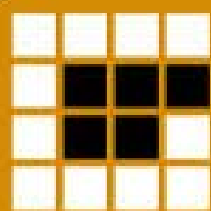


<http://epcg12.isep.ipp.pt>



EPCG

**12º Encontro Português
de Computação Gráfica**

8-10 Outubro, 2003

ISEP - Instituto Superior de Engenharia do Porto

Actas do 12º Encontro Português de Computação Gráfica

**8 – 10 de Outubro 2003
Porto - Portugal**

Patrocinadores



FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA E DO ENSINO SUPERIOR Portugal
**Apoio do Programa Operacional Ciência, Tecnologia, Inovação
do Quadro Comunitário de Apoio III**

Organização

ISEP



GPCG

Grupo Português de Computação Gráfica

Prefácio

O Encontro Português de Computação Gráfica tem lugar este ano, naquela que é a sua 12ª edição, no ISEP – Instituto Superior de Engenharia do Porto, entre os 8 a 10 de Outubro. O 12º Encontro Português de Computação Gráfica vem no seguimento de encontros anteriores realizados anualmente e reúne investigadores, docentes e profissionais nacionais e estrangeiros, que realizam trabalho ou utilizam a Computação Gráfica, Realidade Virtual e Multimédia, assim como todas as suas áreas afins, no sentido de permitir a divulgação de projectos realizados ou em curso e fomentar a troca de experiências e a discussão de questões relacionadas com a Computação Gráfica em Portugal, entre as comunidades académica, industrial e a de utilizadores finais.

A Computação Gráfica assume, nos dias de hoje, o papel de área de conhecimento e tecnologia horizontal das denominadas tecnologias da informação e comunicação, posicionando-se na interface e interacção entre a aplicação computacional e o utilizador final, sempre que se explore a representação gráfica dos dados, processos e aplicações, na forma bi- e tri-dimensional, imersiva, pictórica ou de imagem. Hoje podemos afirmar que a Computação Gráfica é uma peça fundamental do edifício, em contínua construção, que é a Sociedade de Informação.

O 12º Encontro integra este ano, para além da conferência principal, duas *workshops* focando temas mais específicos, que se realizam no dia 8 de Outubro, antes do início dos trabalhos do Encontro, e que reflectem duas outras grandes áreas de actividade do Grupo Português de Computação Gráfica – as aplicações interactivas através do CoopMedia 2003 – 2º *Workshop* em Sistemas de Informação Multimédia, Cooperativos e Distribuídos; e o Ensino de CG através do CGME03 – 2º *Workshop* sobre Computação Gráfica e Multimédia no Ensino. Os trabalhos do 12º Encontro decorrem nos dois dias seguintes, dividindo-se entre cinco sessões técnicas (quatro de comunicações longas; uma de comunicações curtas ou “posters”) e quatro sessões com palestras proferidas por oradores convidados, especialistas de renome, nacionais e internacionais. As quatro sessões técnicas com comunicações longas estão organizadas da seguinte forma: Modelação Geométrica, Multimédia e Visualização Gráfica, Ambientes Virtuais e Realidade Aumentada. A Comissão de Programa avaliou 33 comunicações, tendo sido seleccionadas 14 comunicações longas e 10 comunicações curtas. Em face da elevada qualidade das comunicações seleccionadas e das palestras convidadas, pensamos que mais uma vez valeu a pena o esforço de levar por diante uma nova edição do Encontro Português de Computação Gráfica, sendo possível oferecer aos seus participantes, um programa de elevado interesse e qualidade técnico-científica.

A realização do 12º Encontro Português de Computação Gráfica, à semelhança do sucedido nas anteriores edições, comportou, desde o primeiro momento, o assumir de um risco organizativo, tanto da parte dos organizadores locais como da comissão de programa, que conjuntamente tiveram de garantir a qualidade do programa científico, a logística e o financiamento adequados a uma boa condução dos trabalhos, assim como de todas as actividades associadas. Tal só foi possível no 12º Encontro através do envolvimento generoso e voluntário de uma grande equipa, de várias instituições, para quem é devido um primeiro agradecimento sincero e reconhecido.

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

Aos membros da Comissão de Programa pelo empenhamento colocado no rigor das avaliações e na selecção final das comunicações submetidas. Aos membros da Comissão Organizadora, em particular ao Frederico Figueiredo pelo grande empenho na instalação e bom funcionamento do sistema de gestão de avaliação das comunicações, à Paula Centeno pelo trabalho árduo de compilar e formatar as actas do encontro. Aos conferencistas convidados, Alan Chalmers, Rui Bastos, Graham Thomas e Salvador Marques, a disponibilidade para participarem e contribuírem para o sucesso do 12º Encontro, partilhando a sua experiência, saber e visão. A todos os autores pelo interesse, esforço e empenho colocados na elaboração das suas comunicações. Um agradecimento especial é aqui devido à Fundação Ilídio Pinho pelo apoio que emana de uma saudável atitude proactiva em relação à investigação científica e ensino superior em Portugal e ao Fundo de Apoio à Comunidade Científica da FCT. E finalmente ao ISEP, co-organizador com a APRP e GPCG deste evento, pelo acolhimento e apoio dado a esta realização.

Porto, Outubro de 2003

Adérito Marcos
Ana Mendonça
Miguel Leitão
António Costa
Joaquim Jorge

COMISSÃO ORGANIZADORA

Organização Local: António Costa, Miguel Leitão
Secretariado: Maria João Ferreira
Publicidade: Nuno Correia, Paula Centeno
Tutoriais: Nuno Correia
Patrocínios: António Costa, Miguel Leitão
Sessões Temáticas: Augusto Sousa
Preparação e Impressão de Actas: Joaquim Jorge
Site Web e Artes Gráficas: Miguel Leitão, Vítor Cunha
Conferencistas: Augusto Sousa, António Costa

COMISSÃO CIENTÍFICA

Presidente: Adérito Fernandes Marcos – CCG; DSI/EE/UM
Co-Presidente: Ana Maria Mendonça - FEUP
Abel João Gomes - DI/UBI
Adriano Martins Lopes - DI/FCT/UNL
Augusto Sousa - INESC-Porto; FEUP
Armando J. Pinho – IEETA; UA
Beatriz Sousa Santos – IEETA; UA
F. Mário Martins - DI/EE/UM
Fernando Nunes Ferreira - FEUP
Fernando Pereira – IT; IST/UTL
Francisco Feito - Univ. Jáen, Espanha
Helder Araújo – ISR; UC
João Brisson Lopes – IST/UTL
João Duarte Cunha – LNEC; FCUL
João Luís Monteiro - DEI/EE/UM
João M. Pereira – INESC; IST/UTL
João Paulo Costeira – ISR; IST/UTL
Joaquim A. Jorge – INESC; IST/UTL
Joaquim Madeira – IEETA; UA
Jorge Alves da Silva - FEUP
Jorge Salvador Marques – ISR; IST/UTL
José Carlos Teixeira - DM/UC
José Dionísio – IST/UTL
José Fernandes de Almeida – DSI/EE/UM
José Miguel Salles Dias - ADETTI; ISCTE
Juan Carlos Torres - Univ. Granada, Espanha
Luís Corte-Real - INESC-Porto
Luís Paulo Santos - DI/EE/UM
Maribel Y. Santos - DSI/EE/UM
Márcio Lobo Netto - USP, Brasil
Manuel Gamito – ADETTI; ISCTE
Manuel Próspero dos Santos - DI/FCT/UNL
Miguel Encarnação - CRCG, USA
Miguel Pimenta Monteiro - FEUP
Miguel Velhote Correia - FEUP
Mário Rui Gomes – INESC; IST/UTL
Nuno Correia - DI/FCT/UNL
Pedro F. Lopes – ADETTI; ISCTE
Rafael Bidarra - Univ. Téc. Delft, Holanda
Rogério Caldas Pinto – IST/UTL
Vasco Branco - UA

Programa

Sessão 1

Modelação Geométrica

<i>AIF - Uma estrutura de dados B-rep para modelos poligonais</i>	
Frutuoso Silva, Abel Gomes	1
<i>Representação BSP de Curvas Implícitas 2D</i>	
Francisco Morgado, Abel Gomes	9
<i>BlobMaker: Free-Form Modelling with Variational Implicit Surfaces</i>	
Bruno Araújo, Joaquim Jorge	17
<i>Web-based direct manipulation of feature models</i>	
Rafael Bidarra, Andre van Bunnik, Willem F. Bronsvoort	27

Sessão 2

Multimédia e Visualização Gráfica

<i>VGLib2D - Biblioteca de classes para a visualização gráfica de grandes volumes de dados em 2D</i>	
Sofia Gameiro, Luís Almeida, Adérito Marcos	35
<i>Vídeo Digital para Multimédia: Boas Práticas de Aquisição e Processamento</i>	
Pedro Lopes, Maria Moreira, Noel Santos	43
<i>Varição Dinâmica do Nível de Detalhe na Simulação Estável de Tecidos</i>	
Fernando Birra, Manuel Santos	53

Sessão 3

Ambientes Virtuais

<i>Computação Gráfica e Arqueologia Urbana: o caso de Bracara Augusta</i>	
Paulo Bernardes, Manuela Martins	63
<i>Modelação Expedita de Cenas Urbanas</i>	
António Coelho, A. Augusto Sousa, F. Nunes Ferreira	73
<i>Identificação e reconstrução 3D de objectos numa malha urbana através de nuvens de pontos georeferenciados</i>	
Rodrigo Garcia, Mauro Farracha, José Dionisio, Alexandre Bernardino, João Costeira	83
<i>VRML.... porque não?</i>	
Alexandrino Gonçalves, António Mendes	91

