribuída relativamente à parede anterior.

4.1.1 Remoção de faces partilhadas

Na organização espacial dos objectos físicos a modelar, verificou-se existir uma característica comum nos edifícios, a sua organização espacial (ao longo de ruas) sem intervalos entre si. Prédios adjacentes possuem por isso paredes que se sobrepõem, muitas ficando ocultas a qualquer observação por parte do utilizador. Estas paredes, apesar de escondidas, continuam a gastar recursos sem adicionar melhorias à visualização do modelo. Sendo as faces não visíveis irrelevantes para o modelo[6], a sua eliminação apenas apresenta vantagens. Para esse efeito é necessário determinar quais as faces de edifícios nunca visíveis: o lançamento de raios a partir dos vértices de topo de cada face na direcção da normal da face examinada, testando intersecções com os edificios vizinhos, foi a forma escolhida para identificar as faces candidatas a eliminação. Este procedimento permitiu reduzir o número de faces por quadrícula em quase 10% revertendo esse decréscimo a favor da taxa de refrescamento da imagem.

4.1.2 Utilização de stripes

Os telhados dos edifícios são, para edifícios de geometria complexa, formados por um número elevado de polígonos. O grau de complexidade prende-se com a restrição, imposta por parte do motor de *rendering*, ao uso de polígonos convexos, sendo os polígonos concavos divididos até apenas restarem polígonos convexos. A solução encontrada para reduzir o impacto dos telhados a nível de desempenho foi adoptar a utilizacão de *stripes* de polígonos. Este tipo de primitiva pela forma como organiza os dados de polígonos adjacentes permite obter n polígonos por cada n+2 pontos, proporcionando um acréscimo de desempenho atavés da reutilização de pontos já transformados. Para fazer o *stripping* dos telhados foi desenvolvido um algoritmo que, partindo do conjunto de pontos que formam um telhado, gera um conjunto de construções, sendo escolhida aquela que menor número de *stripes* crie.

Por último, a organização do terreno sobre a forma de malha regular (ver 2.3.1) facilita a sua construção em *stripes*, o que proporciona um acréscimo de desempenho, proveniente da partilha de pontos entre várias "faces" do *stripe* e da consequente poupança em termos de transformações de pontos efectuadas.

[6] são irrelevantes para a navegação do modelo, o que já não é verdade quando falamos na edição do mesmo...

4.2 Hierarquia do Edificado

Os objectos da cena virtual, são organizados hierarquicamente numa estrutura tipo árvore - o grafo da cena. Em cada ciclo de refrescamento, a raiz do grafo é passada ao motor de *rendering* e este atravessa toda a árvore, desenhando os objectos nela contidos.

Após a criação dos edifícios pelo processo de extrusão anteriormente descrito, o grafo da cena organizava-se da seguinte forma: debaixo do nó raiz, um nó pai por cada quadrícula tendo como filhos tantos nós terminais quantos os edifícios nela contidos, estando armazenada nos nós terminais a informação geométrica dos edifícios. Apesar de ser vantajoso atribuir a cada edifício o estatuto de objecto individual, principalmente a nível de interactividade, a organização formada pelo processo de extrusão revelava-se prejudicial, tanto por gerar árvores pouco balanceadas, prejudicando o seu atravessamento, como por o estatuto de objecto a cada edifício implicar um grande dispêndio de memória a nível das estruturas necessárias à atribuição de tal categoria. Que estratégia seguir: continuar a permitir interacção ao nível do edifício com todos os edifícios (mais de 200.000), ou visar um acréscimo de desempenho utilizando uma abordagem menos direccionada aos objectos fisicos e mais virada para as potencialidades e restrições do algoritmo de *renderização*, sendo ainda possível a interacção ao nível do edifício por reatribuição da sua hierarquia. Decidida a remodelação hierárquica do grafo da cena, optou-se por manter a estrutura de quadrículas individuais já mencionada em outras secções, relegando a optimização a operações intra-quadrícula.