Sessão 4

Realidade Aumentada

<i>VIDA: Visualizador Interactivo de Dados (biomecânicos) Aumentados</i>	
José Braz Pereira, João M. Pereira, António Veloso	101
<i>Image Manipulation through Gestures</i>	
José Dias, Pedro Nande, Pedro Santos, Nuno Barata, André Correia	111
<i>ARK multi-utilizador</i>	
Pedro Pereira, Nuno Matos, Luís Grave, Adérito Marcos	119

Sessão 5

Comunicações Curtas

<i>Técnicas de realidade virtual aplicadas ao ensino na área da construção</i>	
A. Zita Sampaio, Pedro G. Henriques, Pedro S. Ferreira.....	125
<i>Sistema de Realidade Virtual para Tratamento de Fobias</i>	
Vitor Cunha, Miguel Leitão.....	131
<i>Simulação e avaliação de um projecto urbano através dos computadores</i>	
Pedro Leão Neto.....	137
<i>Geração de Ambientes Rodoviários Para Simulação de Condução</i>	
Carlos Campos, Vítor Cunha, Miguel Leitão.....	143
<i>Using Sketches to Edit Electronic Documents</i>	
Pedro Coelho, Gonçalo Faria, Daniel Gonçalves, Joaquim Jorge.....	149
<i>Task Analysis and Scenario-Based Design of Calligraphic Interfaces</i>	
Filipe Dias, Manuel Fonseca, Joaquim Jorge.....	153
<i>Polygon Detection from a Set of Lines</i>	
Alfredo Ferreira Jr., Manuel Fonseca, Joaquim Jorge.....	159
<i>Corte de Peças de Pedra: Uma Aplicação da Computação Gráfica</i>	
Samuel Silva, Filipe Marques, Manuel Raposo, Joaquim Madeira, Beatriz Sousa Santos.....	163
<i>GIDes ++</i>	
Joaquim Jorge, Nelson Silva, Tiago Cardoso.....	167
<i>Construção dinâmica de um sistema interactivo para visualização do código intermédio do processo de compilação</i>	
Paulo Matos, Pedro Henriques.....	173

Sessão 1

MODELAÇÃO GEOMÉTRICA

AIF – Uma estrutura de dados B-rep para modelos poligonais

Frutuoso G. M. Silva Abel J. P. Gomes
IT – Grupo de Redes e Multimedia
Dep. Informática, Univ. da Beira Interior
Rua Marquês d'Ávila e Bolama, 6201-001 Covilhã
{fsilva, agomes}@di.ubi.pt

Resumo

Este artigo introduz uma nova estrutura de dados *b-rep* (*boundary representation*) designada de AIF (*Adjacency and Incidence Framework*). A estrutura de dados AIF é concisa e permite o acesso rápido à sua informação topológica. A sua concisão resulta do facto de ser uma estrutura de dados orientável mas não orientada, isto é, a orientação é induzida topologicamente como é necessário em várias aplicações. É uma estrutura de dados óptima na classe C_4^9 , o que significa que um número mínimo de acessos é necessário para aceder a toda a informação sobre adjacências e incidências. A estrutura de dados AIF suporta a representação de modelos poligonais, não necessariamente triangulares, mesmo em condições de “non-manifold”.

Palavras chave

Estruturas de dados, modelos poligonais, representação de sólidos.

1. INTRODUÇÃO

Na computação gráfica, a complexidade dos modelos tridimensionais tem aumentado notoriamente devido em parte às tecnologias de aquisição de dados tridimensionais como os *scanners* 3D. Este tipo de tecnologias produzem modelos poligonais, normalmente designados por malhas poligonais usadas nas mais variadas aplicações como, por exemplo, multi-resolução, ambientes virtuais ou ainda, visualização de sólidos e superfícies. De facto, muitos trabalhos de investigação tem sido feitos nos últimos anos no âmbito de malhas poligonais, particularmente na análise multi-resolução [Heckbert97, Garland99], morfização de malhas [Lee99], edição interactiva de malhas [Zorin97], e na compressão geométrica e transmissão pela Internet [Taubin98].

Aplicações diferentes têm necessidades diferentes. Algumas necessitam mais de memória, outras necessitam mais de acesso rápido à informação sobre adjacências e incidências. Infelizmente, estas operações de acesso nem sempre são rápidas e eficientes como é necessário nalgumas aplicações e estruturas de dados, em particular em malhas com grande número de células (vértices, arestas e faces). Isto deve-se sobretudo ao *design* das estruturas de dados. Por exemplo, encontrar uma célula na estrutura de dados CTR (Cell-Tuple Representation)[Brisson93] requer processar todos os tuplos, o que é obviamente pouco prático e lento em malhas de grandes dimensões.

Algumas estruturas de dados representam uma malha triangular através dum conjunto de faces, sendo cada face composta pelos vértices [Hoppe98]. Este é também o caso do formato VRML (Virtual Reality Modeling Language) [Carey97]. Contudo, esta estratégia torna difícil a implementação de algoritmos rápidos de acesso à informação topológica. Por exemplo, determinar as faces incidentes num vértice é uma operação lenta porque é necessário percorrer todas as faces da estrutura de dados.

Uma forma de acelerar os algoritmos de procura é a utilização de estruturas de dados orientadas (Winged-Edge [Baumgart72], Half-edge [Mantyla88], ou Radial Edge [Weiler88]). No entanto, estas estruturas de dados necessitam de mais memória para armazenar as células orientadas. Por exemplo, na estrutura de dados Radial Edge, cada face tem associados dois conjuntos de arestas (loops), um para cada lado da face; além disso, se houver três faces incidentes numa aresta, são necessárias seis arestas orientadas.

Lee e Lee [Lee01] propuseram uma nova estrutura de dados orientada, designada por *partial entity structure*, para modelos não *manifold*, que não é mais que uma optimização da estrutura Radial Edge que necessita de menos memória. Esta estrutura contém para além das entidades vértice, aresta, face, *loop* e *shell* outro tipo de entidades, designadas de *partial entity*, para representar os casos não *manifold*. Assim contém as entidades *partial vertex*, *partial edge* e *partial face* para

representar as situações não *manifold* para vértices, arestas e faces, respectivamente.

Kallmann e Thalmann [Kallmann01] propuseram uma nova estrutura de dados, chamada Star-vertices, para representar malhas poligonais. Esta estrutura de dados é baseada na incidência de células à volta de um vértice. É uma estrutura de dados concisa mas lenta no acesso à informação topológica, pois não armazena informação explícita sobre arestas e faces.

Outras estruturas de dados foram desenhadas apenas para malhas triangulares como, por exemplo, a Directed-edges [Campagna98], a Tri-edges [Loop00] e mais genericamente a PSC (Progressive Simplicial Complexes) [Popovic97]. As duas primeiras são estruturas de dados orientadas, enquanto que a PSC é não-orientada. Portanto, a PSC não possui quaisquer células orientadas. Consequentemente, a PSC permite representar objectos orientados e não orientados, mas ao contrário das estruturas de dados orientadas (b-reps), não existe redundância na estrutura de dados PSC.

Este artigo introduz uma nova estrutura de dados b-rep, chamada AIF (Adjacency and Incidence Framework), para malhas genéricas. Estas malhas não necessitam de ser *manifold* nem triangulares. Topologicamente falando, a estrutura de dados AIF é orientável mas não orientada, isto é, não possui células orientadas. O resultado é uma estrutura de dados mais concisa e flexível, sem perda de controlo da informação sobre adjacências e incidências.

Este artigo está organizado em várias secções: A secção 2 descreve a estrutura de dados AIF. A secção 3 apresenta o operador de navegação da estrutura de dados AIF, designado por operador máscara, e que permite um acesso rápido à informação topológica. A secção 4 apresenta a implementação da estrutura de dados AIF. A comparação entre diversas estruturas de dados, incluindo a estrutura de dados AIF, é apresentada na secção 5. Por último, a secção 6 apresenta algumas conclusões e trabalho futuro.

2. ESTRUTURA DE DADOS AIF

Diz-se que uma célula x é adjacente a uma célula y (simbolicamente $x \prec y$) se x está contido na fronteira de y (ou equivalentemente, $fr(y) \cap x \neq \emptyset$) e a dimensão de x é menor que a dimensão de y ; equivalentemente, diz-se que y é incidente em x (simbolicamente, $y \succ x$). Por exemplo, um vértice de uma aresta diz-se que lhe é adjacente, mas podem existir várias arestas incidentes no mesmo vértice. A relação de adjacência (incidência) é transitiva, isto é, se $v \prec e$ e $e \prec f$ então $v \prec f$.

Além disso, existem apenas duas relações básicas de adjacência para malhas poligonais, nomeadamente $V \prec E$ e $E \prec F$; as relações inversas $E \succ V$ e $F \succ E$ são as relações de incidência. Estas quatro relações podem ser compostas para formar nove relações de adjacência introduzidas por Weiler [Weiler85]. Por exemplo, as relações $V \rightarrow F$ e $E \rightarrow E$ podem ser obtidas do seguinte modo:

$$V \rightarrow F = (V \rightarrow E) \circ (E \rightarrow F) = (E \succ V) \circ (F \succ E)$$

$$E \rightarrow E = (V \rightarrow E) \circ (V \rightarrow E) = (V \prec E) \circ (E \succ V)$$

Assim, as quatro relações referidas acima constituem uma representação na classe C_4^9 (veja-se Ni e Bloor [Ni94] para mais detalhes). Esta é a melhor representação em termos do número de acessos à informação topológica, isto é, requer um número mínimo de acessos directos e indirectos à estrutura de dados para aceder às quatro relações explícitas e às restantes cinco relações implícitas, respectivamente. Um acesso directo requer uma simples chamada ao operador máscara, enquanto que um acesso indirecto requer duas ou mais chamadas ao operador máscara. O operador máscara é descrito na secção seguinte.

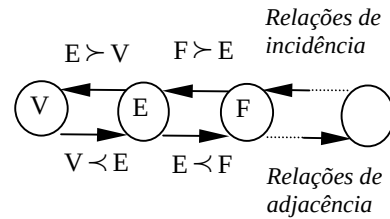


Figura 1: Estrutura de dados AIF .

A estrutura de dados AIF acomoda uma representação óptima na classe C_4^9 para malhas bidimensionais, mas pode ser facilmente generalizada para $C_{2n}^{(n+1)^2}$ quando se pretende representar objectos de dimensão n (Figura 1). A estrutura de dados AIF mantém a informação essencial sobre as adjacências e incidências de células, mas permite inferir informação adicional de modo a atravessar qualquer malha duma forma eficiente.

Uma malha bidimensional é definida em termos de um conjunto $M = \{V, E, F\}$, onde V é um conjunto finito de vértices, E é um conjunto finito de arestas, e F é um conjunto finito de faces simplesmente conexas.

Um vértice $v \in V$ é definido por um conjunto de arestas incidentes em v , isto é, $v = \{e_1, e_2, e_3, \dots, e_k\}$, onde e_i ($i=1 \dots k$) é uma aresta incidente em v . Esta é a forma como a relação $E \succ V$ está representada na AIF.

Uma aresta $e \in E$ é um par de conjuntos, o primeiro para os vértices adjacentes à aresta, e o segundo para as faces incidentes na aresta, isto é, $e = \{\{v_1, v_2\}, \{f_1, f_2, \dots, f_k\}\}$, onde v_1 e v_2 são os vértices da aresta, e $f_1, f_2, \dots, f_k$ são as faces incidentes na aresta. Assim, temos as relações $V \prec E$ e $F \succ E$ também representadas na AIF.

Uma face $f \in F$ é definida em termos de um conjunto de arestas que lhe são adjacentes, ou seja, $f = \{e_1, e_2, e_3, \dots, e_k\}$, onde cada e_i ($i=1 \dots k$) é uma aresta adjacente a f . Esta definição inclui a relação $E \prec F$.

Com estas quatro relações básicas de adjacência e incidência na estrutura de dados AIF podemos representar malhas poligonais *manifold* (Figura 2(a)(c)(d)(e)(f)) e não-*manifold* (Figura 2(b)).

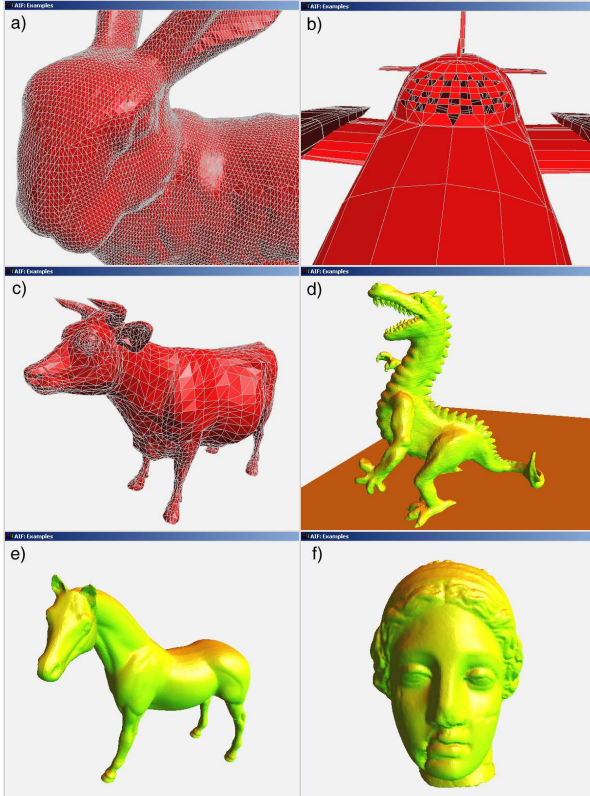


Figura 2: Modelos AIF.

Na Figura 2(b) o modelo do avião é não-*manifold* porque faltam algumas faces na zona do cockpit (zonas mais escuras).

Note-se que uma aresta pode ter um número qualquer de faces incidentes. Uma aresta e sem faces incidentes é representada por $e = \{\{v_1, v_2\}, \{\}\}$, isto é, uma aresta remanescente. Uma aresta e com mais do que duas faces incidentes (Figura 3) é representada através dos seus vértices adjacentes e das faces incidentes, ou seja, $e = \{\{v_1, v_2\}, \{f_1, f_2, f_3\}\}$. Assim, a estrutura de dados AIF permite-nos distinguir entre objectos *manifold* e não-*manifold*.

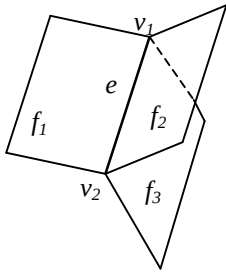


Figura 3: Malha não-*manifold*.

Além disso, a estrutura de dados AIF suporta malhas poligonais genéricas, quer sejam triangulares (Figura 2(a)(c)(d)(e)(f)) ou não (Figura 2(b)).

3. O OPERADOR MÁSCARA

O tempo é um factor crítico nas aplicações de tempo real (e.g. animação e jogos) que fazem uso de malhas poligonais. São necessárias pesquisas e algoritmos rápidos para identificar localmente as células adjacentes

ou incidentes noutras células de modo a rapidamente simplificar ou refinar as ditas malhas.

A estrutura de dados AIF usa um único operador, chamado operador máscara, para aceder a toda a informação sobre adjacências e incidências. O operador máscara é definido por: $\triangleright_d: V \times E \times F \rightarrow C$, com $C = V \cup E \cup F$ sendo V o conjunto dos vértices, E o conjunto de arestas, e F o conjunto de faces, tal que $\triangleright_d(v_i, e_j, f_k) = \{c_l^d\}$ tem como resultado um conjunto de células de dimensão d . Os argumentos do operador máscara são células do conjunto $V \times E \times F$. A célula NULL como argumento de dimensão $n=d$ significa que todas as células de dimensão n que satisfaçam as condições de adjacência/incidência expressas pelos argumentos são devolvidas como resultado. Caso contrário, se $n \neq d$ e a célula de dimensão n é também NULL, então nenhuma restrição de adjacência/incidência é imposta ao resultado. No caso de $n=d$ e a célula de dimensão n é diferente de NULL, o operador máscara devolve todas as células de dimensão n como anteriormente, à excepção da célula argumento de dimensão n . Se $n \neq d$ e a célula de dimensão n é diferente de NULL, então a célula de dimensão n coloca uma restrição adicional de adjacência/incidência ao resultado.

A chamada do operador máscara é feita pelo menos com um argumento não nulo, podendo ter dois ou mesmo os três argumentos não nulos. Qualquer combinação de argumentos é possível. No entanto, no caso da chamada do operador índice zero com apenas o primeiro argumento diferente de NULL, significa que este devolverá os vértices vizinhos do vértice argumento. Assim, se pretendermos saber os vértices do vértice, as arestas da aresta ou as faces da face o operador devolver-nos os vértices, arestas ou faces adjacentes ao argumento respectivamente.

Consideremos a malha da Figura 4 para ilustrar o funcionamento do operador máscara em conjunção com a estrutura de dados AIF.

1. $\triangleright_1(v_1, \text{NULL}, \text{NULL}) = \{e_1, e_2, e_3\}$, devolve todas as arestas incidentes em v_1 .
2. $\triangleright_2(v_1, \text{NULL}, \text{NULL}) = \{f_1, f_2, f_3\}$, devolve indirectamente todas as faces incidentes em v_1 . Requer uma chamada intermédia ao operador máscara $\triangleright_1(v_1, \text{NULL}, \text{NULL})$ para devolver todas as arestas e_1, e_2, e_3 incidentes em v_1 . Depois o operador máscara $\triangleright_2(v_1, e_i, \text{NULL})$ é chamado para cada aresta e_i ($i = 1, 2, 3$) de modo a calcular as faces incidentes em e_i e v_1 .
3. $\triangleright_0(\text{NULL}, e_1, \text{NULL}) = \{v_1, v_2\}$, devolve directamente os vértices da aresta e_1 .
4. $\triangleright_2(\text{NULL}, e_1, \text{NULL}) = \{f_2, f_3\}$, devolve directamente as faces incidentes em e_1 .
5. $\triangleright_0(\text{NULL}, \text{NULL}, f_1) = \{v_1, v_3, v_4\}$, devolve indirectamente todas os vértices da face f_1 . Requer uma chamada intermédia ao operador máscara

- $\triangleright\blacktriangleleft_1(\text{NULL}, \text{NULL}, f_1)$ para primeiro determinar as arestas adjacentes a f_1 . Depois o operador máscara é chamado $\triangleright\blacktriangleleft_0(\text{NULL}, e_i, \text{NULL})$ para cada aresta e_i para determinar os vértices da aresta e_i e f_1 .
6. $\triangleright\blacktriangleleft_1(\text{NULL}, \text{NULL}, f_1) = \{e_2, e_3, e_5\}$, devolve directamente todas as arestas adjacentes à face f_1 .
 7. $\triangleright\blacktriangleleft_2(v_3, e_1, \text{NULL}) = \{f_2\}$, devolve directamente as faces incidentes em e_1 e v_3 .
 8. $\triangleright\blacktriangleleft_1(v_1, e_2, f_1) = \{e_3\}$, devolve directamente as arestas adjacentes a f_1 incidentes em v_1 e diferentes de e_2 .
 9. $\triangleright\blacktriangleleft_0(v_4, \text{NULL}, f_1) = \{v_1, v_3\}$, devolve indirectamente todos os vértices da face f_1 diferentes de v_4 . Requer uma chamada intermédia ao operador máscara $\triangleright\blacktriangleleft_1(\text{NULL}, \text{NULL}, f_1)$ para primeiro determinar as arestas adjacentes a f_1 . Depois o operador máscara é chamado $\triangleright\blacktriangleleft_0(v_4, e_i, \text{NULL})$ para cada aresta e_i para determinar os vértices de e_i e f_1 diferentes de v_4 .
 10. $\triangleright\blacktriangleleft_0(v_1, \text{NULL}, \text{NULL}) = \{v_2, v_3, v_4\}$, devolve indirectamente todos os vértices adjacentes a v_1 . Requer uma chamada intermédia ao operador máscara $\triangleright\blacktriangleleft_1(v_1, \text{NULL}, \text{NULL})$ para determinar primeiro as arestas incidentes em v_1 . Depois o operador máscara é chamado $\triangleright\blacktriangleleft_0(v_1, e_i, \text{NULL})$ para cada aresta e_i para determinar os vértices de cada aresta e_i diferentes de v_1 .