Cada edifício possui tipicamente dois materiais: o material aplicado às suas fachadas e o material aplicado ao seu telhado; existem assim duas mudanças de material por cada edifício *renderizado*, sendo o número de mudanças de material por quadrícula dado pela expressão:

(nº edifícios da quadrícula * 2 mudanças de material por edifício)

A solução implementada para evitar tantas mudanças de material consistiu em reunir todos os edifícios com o mesmo material debaixo do mesmo nó pai. Esta opção apesar de permitir a existência de apenas uma mudança de material por cada material de fachada presente na quadrícula, tem o inconveniente de efectuar várias mudanças de material, por os telhados dos vários edifícios de fachadas idênticas poderem ter materiais distintos.

Para resolver este problema foi atribuído o mesmo material de telhados a todos os edifícios de mesmo material de fachada, obtendo-se desta forma o número efectivo de mudanças de material:

(nº de materiais de fachadas na quadrícula * 2 mudanças de material por material de fachada)

Apesar do aumento de *performance* atingido para a generalidade das situações de visualização (até 20% de aumento da *frame rate*), da poupança de memória e do aumento da velocidade de carregamento dos dados, a reordenação hierárquica da cena implicou como efeito secundário uma perda de *frame rate*, para visualizações de poucos edifícios. Esse facto explica-se por a distribuição dos edifícios de um determinado material abranger toda a quadrícula. Quando, por exemplo, o utilizador visualiza um edifício de um determinado material, o algoritmo de *frustum culling* valida para *renderização* todos os edifícios da quadrícula que contêm esse material.

5. INTERACÇÃO E OPTIMIZAÇÃO

O desenvolvimento de aplicações eficazes em geral, e particularmente em ambientes de realidade virtual, envolve a optimização da interacção entre o utilizador e a aplicação, de forma a permitir a concentração deste último em tarefas de alto nível e a maximização do seu desempenho na utilização da aplicação.

As aplicações de realidade virtual são geralmente de interacção complexa, e frequentemente providas de soluções de interacção muito dedicadas e com metáforas reconhecidamente de ambientes 2D e desktop.

Uma tão grande dificuldade na idealização e implementação de interfaces eficazes pode dever-se à dificuldade que um utilizador possui em interagir com um mundo tridimensional onde são permitidos um número de graus de liberdade superior aos habituais noutro tipo de aplicações, que utilizam 2D tradicionais para navegação. Nesta secção abordam-se várias opções tomadas para assegurar um nível de usabilidade que permita uma interacção eficaz com o modelo.

Como se pretende simular um mundo tridimensional em que supostamente o utilizador se encontra imerso, uma forma de facilitar essa abstração ao utilizador, é pelo desenvolvimento de ferramentas de interação baseadas em metaforas, de forma a que seja possível transpôr as acções realizadas sobre os objectos do mundo virtual, para acções realizadas no mundo real. Esta abordagem alêm de reforçar a credibilidade do modelo, tem como virtude permitir ao utilizador recorrer a modelos mentais já formados e refinados sobre o mundo real, para compreender o sistema.

Das várias tarefas a realizar destacam-se a navegação através do mundo tridimensional, bem como operações a selecção, posicionamento, orientação e alteração de objectos do mundo. As soluções propostas para as várias tarefas foram desenvolvidas tendo como objectivo a sua incorporação no projecto. No entanto, o seu âmbito estende-se a qualquer outra aplicação com cenários tridimensionais complexos.

5.1 Navegação

Um problema muito comum na navegação em mundos tridimensionais, deve-se à típica extensa dimensão destes, existindo muitas formas dos utilizadores se desorientarem em relação á sua posição em cada instante. Uma forma possível de lidar com este problema, é restringir o número de graus de liberdade da câmara virtual, aos estritamente necessários para simular uma observação real. Outra possibilidade é a de incorporar elementos na cena, que permitam ao utilizador perceber qual a sua posição no mundo, como marcas, objectos especiais ou mesmo uma representação gráfica do percurso anterior do utilizador.

A estratégia escolhida para navegação através do modelo urbano virtual baseia-se na metáfora do tapete voador. O utilizador deve imaginar que se encontra a sobrevoar o mundo como que sobre um tapete voador dos contos das mil e uma noites. São permitidas as translacções segundo um referencial XYZ com uma limitação intencional das rotações a dois eixos (Roll, Pitch), para minorar a desorientação espacial.