Com o operador máscara não existe a necessidade de manipular todas as células da estrutura de dados. Ele permite-nos manipular a malha localmente. Assim, o seu desempenho é independente do tamanho da malha. Isto é muito importante na manipulação de malhas com grande número de células (vértices, arestas e faces), em particular em operações em tempo real.

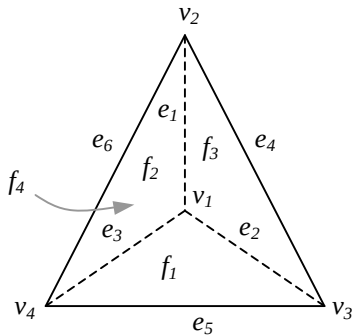


Figura 4: Malha manifold.

4. IMPLEMENTAÇÃO DA ESTRUTURA AIF

A estrutura de dados AIF representa malhas que consistem em conjuntos de células, vértices, arestas e faces. Cada tipo de célula foi codificado através de uma classe em C++, mas também se poderia codificar todos os tipos de células com uma única classe, como é necessário para malhas de dimensão n . Isto é possível porque qualquer célula de dimensão n é vista como um conjunto de células incidentes de dimensão $(n+1)$ e um conjunto de células adjacentes de dimensão $(n-1)$. A

estrutura de dados AIF foi então codificada do seguinte modo:

```
class Vertex {
    evector li;    //arestas incidentes
    Point *pt;    //geometria
}
class Edge {
    Vertex*v1,*v2; //vertices adjacente
    fvector li;    //faces incidentes
}
class Face {
    evector la;    //arestas adjacentes
    Point *nv;    //normal à face
}
class Mesh {
    int ID;        //identificador
    vvector vv;    //vector de vértices
    evector ev;    //vector de arestas
    fvector fv;    //vector de faces
}
```

De notar que a estrutura de dados AIF não é topologicamente orientada, pois não possui células orientadas. Ela é geometricamente orientada pela normal à face na classe **Face**. As normais às faces são determinadas por indução topológica duma orientação consistente nas faces como, por exemplo, a orientação no sentido dos ponteiros do relógio (Figura 5). Isto requer a travessia da fronteira de todas as faces no mesmo sentido. Para isso, aplica-se alternadamente os operadores máscara indexados $\triangleright\blacktriangleleft_0$ e $\triangleright\blacktriangleleft_1$ às arestas e vértices adjacentes a cada face. O operador $\triangleright\blacktriangleleft_0$ retorna o próximo vértice na fronteira da face, e o operador $\triangleright\blacktriangleleft_1$ devolve a próxima aresta adjacente à face. Isto é, percorrendo a malha com o auxílio do operador máscara é possível induzir uma orientação na malha.

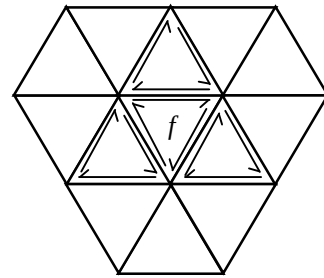


Figura 5: Mecanismo de orientação.

Consideremos a malha da Figura 4 e vejamos como com o auxílio do operador máscara podemos percorrer uma face. Consideremos, por exemplo, a face f_4 . O operador $\triangleright\blacktriangleleft_0(\text{NULL}, \text{NULL}, f_4)$ devolve indirectamente todos os vértices da face f_4 , ou seja, $\{v_2, v_3, v_4\}$. Partindo agora de um vértice qualquer, por exemplo, v_2 podemos chamar o operador máscara $\triangleright\blacktriangleleft_1(v_2, \text{NULL}, f_4)$ que nos devolverá as arestas que contém v_2 e pertence a f_4 , ou seja, $\{e_6, e_4\}$. Agora escolhendo uma das arestas, por exemplo, e_4 podemos chamar de novo o operador $\triangleright\blacktriangleleft_0(v_2, e_4, f_4)$ que nos devolverá o próximo vértice da aresta e_4 diferente de

v_2 na face f_4 , ou seja, v_3 . Depois o operador $\triangleright \triangleleft_1(v_3, e_4, f_4)$ devolve-nos a próxima aresta incidente em v_3 diferente de e_4 na face f_4 , ou seja, e_5 . Continuando este processo podemos percorrer toda face.

5. ESTRUTURA DE DADOS - COMPARAÇÕES

5.1. Memória

A Tabela 1 compara diversas estruturas de dados em termos da memória necessária ao armazenamento de uma malha triangular com n vértices. A coluna “Tipo de malha” designa o tipo de malha suportada pela estrutura de dados, ou seja, triangular (Δ) ou poligonal (Pol.), enquanto a coluna “não *manifold*” indica se a estrutura de dados suporta malhas não-*manifold* ou não. A última coluna “Bytes/ Δ ” designa o número de bytes necessários ao armazenamento de uma face triangular. A variável k designa o grau do vértice, isto é, o número de arestas incidentes no vértice.

A estrutura de dados Triangle List é a estrutura mais concisa porque armazena apenas as faces e os vértices. Normalmente, esta estrutura de dados é usada apenas para visualização visto que o hardware gráfico está optimizado para triângulos.

Estruturas de dados	Tipo de malha	Não <i>manifold</i>	Bytes/ Δ
Triangle list	Δ	sim	18
Star-vertices	Pol.	sim	$10+4k$
Progressive Meshes	Δ	não	33
Tri-edges	Δ	não	35
AIF	Pol.	sim	$29+2k$
PSC	Δ	sim	$37+2k$
Directed-edges	Δ	sim	44
Half-edge	Pol.	não	46
FastMesh	Δ	não	53
Radial edge	Pol.	sim	56
Winged-edge	Pol.	não	60

Tabela 1: Comparação do espaço em memória.

A estrutura de dados Star-vertices apenas armazena vértices e os vértices vizinhos daqueles. A cada vértice vizinho está associado um índice que permite efectuar a navegação à volta dum dado vértice segundo uma ordem pré-definida. É uma estrutura de dados concisa pois apenas necessita de $10+4k$ bytes por triângulo. Além disso, esta estrutura de dados suporta malhas genéricas, mas os acessos à informação sobre adjacências são lentos, pois não inclui arestas e faces explicitamente. A ausência de arestas e faces implica que seis relações de adjacência/incidência não estejam definidas (veja-se na coluna “Não definidas” da Tabela 2).

As estruturas de dados Progressive Meshes [Hoppe98] e Tri-edges [Loop00] são também bastante concisas, visto que necessitam apenas de 33 e 35 bytes por triângulo, respectivamente. No entanto, estas estruturas de dados apenas suportam malhas triangulares *manifold*.

Por outro lado, uma malha triangular com n vértices na estrutura de dados AIF requer aproximadamente 35, 37, 39, 41 bytes por triângulo para $k=3,4,5$ e 6,

respectivamente. De facto, de acordo com a fórmula de Euler para uma malha bidimensional triangular, uma malha com n vértices tem cerca de m faces ($m=2n$) e a arestas ($2m=3a$). Assim, assumindo que um real e um ponteiro ocupam 4 bytes cada, a memória necessária a uma malha triangular na estrutura de dados AIF é: $(3 \times 4 + 4k)n = (12 + 4k)n = (6 + 2k)m$ bytes para as coordenadas dos vértices, $(2 \times 4 + 2 \times 4)e = 16e \approx 11m$ bytes para as arestas (referências aos vértices e faces) e $(3 \times 4)m = 12m$ bytes para as faces (referências às arestas).

Note-se que a estrutura de dados AIF não inclui células orientadas (e.g. half-edges). Por isso é mais concisa que os b-reps tradicionais (e.g. Winged-edge [Baumgart72]) e as suas variantes orientadas (e.g. Half-edge [Mantyla88], Directed-edges [Campagna98], FastMesh [Pajarola01], e Radial Edge [Weiler88]). Por fim, a estrutura de dados PSC [Popovic97] suporta apenas malhas triangulares e requer mais memória que a estrutura de dados AIF.

5.2. Acessos às Adjacências e Incidências

A Tabela 2 mostra-nos o número de acessos necessários para aceder à informação de adjacência e incidência representada em várias estruturas de dados. Visto que temos três entidades topológicas (vértice, aresta e face), logo existem nove relações de adjacência e incidência [Weiler85].

Na Tabela 2 a coluna “Classe” dá-nos o número de relações de adjacência e incidência representadas explicitamente na estrutura de dados. Por exemplo, uma estrutura de dados C_4^9 tem quatro relações explicitamente representadas em nove relações possíveis. Isto é, temos acesso directo a quatro relações sem nenhum processamento adicional. As restantes cinco relações estão representadas implicitamente, visto que todas as estruturas de dados são completas, ou seja, representam malhas sem ambiguidades. Isto significa que temos de ter mecanismos de inferência para aceder a estas cinco relações de adjacência e incidência. A estes mecanismos de inferência chamamos acessos indirectos. Acontece que os acessos indirectos são mais difíceis porque algumas células topológicas não existem nas estruturas de dados, referidas na coluna “Não definidas”. Por exemplo, a estrutura de dados Star-vertices não possui arestas e faces representadas explicitamente; como consequência, seis relações de adjacência/incidência não estão definidas (Tabela 2).

Na Tabela 2 podemos ver que as estruturas de dados AIF e PSC são as mais rápidas em termos do número de acessos. Claro, que a rapidez depende do número de relações de adjacência e incidência explicitamente armazenadas na estrutura de dados, mas quantas mais relações armazenamos mais memória é necessária. Ambas as estruturas de dados AIF e PSC pertencem à classe C_4^9 . De acordo com Ni e Bloor [Ni94], podemos dizer que ambas são estruturas de dados óptimas nesta classe, pois um número mínimo de acessos é necessário para aceder a toda informação sobre adjacências e incidências. Contudo, a estrutura de dados PSC apenas

suporta malhas triangulares e necessita de mais memória (ver Tabela 1).

Estrutura de dados	Não definidas	Classe
AIF	0	C_4^9
PSC	0	C_4^9
Winged-edge	0	C_2^9
Radial edge	0	C_2^9
Half-edge	0	C_2^9
FastMesh	3	C_2^9
Directed-edges	3	C_2^9
Triangle List	6	C_2^9
Star-vertices	6	C_2^9
Progressive Meshes	6	C_2^9
Tri-edge	6	C_2^9

Tabela 2: Comparação do número de acessos.

As outras estruturas de dados pertencem à classe C_2^9 (dois acessos directos e sete acessos indirectos para obter toda a informação sobre adjacências e incidências). Por exemplo, a estrutura de dados FastMesh não possui a entidade vértice. Assim, as três relações de adjacência e incidência envolvendo vértices não estão definidas, podendo, no entanto, ser inferidas com um custo adicional em termos de tempo. As restantes seis relações envolvendo arestas e faces já podem ser obtidas mais facilmente mas apenas duas estão explícitas na estrutura de dados.

5.3. Desempenho da estrutura de dados AIF

O desempenho da estrutura de dados AIF foi testado num PC equipado com um processador Pentium 4, com 768MB de memória, uma placa gráfica Nvidia Riva TNT2, e com o sistema operativo Windows 2000. A Tabela 3 mostra-nos alguns resultados para diferentes modelos AIF (Figura 2).

Malha	#V	#F	Ler o ficheiro	Orientar	Criar primitiva	Desenhar primitiva
Vaca	2904	5804	0.180	0.060	0.020	0.040
Avião	3745	3946	0.180	0.040	0.020	0.030
Avião	6795	13546	0.491	0.130	0.030	0.040
Dragão	25418	50761	1.622	0.661	0.240	0.351
Coelho	34835	69473	2.674	1.001	0.311	0.480
Cavalo	48485	96966	3.374	1.372	0.401	0.651
Venus	50002	100000	3.445	1.442	0.441	0.681

Tabela 3: Tempos de inicialização e rendering em segundos.

A criação de um modelo AIF a partir de um ficheiro de texto inclui também a orientação da malha através do mecanismo de indução topológica. A visualização de um modelo AIF envolve duas etapas:

- a criação da primitiva GL (Graphics Library);
- desenhar a primitiva no ecrã.

Olhando para a Tabela 3 podemos ver que a visualização de malhas *manifold* (ex. Venus, o cavalo e o coelho da Figura 2) ou malhas poligonais não-*manifold* (e.g. avião da Figura 2) é rápida, mesmo considerando que o *rendering* é efectuado por software e não por hardware.

6. CONCLUSÃO E TRABALHO FUTURO

Este artigo introduz uma nova estrutura de dados, chamada AIF, juntamente com um operador de pesquisa para satisfazer três requisitos fundamentais, nomeadamente:

- Generalidade.** A estrutura de dados AIF suporta modelos poligonais e não apenas triangulares, *manifold* e não-*manifold*, mesmo para dimensões superiores.
- Concisão.** Ao contrário das outras estruturas de dados b-rep, a estrutura de dados AIF não é orientada, ou seja, não contém células orientadas. Assim, é mais concisa que os b-rep tradicionais.
- Rapidez.** A estrutura de dados AIF foi concebida para assegurar pesquisas rápidas através de um único operador, o operador máscara. Este permite aceder rapidamente à informação sobre adjacências e incidências independentemente do tamanho da malha.

Na prática, a estrutura de dados AIF e o operador máscara provaram ser poderosos no acesso à informação sobre adjacências e incidências. Em termos de eficiência no acesso à informação sobre adjacências e incidências apresenta a melhor performance na classe C_4^9 tal como a estrutura de dados PSC. No entanto a estrutura de dados AIF necessita de menos memória e suporta malhas poligonais genéricas ao contrário da estrutura de dados PSC.

No futuro esperamos que a estrutura de dados AIF possa ser utilizada em diferentes tipos de aplicações devido à sua generalidade, desempenho e concisão como, por exemplo, na multi-resolução, nos sistemas de animação e realidade virtual.

AGRADECIMENTOS

Os modelos utilizados são cortesia de:

Cyberware (<http://www.cyberware.com>) - Cavalo e Venus; Stanford University (<http://graphics.stanford.edu>) - Coelho; Viewpoint Digital Inc. (<http://www.viewpoint.com>) - Avião; Avalon archive (<http://avalon.viewpoint.com/>) - Vaca; 3D Cafe (<http://www.3dcafe.com/>) - Dragão.

REFERÊNCIAS

- [Baumgart72] B. G. Baumgart. Winged-edge polyhedron representation. Technical report, STAN-CS-320, Stanford University, 1972.
- [Brisson93] E. Brisson. Representing geometric structures in d dimension: topology and order. *Discrete & Computational Geometry*, Vol.9, no.4, pp.387-426, 1993.
- [Campagna98] S. Campagna, L. Kobbelt, and Hans-Peter Seidel. Directed Edges - A Scalable Representation for

- Triangular Meshes. *Journal of Graphics Tools: JGT*, Vol. 3, no.4, pp.1-12, 1998.
- [Carey97] R. Carey, G. Bell, and C. Marrin. The Virtual Reality Modeling Language ISO/IEC 14772-1, 1997. <http://www.vrml.org/technicalinfo/specifications/vrml97/index.htm>.
- [Garland99] M. Garland. Multiresolution modeling: Survey & future opportunities. In *Eurographics '99 - State of the Art Reports*, pp.111-131, 1999.
- [Heckbert97] P. S. Heckbert and M. Garland. Survey of Polygonal Surface Simplification Algorithms. *Siggraph'97 Course Notes* 25, 1997.
- [Hoppe96] H. Hoppe. Progressive Meshes. In *Proceedings of Siggraph'96*, pp.99-108, 1996.
- [Hoppe98] H. Hoppe. View-Dependent Refinement of Progressive Meshes. *Technical Report MSR-TR-98-02*, Microsoft Research, 1998.
- [Kallmann01] M. Kallmann and D. Thalmann. Star-vertices: A Compact Representation for Planar Meshes with Adjacency Information. *JGT*, Vol.6, no.1, 2001.
- [Lee99] A. Lee, D. Dobkin, W. Sweldens and P. Schroder. Multiresolution Mesh Morphing. In *Proceedings of Siggraph'99*, pp.343-350, 1999.
- [Lee01] S. H. Lee and K. Lee. Partial Entity Structure: A Fast and Compact Non-Manifold Boundary Representation Based on Partial Topological Entities. In *Proceedings of the Sixth ACM Symposium on Solid Modeling and Applications*, pp.159-170, 2001.
- [Loop00] C. Loop. Managing Adjacency in Triangular Meshes. *Tech. Report No. MSR-TR-2000-24*, Microsoft Research, January 2000.
- [Mantyla88] M. Mäntylä. *An Introduction to Solid Modeling*, Computer Science Press, 1988.
- [Ni94] X. Ni, and M. Susan Bloor. Performance Evaluation of Boundary Data Structures. *IEEE Computer Graphics and Applications*, Vol.14, no.6, pp.66-77, 1994.
- [Pajarola01] R. Pajarola. FastMesh: Efficient View-dependent Meshing. In *Proceedings of the Pacific Graphics 2001*, pp.22-30, 2001.
- [Popovic97] J. Popovic, H. Hoppe. Progressive Simplicial Complexes. *Computer Graphics*, Vol.31, pp.217-224, 1997.
- [Taubin98] G. Taubin and J. Rossignac. Geometric compression through topological surgery. *ACM Transactions on Graphics*, Vol.17, no.2, pp.84-115, 1998.
- [Wavefront93] Wavefront Technologies, Inc., Wavefront File Formats, Version 4.0 RG-10-004, first ed. Santa Barbara. CA. 1993.
- [Weiler88] K. Weiler. The radial edge structure: a topological representation for non-manifold geometric boundary modeling. In *Geometric modelling for CAD applications*. M. Wozny, H. McLaughlin, and J. Encarnacao, (eds.), North-Holland, pp. 3-36, 1988.
- [Weiler85] K. Weiler. Edge-Based data structure for solid geometric in Curved-Surface Environments. *IEEE CG&A*, Vol.5, no.1, pp.21-40, 1985.
- [Zorin97] D. Zorin, P. Schröder and W. Sweldens. Interactive Multiresolution Mesh Editing. In *Proceedings of the Siggraph'97*, pp.259-268, 1997.