Como suporte à navegação e à percepção pelo utilizador da sua posição e orientação, foram incorporados no modelo um conjunto de âncoras visuais formado por vários edifícios históricos, que se destacam dos restantes, por possuirem características geométricas especiais e elevado foto-realismo. O mesmo princípio foi aplicado a zonas de baixa edificação ou praças principais.

5.2 Operações Sobre os Objectos da Cena

A aplicação que serviu de base à implementação destas técnicas compôs-se de um mundo tridimensional contendo uma malha poligonal irregular[7] representando o solo (desenvolvendo-se em XoZ) sobre a qual existem objectos (edifícios) criados. Como ferramenta de interacção com os elementos da cena, o utilizador dispõe de uma mão virtual, posicionável e orientável espacialmente.

- Construção

Para construir um novo objecto, é projectado um raio a partir do centro da mão em direcção à malha poligonal (direcção vertical, sentido de cima para baixo). As coordenadas do ponto de intersecção do raio com a malha do solo serão as coordenadas do centro da base do objecto.

- Selecção

Esta tarefa consiste na escolha de um elemento a partir de um conjunto de objectos seleccionáveis sobre a malha poligonal. A selecção de objectos em mundos tridimensionais é uma das tarefas mais habituais e, devido a esse facto, uma das mais estudadas e avaliadas. Várias propostas têm sido submetidas a avaliação e, geralmente, podem ser agrupadas segundo a metáfora utilizada, a da "mão virtual" ou a do "ponteiro virtual".

Na metáfora da "mão virtual" o utilizador selecciona um objecto "tocando-lhe" com uma representação virtual da sua mão. A "mão virtual" é posicionada e orientada pelo

[7] tratou-se da primeira versão da geometria do terreno; foi entretanto substituída por uma malha regular

utilizador através de um dispositivo de entrada. Na metáfora do "ponteiro virtual" o utilizador selecciona um objecto apontando para o mesmo, ou seja, é lançado um raio na direcção que se encontra a mão, raio esse que selecciona um objecto quando o intersecta.

[Poupyrev et al.] efectuaram um estudo de usabilidade das duas metáforas na selecção de objectos. A principal conclusão retirada desse estudo indica que não existe uma técnica que seja universalmente superior. Embora as técnicas baseadas na metáfora do "ponteiro virtual" exibissem melhores resultados nos casos mais gerais, o seu desempenho degradava-se bastante quando o tamanho dos objectos diminuía. Em relação às técnicas baseadas na metáfora da "mão virtual" verificou-se que o seu desempenho não se degradava muito com a dimensão dos objectos, mas globalmente tinham um desempenho um pouco inferior às outras técnicas. Para tentar aproveitar as vantagens relativas das duas técnicas bem como para minimizar as suas desvantagens, propõe-se uma nova técnica, a da "selecção por aproximação".

Esta técnica selecciona o objecto que se encontra mais próximo da mão. Isto resulta num decréscimo substancial do tempo médio de selecção em relação à técnica da "mão virtual", devido ao facto de não ser necessário ao utilizador colidir com o objecto desejado para o seleccionar. Optou-se ainda por definir uma distância máxima de selecção descartando objectos que se encontrem muito distantes do utilizador.

Um problema desta abordagem é o de, em alguns casos, poder ser seleccionado um objecto não pretendido. Este problema surge principalmente na selecção de objectos muito grandes, devido ao facto da distância ser calculada em relação ao centro de inércia do objecto. A figura 6 exemplifica esse erro em duas dimensões: embora a mão se encontre dentro do objecto A, a distância ao centro de inércia do objecto B é menor do que a distância ao centro de inércia do objecto A. Assim, o algoritmo seleccionaria o objecto B, quando o utilizador pretendia seleccionar A.