Representação BSP de Curvas Implícitas 2D

Francisco Morgado
Dept. Informática, ESTV-IPV
3500 Viseu
fmorgado@di.estv.ipv.pt

Abel Gomes
Dept. Informática, UBI
6200-001 Covilhã
agomes@di.ubi.pt

Sumário

Os sistemas gráficos actuais incluem primitivas para desenhar segmentos de recta, circunferências, curvas e superfícies de Bézier, NURBS (Non-Uniform Rational B-Splines). No entanto, estes sistemas não fornecem primitivas gerais para representar graficamente curvas implícitas. A razão fundamental para este estado-de-coisas prende-se com o facto destas curvas poderem apresentar singularidades (e.g. auto-intersecções). Este artigo introduz um algoritmo genérico, eficiente e robusto que permite representar qualquer curva analítica definida implicitamente. Para isso, é utilizado um algoritmo BSP (Binary Space Partition) que particiona recursivamente o espaço ambiente $\Omega \subseteq \mathbb{R}^2$ numa forma não-uniforme de modo a determinar um conjunto de pontos que constituem uma aproximação discreta da curva. Cada ponto não é mais do que a intersecção da recta de bissecção dum subespaço com a curva, sendo determinado através dum algoritmo de aproximação numérica. Não são utilizadas quaisquer técnicas de diferenciação. Ao contrário doutros algoritmos de decomposição, este algoritmo permite também determinar pontos isolados, utilizando o conceito de extremo local de uma função num dado intervalo.

Palavras-chave

Curvas implícitas, BSP, aproximação numérica.

1. INTRODUÇÃO

Neste artigo, considera-se o problema da representação de curvas analíticas definidas implicitamente em $\mathbb{R}^2$. Por outras palavras, pretende-se calcular uma aproximação poligonal da curva $C = \{(x,y) \in \Omega \subseteq \mathbb{R}^2 : f(x,y)=0\}$ definida pelo conjunto de zeros duma função analítica f de $\mathbb{R}^2$ em $\mathbb{R}$. Refira-se que o algoritmo proposto neste artigo não se aplica somente a curvas algébricas como em [Blinn82], mas também às curvas analíticas.

Basicamente, existem três categorias de algoritmos para representar curvas implícitas:

- *Conversão da representação.* Estes algoritmos convertem uma curva implícita numa curva paramétrica [Allgower91], [Bloomenthal88], [Lorenson87]. Contudo, raramente existe uma parametrização *global* para uma curva implícita. O que é garantido é que existe uma parametrização *local* na vizinhança de qualquer ponto regular da curva, i.e. $f(x,y)=0$ e $\nabla f(x,y) \neq 0$. Mas, existindo uma parametrização global da curva, utiliza-se a sua representação paramétrica para desenhar a curva.
- *Perseguição ou continuação da curva.* A ideia geral deste tipo de algoritmos é calcular um ponto da curva à custa do ponto anteriormente calculado [Chandler88], [Moller95]. Para isso, efectua-se o cálculo do vector Hamiltoniano

$\left(-\frac{\partial f}{\partial y}, \frac{\partial f}{\partial x}\right)$ combinado com o método denomi-

nado por Newton Corrector [Allgower90]. Estes métodos são atractivos, pois concentram o esforço de processamento onde é necessário, adaptando-se também à variação da forma local da curva. No entanto, há sempre o risco de nem todas as componentes da curva serem desenhadas. Para que todas as componentes da curva sejam desenhadas temos de calcular um ponto inicial em cada uma delas, o que nem sempre é possível.

- *Subdivisão do espaço.* Estes algoritmos fazem uma subdivisão do espaço em subespaços de menor área, eliminando imediatamente os subespaços onde não existe qualquer segmento da curva [Bloomenthal88], [Bloomenthal94], [Triquet01]. Contudo, as técnicas de subdivisão não têm sido muito atractivas, pois o processamento intensivo dos algoritmos de subdivisão tornam lentas muitas aplicações [Snyder92]. Recentemente [Lopes01] apresentou uma técnica usando aritmética intervalar que permite aumentar o desempenho de representação da curva.

Este artigo apresenta uma técnica de subdivisão para curvas implícitas no plano. O algoritmo adapta-se à curvatura da curva, efectuando mais subdivisões onde a curvatura muda notoriamente. O algoritmo permite também a

determinação de pontos isolados da curva. Um ponto isolado é determinado através de um processo de aproximação (apoiado no cálculo dos extremos relativos de f) que converge para o tal ponto isolado.

O artigo está organizado em secções. Na secção 2 descreve-se o método de partição BSP não-uniforme, a estrutura BSP usada, assim como o modo como são determinados os pontos isolados. Na secção 3 é descrita a continuação dos pontos na estrutura BSP, de modo a desenhar correctamente a curva. Na secção 4 apresentam-se alguns resultados experimentais relevantes. O artigo termina com as conclusões na secção 5.

2. PARTIÇÃO BINÁRIA NÃO-UNIFORME

2.1 Estrutura de dados BSP

A Figura 1 ilustra o método de partição binária não-uniforme do espaço ambiente $\Omega \subseteq \mathbb{R}^2$ onde se encontra parte ou a totalidade da curva implícita C . Este método divide recursivamente o espaço Ω em dois novos subespaços Ω_{left} e Ω_{right} através de uma recta l que intersecta a curva. A recta l é denominada por recta de bissecção (ou recta BSP). Se, porventura, um subespaço contém um segmento da curva, então será novamente subdividido em dois subespaços, a não ser que a distância entre os pontos finais do segmento da curva seja menor ou igual a uma pequena tolerância ξ , ou que a curvatura ao longo do segmento da curva não varie numa forma significativa. Os subespaços onde não existe qualquer segmento da curva são desde logo abandonados.

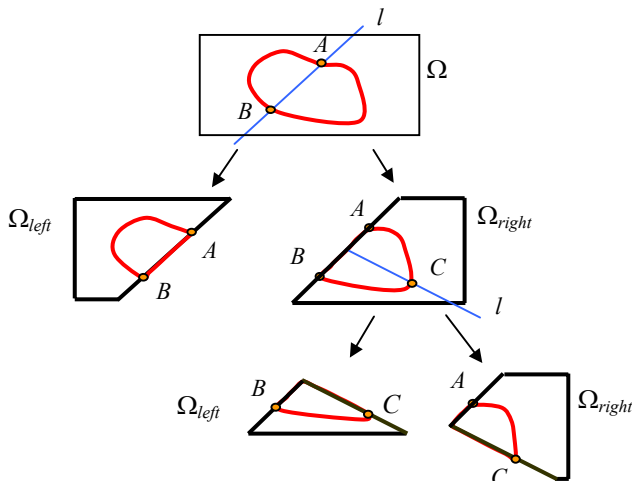


Figura 1: Árvore BSP

A correspondente estrutura de dados é uma árvore binária, designada por árvore BSP. Em termos de código C++, cada nó da árvore BSP é codificada do seguinte modo:

```
class BSP_Node {
    List *fr;
    List *lip;
    BSpline *l;
    BSP_Node *left, *right;
    BSP_Node *next;
}
```

A variável *fr* identifica a fronteira $\text{Fr}(\Omega)$ de um subespaço convexo Ω . A fronteira dum subespaço consiste numa lista de segmentos de recta que o delimitam; *lip* é a lista de pontos resultante da intersecção $\text{Fr}(\Omega) \cap C$; *l* é a recta que subdivide Ω em dois novos subespaços, Ω_{left} e Ω_{right} ; *left* identifica Ω_{left} enquanto que *right* identifica Ω_{right} ; *next* é um ponteiro que permite a travessia directa das folhas da árvore, o que permite visualizar C mais rapidamente.

2.2 Recta BSP

A determinação da recta de bissecção (ou recta BSP) l envolve os seguintes cálculos no subespaço a subdividir:

- Determinam-se os pontos de intersecção $\text{Fr}(\Omega) \cap C$. Caso a intersecção seja nula, são geradas várias rectas até que uma delas intersecte a curva;
- Escolhem-se os pontos mais afastados P e Q do passo anterior;
- Determina-se a mediatriz l do segmento $\overline{PQ}$.

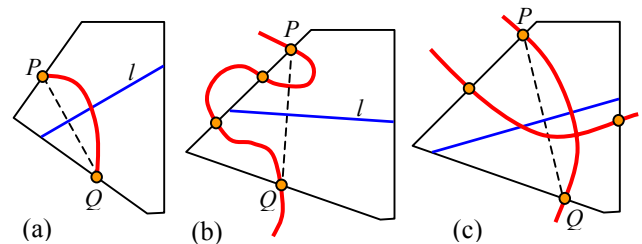


Figura 2: Determinação da recta BSP

A recta BSP é precisamente a mediatriz do segmento de recta $\overline{PQ}$ (Figura 2). A ideia subjacente à escolha dos pontos mais afastados, P e Q , é permitir subdividir o espaço o mais equitativamente possível. Deste modo, a árvore BSP resultante terá tendencialmente menos nós. De facto, através deste critério de subdivisão há a tendência para obter mais rapidamente segmentos da curva quase lineares com um menor número de subdivisões.

O correspondente algoritmo para determinar a recta BSP, poderá ser escrito do seguinte modo:

ALGORITMO 1 (BSPLINE)

INPUT:

(a) Ω : subespaço de $\mathbb{R}^2$

OUTPUT:

(a) l : recta BSP

Begin

1. $lip \leftarrow \text{Fr}(\Omega) \cap C$

2. **if** (lip)

– $l \leftarrow$ recta arbitrária que passa pelo centro de Ω e intersecta C

else

– determinar os pontos P e Q mais afastados de lip

– $l \leftarrow$ mediatriz de $\overline{PQ}$

– **if** ($d(P, Q) < \xi$) **return** NULL

– **if** ($d(P, Q) < \tau$ e $\#(\text{Fr}(\Omega) \cap C) == 2$)

• $R \leftarrow l \cap C$

• **if** ($\angle(\overline{RP}, \overline{RQ}) \approx 180^\circ$) **return** NULL

3. **return** l

End

O algoritmo de partição binária não depende da existência de singularidades na curva (e.g. auto-intersecções). De facto, se um subespaço contiver uma auto-intersecção, a sua partição recursiva tende a convergir para tal singularidade. Terminada a partição recursiva do espaço inicial, as singularidades estarão certamente nos nós ou subespaços terminais da árvore BSP.