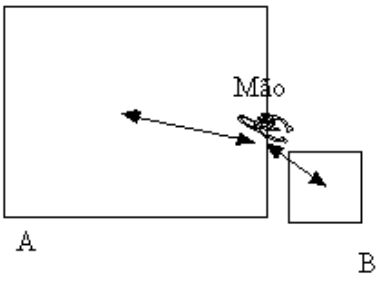


Figura 6

A "selecção por aproximação" não invalida que se efectue um teste prévio semelhante ao da "mão virtual". O algoritmo proposto representa, no seu conjunto, uma melhoria substancial em relação às técnicas base. Consegue-se assim uma selecção de objectos rápida e um aumento de eficácia no uso da aplicação. Outro aspecto importante é a retroacção: como comunicar ao utilizador que o objecto se encontra seleccionado? Apesar de inicialmente se optar pela representação do objecto em "modelo de arame", a ambiguidade deste tipo de representação (nomeadamente na visualização da topologia do objecto) depressa conduziu à opção de mapear o objecto com um material semi-transparente (fig. 7), com resultados bem mais positivos, pois permite visualizar perfeitamente tanto a topologia do objecto como a sua inserção no meio que o rodeia.

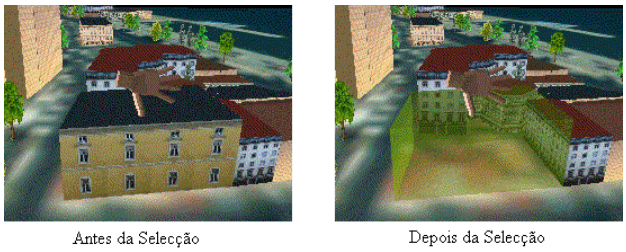


Figura 7

5.3 Operações de Manipulação e Transformação

Uma vez seleccionado um edifício, a aplicação permite ao utilizador a realização de um conjunto de tarefas como translacções, rotações, escalamento, alteração da geometria, e a mudança de atributos como cor e textura.

As tarefas de translacção, orientação e escalamento mereceram uma atenção especial, por exigirem controlo tridimensional sobre os objectos. A solução proposta, baseia-se na possibilidade de controlar o posicionamento utilizando apenas duas dimensões, sendo a terceira dimensão sempre automaticamente definida pela altura do solo. Não sendo relevante para o posicionamento do objecto a altura a que a mão se encontra, deixa de ser necessário que o utilizador consuma recursos cognitivos no controlo dessa dimensão, focando-se exclusivamente no controlo das restantes e melhorando o seu desempenho.

Devido à possibilidade de realização com sucesso da mesma tarefa, independentemente da “altitude” da mão do utilizador, deixava de ser necessário existir contacto entre a mão virtual e o objecto manipulado. Para evitar este comportamento pouco natural, foi incorporada uma pega extensível em altura aos edifícios seleccionados, que estabelece uma ligação "física" virtual entre o objecto e a mão que o manipula, tal como uma azeitona na ponta dum palito. A posição do edifício em altura é independente da posição da mão, ajustando-se automaticamente á altura da malha do terreno nessa posição.

As expectativas ao nível da interacção e funcionalidades entregues ao utilizador foram quase todas atingidas, tendo sido atingido um nível de usabilidade que permite a um utilizador não perito efectuar de forma eficiente um

conjunto vasto de tarefas ligadas à construção e alteração de objectos em cenários virtuais.

6. CONCLUSÕES E TRABALHO FUTURO

Não é tarefa simples tentar sistematizar as contribuições parciais de cada módulo na cadeia de visualização. Todavia, a perspectiva e o ênfase dado à contribuição relativamente descritiva dos vários processos, cuja mais valia não é a excelência de um ou mais algoritmos, não é sequer uma análise radical e inovadora no sentido do desenvolvimento de metodologias de processamento de dados em tempo real, mas que se pode resumir em três linhas fundamentais de acção:

- a preocupação do conceito de generalidade da metodologia em causa; as técnicas usadas não são dependentes do modelo Lisboa Virtual, antes se podem aplicar a modelos de complexidade elevada e com os fins similares de visualização e interacção em tempo real
- a automatização parcial do processo de construção do modelo; sempre que possível, e módulo a módulo, tentou-se desenvolver técnicas de optimização focadas em cada tipo de objectos e suas características
- o controlo do balanceamento de cargas entre o *CPU* e o *GPU*; seja pela diminuição das mudanças de estado entre trocas de materiais, pela partilha de pontos entre objectos do mesmo tipo, geração dinâmica de *bounding boxes* e níveis de detalhe, entre outras.

Finalmente, uma referência à existência do modelo tridimensional, independentemente da sua optimização para visualização em tempo real, foi que a disponibilidade física da aplicação permitiu, pela primeira vez, validar visualmente de uma forma integrada e sistemática, todo o conjunto de dados, e a análise de erros pela troca de *layers*, pontos falsos de cota de terreno e edificado, análise de densidade espacial, entre outras.

7. AGRADECIMENTOS

Uma última palavra de apreço pelo apoio prestado, a toda a equipa do projecto Lisboa Virtual do IST/SAE e da CML/DEIU, que não é possível enunciar extensivamente, na pessoa dos responsáveis por este projecto, respectivamente a srª vereadora drª Margarida Magalhães (CLM/DEIU) e o prof. Delgado Domingos (IST/SAE).

8. BIBLIOGRAFIA

[Dodge98] Dodge, Martin et al. Towards the Virtual City: VR & Internet GIS for Urban Planning, May 1998

http://www.casa.ucl.ac.uk/publications/birkbeck/vrcity.html

[Goodfellow96] Goodfellow, David. Collaborative Urban Design through Computer Simulations, April 1996

http://www.fes.uwaterloo.ca/u/gbhall/research/davidweb/HMTHESIS.html

[Haines94] Haines, Eric. Point in Polygon Strategies, Graphic Gems 1994

[Hand97] Hand, Chris. A Survey of 3D Interaction Techniques, Computer Graphics Forum, 16 (5): 269-281. December 1997

[Knopfle2000] Knopfle, Christian. Interacting with Simulation Data in an Immersive Environment. Data Visualization 2000, Symposium on Visualization, Amsterdam, The Netherlands, May 29-31, 2000

[Koller95] Koller, David et al. Virtual GIS: A Real-time 3D Geographic Information System, Proceedings Visualization´95, pp. 94-100, 1995

[Poupyrev98] Poupyrev, I. et al. Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation for Interaction Techniques. Eurographics‘98

[Poupyrev96] Poupyrev, I. et al. The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. ACM 96

[Reddy99] Reddy, Martin et al. Terravision II: Visualizing Massive Terrain Databases in VRML, March 1999

http://www.computer.org/cga/cg1999/g2030abs.htm

[VrCom2000] VrCom. Virtual Design 2 Programmer´s Guide, February 2000

[Weiler94] Weiler, Kevin. An Incremental Angle Point in Polygon Test, Graphics Gems, 1994

[West93] West, A.J. et al. AVIARY – A Generic Virtual Reality Interface for Real Applications

http://citeseer.nj.nec.com/west88aviary.html

[Wonka2000] Wonka, Peter et al. Visibility Preprocessing with Occluder Fusion for Urban Walkthroughs

http://www.cg.tuwien.ac.at/research/vr/occfusion/

[Schmalstieg et al.] Schmalstieg, Dieter; Schaufler, Gernot. - Smooth Levels of Detail http://www.cg.tuwien.ac.at/research/vr/smoothlods/

[Aliaga98] Aliaga, D. ; Cohen, J. ; Wilson, A. et al. – A Framework for the Real-Time Walkthrough of Massive Models - UNC TR # 98-013 http://www.cs.unc.edu/~hoff/papers/mmr/i3d99.pdf

[Sillion97] Sillion, François, Drettakis, George, Bodelet, Benoit. Efficient Impostor Manipulation for Real-time Visualization of Urban Scenery. E*UROGRAPHICS 97, Vol. 16, (1997), nr. 3.*

[Pinnel99] Pinnel, L. Denise, Dockrey, Matthew, Brush, A. J. Bernheim, Borning, Alan. Data Visualization 2000, Symposium on Visualization, Amsterdam, The Netherlands, May 29-31, 2000