Um subespaço Ω contém uma auto-intersecção se o número de pontos da curva na sua fronteira for maior ou igual a 3, i.e. $\#(\text{Fr}(\Omega) \cap C) \geq 3$. Contudo, para os nós intermédios esta condição não é válida. Por exemplo, na Figura 2(b), a curva intersecta a fronteira de Ω em quatro pontos, mas não existe qualquer auto-intersecção da curva em Ω .

Note-se que os subespaços terminais são relativamente pequenos uma vez que satisfazem a condição $d(P, Q) < \xi$, i.e. a distância entre os dois pontos mais afastados $P, Q \in \text{Fr}(\Omega) \cap C$ é menor que ξ . Esta condição é o critério de paragem do algoritmo de partição binária recursiva. Nestas condições, assume-se que uma auto-intersecção dum subespaço terminal é o ponto médio de $\overline{PQ}$. Este critério funciona bastante bem mesmo em casos degenerados em que a distância entre P e Q é muito pequena, mas o comprimento do arco pode ser significativo como acontece com, por exemplo, a curva $y - \sin(1/x) = 0$.

O segundo critério de paragem tem a ver com o controlo da curvatura. Este critério é a conjunção das condições $d(P, Q) < \tau$ e $\#(\text{Fr}(\Omega) \cap C) == 2$ implementadas no algoritmo BSPLINE, onde $\tau = 3\xi$. Neste caso, se o ângulo $\angle(\overline{RP}, \overline{RQ}) \approx 180^\circ$, a partição do subespaço termina, i.e. nenhuma recta BSP é calculada, pois os pontos R, P e Q

são colineares a menos de uma pequena tolerância. (O ponto R é o ponto de intersecção entre a mediatriz l e a curva C). Caso contrário, a recta BSP, previamente calculada, é retornada e o processo de subdivisão continua.

2.3 Posicionamento dos pontos da curva em relação à recta BSP

Antes de subdividir o subespaço Ω em dois novos subespaços Ω_{left} e Ω_{right} através da recta l , terá de ser feita a classificação dos pontos da curva que se encontram na fronteira de Ω de modo a decidir quais destes pontos pertencem a Ω_{left} e a Ω_{right} .

Seja P um ponto da recta l (Figura 3). O algoritmo de classificação poderá ser escrito do seguinte modo:

- Determina-se o vector unitário $\vec{u}$ em P com a direcção de l ;
- Determina-se o vector unitário $\vec{w}$ perpendicular a Ω (em $\mathbb{R}^3$);
- Determina-se o vector unitário $\overrightarrow{PX}$ para todo o ponto X que intersecta a fronteira de Ω ;
- Se o produto misto $(\overrightarrow{PX} \times \vec{u}) \cdot \vec{w} > 0$, o ponto X pertence ao subespaço Ω_{left} ;
- Se o produto misto $(\overrightarrow{PX} \times \vec{u}) \cdot \vec{w} < 0$, o ponto X pertence ao subespaço Ω_{right} .

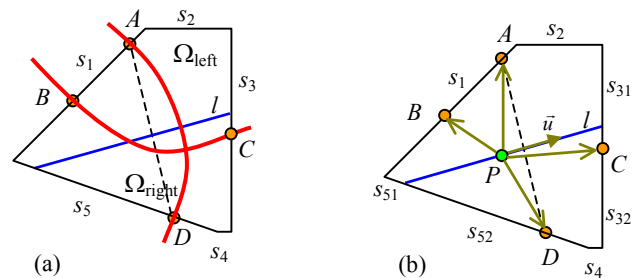


Figura 3: Classificação dos pontos da curva em relação à recta BSP

Na Figura 3, os pontos A e B pertencem ao subespaço Ω_{left} , enquanto que C e D pertencem a Ω_{right} . Se o produto misto $(\overrightarrow{PX} \times \vec{u}) \cdot \vec{w} = 0$, o ponto X pertence ao subespaço Ω_{left} e também ao subespaço Ω_{right} . Isto acontece quando X (e.g. P na Figura 3(b)) é um ponto de intersecção entre a curva e a recta BSP.

2.4 Criação e actualização dos subespaço Ω_{left} e Ω_{right}

Após a classificação dos pontos de $\text{Fr}(\Omega) \cap C$ nos novos subespaços Ω_{left} ou Ω_{right} , temos de redefinir as fronteiras destes novos subespaços. Isto é ilustrado na Figura 3, onde $\text{Fr}(\Omega) = \{s_1, s_2, s_3, s_4, s_5\}$ é a fronteira do subespaço convexo Ω em $\mathbb{R}^2$ e l é a recta que intersecta $\text{Fr}(\Omega)$ em

dois pontos. A primeira intersecção subdivide s_3 em dois novos segmentos, s_{31} e s_{32} , enquanto que o segundo subdivide s_5 em s_{51} e s_{52} . Estas duas intersecções originam o segmento $s_l \subset l$ que subdivide Ω em $\Omega_{\text{right}} = \{s_l, s_{32}, s_4, s_{52}\}$ e $\Omega_{\text{left}} = \{s_l, s_{51}, s_1, s_2, s_{31}\}$.

A criação destes dois novos subespaços requer também a classificação dos seus segmentos fronteiros em relação à recta l . A classificação dos segmentos é baseada no processo de classificação dos pontos descritos na subsecção anterior. Se o produto misto $(\overrightarrow{PX} \times \vec{u}) \cdot \vec{w}$ é maior ou igual a 0 para os dois pontos que definem um dado segmento, então este segmento fará parte da fronteira do subespaço Ω_{left} , mas se for menor que 0, então o segmento pertence à fronteira de Ω_{right} .

O correspondente algoritmo de criação e actualização dos subespaços é então:

ALGORITMO 2 (SUBSPACES)

INPUT:

- (a) l : recta de partição de Ω
- (b) Ω : subespaço de $\mathbf{R}^2$

OUTPUT:

- (a) Ω_{left} : $\Omega_{\text{left}} \subset \Omega$
- (b) Ω_{right} : $\Omega_{\text{right}} \subset \Omega$

Begin

- $\Omega_{\text{left}} \leftarrow \emptyset$
- $\Omega_{\text{right}} \leftarrow \emptyset$
- Determinar os dois pontos de intersecção $\text{Fr}(\Omega) \cap l$
- Classificar os pontos $\text{Fr}(\Omega) \cap C$ e $\text{Fr}(\Omega) \cap l$
- Actualizar fronteira de Ω_{left}
- Actualizar fronteira de Ω_{right}

End

2.5 Intersecção entre a curva e recta BSP

O método da secante é um método de procura de zeros de funções [Akai94]. Basicamente, este método parte de duas estimativas distintas A e B para a solução de $f(x)=0$ (veja-se Figura 4).

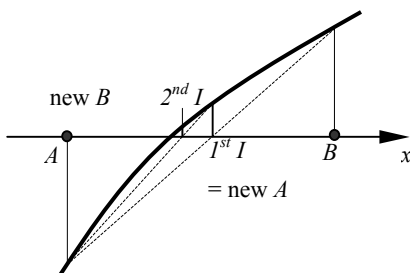


Figura 4: Representação do método da secante.

O processo é iterativo e envolve interpolação linear, sendo actualizados A ou B em cada iteração. A interpolação é dada pela fórmula

$$I = B - f(B) \frac{B - A}{f(B) - f(A)}$$

onde I é a próxima estimativa que aproxima o ponto de intersecção.

O método da secante faz parte do algoritmo para a determinação da intersecção entre a curva C e a mediatriz l do subespaço. Inicialmente, a mediatriz é subdividida em vários segmentos mais pequenos. Depois, a cada segmento de recta é aplicado o método da secante se for satisfeita a condição $f(A)f(B) < 0$. Esta condição garante que neste segmento existe ponto de intersecção $C \cap l$. A divisão da mediatriz em pequenos segmentos é necessária, pois pode existir mais do que um ponto de intersecção entre a mediatriz e a curva C .

Seja MAXLENGTH o comprimento máximo admissível para a mediatriz, i.e. o comprimento da diagonal do espaço inicial Ω , e MAX=10 o número máximo de subdivisões da mediatriz inicial. O número de subdivisões da mediatriz é adaptativa e depende do comprimento da mesma. Seja LENGTH o comprimento da mediatriz. Se $LENGTH \leq (MAXLENGTH / MAX)$, então aplica-se o método da secante à própria mediatriz (pois esta já é relativamente pequena); caso contrário, divide-se a mediatriz em NSEG segmentos mais pequenos, onde NSEG é dado por

$$NSEG = \frac{MAX \times LENGTH}{MAXLENGTH}$$

sendo depois aplicado o método da secante a cada segmento.

2.6 Determinação de pontos isolados

Por definição, um *ponto isolado* numa curva é todo o ponto P para o qual não existe variação de sinal de $f(x)$ numa pequena vizinhança centrada em P .

A determinação de pontos isolados é feita nos subespaços terminais da árvore BSP. A ideia base consiste em determinar um ponto da fronteira de Ω onde existe variação da monotonia de f . Note-se que existe variação de monotonia em extremos locais de f . Normalmente, esta variação de monotonia é calculada através do cálculo da derivada. No entanto, dada a restrição auto-imposta de não usar técnicas de derivação, a variação de monotonia é aproximada pela variação de sinal da taxa de variação média (TVM) entre dois pontos, A e B :

$$TVM_{[A,B]} = \frac{f(B) - f(A)}{B - A}.$$

Note-se que o processo de determinar um ponto isolado não poderá ser nunca um método baseado no método da

secante, pois este pressupõe que existe sempre variação de sinal de f no intervalo que estamos a considerar. Ora, por definição de ponto isolado, essa variação de sinal de f não existe numa vizinhança do ponto isolado.

Portanto, o algoritmo de determinação dum ponto isolado começa por determinar o ponto extremo (em valor absoluto) de f num dos segmentos fronteiros dum subespaço terminal. Depois, calcula-se o segmento de recta perpendicular ao segmento anterior no ponto extremo. Repete-se este processo de calcular novo ponto extremo e novo segmento perpendicular ao segmento anterior até atingir o ponto mínimo em valor absoluto (ou seja, o ponto isolado) que é um zero da função.

Este processo de determinação do ponto isolado consiste basicamente em determinar a sequência de pontos $P_0, P_1, \dots, P_n$, sendo que os vários P_i calculados são os pontos mínimos relativos de f nos vários segmentos gerados. Este processo poderá ser interpretado como o percurso dos pontos mínimos relativos até chegar ao ponto isolado, se ele existir (Figura 5).

O algoritmo para determinar um ponto isolado será então:

ALGORITMO 3 (ISOLATEDPOINT)

INPUT:

- (a) Ω : subespaço terminal de $\mathbf{R}^2$
- (b) S : segmento fronteiro de Ω

OUTPUT:

- (a) P_i : ponto isolado

Begin

- $P_e \leftarrow \text{PONTOEXTREMO}(S)$
- $P_i \leftarrow \text{NOVOPONTOEXTREMO}(P_e, S, \Omega, 5)$
- **if** (P_i) **return** P_i
- **return** $NULL$

End

O algoritmo para determinar o ponto candidato é o seguinte:

ALGORITMO 4 (PONTOEXTREMO)

INPUT:

- (a) S : segmento de recta pertencente a $\text{Fr}(\Omega)$

OUTPUT:

- (b) P_e : ponto extremo

Begin

Percorrer o segmento de recta S , e determinar o ponto onde existe variação da $\text{TVM}_{[A,B]}$, onde $A, B \in S$ e não existe variação de sinal de f .

End

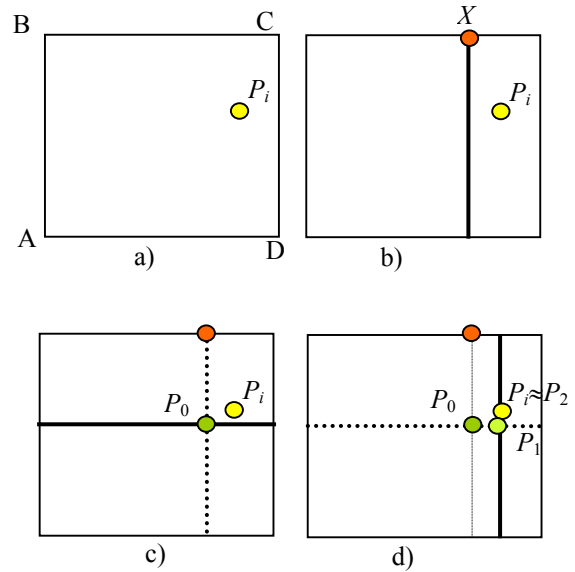


Figura 5: Determinar ponto isolado.

ALGORITMO 5 (NOVOPONTOEXTREMO)

INPUT:

- (a) P_e : ponto extremo
- (b) S : segmento fronteiro de Ω ao qual pertence P_e
- (c) Ω : subespaço terminal de $\mathbf{R}^2$
- (d) N : nível de recursividade

OUTPUT:

- (a) P_i : ponto isolado

Begin

- **if** ($(N=0) \vee (P_e = NULL)$) **return** $NULL$
- **if** ($|f(P_e)| \leq TOL$) **return** P_e
- $S \leftarrow$ segmento contendo P_e e $\perp S$ em Ω
- $P_e \leftarrow \text{PONTOEXTREMO}(S)$
- **return** $\text{NOVOPONTOEXTREMO}(P_e, S, \Omega, N-1)$

End

3. CONTINUAÇÃO E VISUALIZAÇÃO DA CURVA

Para representar a curva, é necessário ter um processo que permita atravessar a árvore BSP e estabelecer uma sequência correcta dos seus pontos. Pela Figura 1, torna-se claro que só os nós terminais têm os pontos que interessam representar. Os nós intermédios servem somente para controlar o processo de subdivisão do espaço. Deste modo, para efeitos de visualização, interessa somente percorrer os nós terminais numa forma sequencial, de modo que o processo de visualização da curva seja equivalente a percorrer uma lista ligada.

Desta forma, há um ganho que é tendencialmente 50% na representação da curva C em relação à travessia da árvore BSP na sua totalidade, visto que uma árvore BSP de altu-

ra N tem $2^{N+1}-1$ nós, ao passo que a lista dos nós terminais tem somente 2^N nós (cerca de 50% dos nós da BSP).

Deste modo, o algoritmo geral para representar graficamente a curva implícita será o seguinte:

ALGORITMO 6 (CURVE)

INPUT:

(a) C : a curva implícita

Begin

1. $\Omega \leftarrow$ região rectangular de $\mathbf{R}^2$
2. $BSP(\Omega, 7)$
3. Visualizar C

End

O sub-algoritmo BSP (passo 2) é a parte principal de todo o algoritmo, sendo descrito do seguinte modo:

ALGORITMO 7 (BSP)

INPUT:

(a) Ω : subespaço de $\mathbf{R}^2$

(b) d : nível de recursividade

Begin

1. **if** ($d=0$) **return**
2. $l \leftarrow BSPLINE(\Omega)$
3. **if** (l)
 - $SUBSPACES(l, \Omega, \Omega_{left}, \Omega_{right})$
 - $BSP(\Omega_{left}, d - 1)$
 - $BSP(\Omega_{right}, d - 1)$
- else** // estamos perante um subespaço terminal
 - $ISOLATEDPOINT(\Omega)$

End

Note-se que o terceiro critério de paragem é dado pelo nível de recursividade (ou grau da árvore BSP). Para os testes realizados, assumiu-se que o nível máximo de recursividade era 7.

4. RESULTADOS EXPERIMENTAIS

O algoritmo apresentado mostrou-se relativamente rápido, mesmo tendo em conta que é um algoritmo de BSP. Curiosamente, o algoritmo é mais rápido em segmentos da curva com maior variação de curvatura. Isto resulta do facto de os subespaços terminais se tornarem rapidamente mais pequenos em curvas com grande oscilação da curvatura, tornando a procura das soluções mais rápida.

As curvas apresentadas na Figura 6 foram desenhadas com a precisão de $\xi = 10^{-3}$ e com o nível máximo de recursividade na árvore BSP igual a 7. A figura de mérito $fcpp$ (acrónimo de *function evaluations per point*) refere-

se ao número de vezes que uma função é calculada para que seja determinado com precisão um ponto da curva.

Na Figura 6 mostram-se várias curvas e respectivas subdivisões espaciais. Na Figura 6(a) está representada uma aproximação à curva $(x-1)(y-1)=0$ através dum conjunto finito de pontos. Esta curva pode ser vista como a união das rectas perpendiculares $x-1=0$ e $y-1=0$ em $\mathbf{R}^2$.

Na Figura 6(b) e (c) estão representadas duas curvas, cada uma das quais tem um ponto isolado. A curva da Figura 6(d) tem três componentes, ao passo que as restantes curvas da Figura 6 têm duas componentes, mas nenhuma delas tem pontos isolados.

Note-se que o tempo para desenhar cada curva (e respectiva subdivisão do espaço ambiente) é uma fracção pequena de um segundo, o que é bastante razoável para algoritmos de subdivisão espacial. Isto deve-se em parte, por comparação com outros algoritmos de subdivisão, ao menor número de vezes que a função é calculada para cada ponto que é determinado, i.e. o valor de $fcpp$ é menor. Este valor permite, de algum modo, avaliar a eficiência do algoritmo; por exemplo, em [Lopes01], esse valor é cerca de 23. Os resultados dos testes apresentados na Figura 6 foram realizados num PC com processador Intel Pentium 500MHz, 128MB RAM e com o sistema operativo Windows NT.

Os sistemas de software comerciais como, por exemplo, o Maple e o Mathematica não incorporam algoritmos precisos e correctos para representar curvas implícitas. Em geral, são capazes de representar parcialmente curvas implícitas, mas falham claramente na representação das suas singularidades (e.g. auto-intersecções). Por exemplo, o ponto $P=(1,1)$ de auto-intersecção da curva $(x-1)(y-1)=0$, representada na Figura 6(a), não é representável naqueles sistemas comerciais. Os pontos isolados das curvas representadas em (b) e (c) da Figura 6 também não são detectáveis pelos referidos sistemas.

Note-se que, em geral, os algoritmos de subdivisão do espaço permitem determinar uma boa aproximação duma curva implícita [Lopes01]. Infelizmente, estes algoritmos não conseguem distinguir um ponto regular de um ponto singular (ou singularidade). Além disso, só por mera sorte estes algoritmos conseguem determinar a existência dum ponto isolado.

5. CONCLUSÕES

O algoritmo proposto resultou do desenvolvimento dum outro algoritmo proposto pelos autores (veja-se [Morgado02]). No entanto, ao contrário do anterior algoritmo, o algoritmo aqui e agora proposto baseia-se unicamente no conceito de subdivisão binária do espaço. O algoritmo descrito em [Morgado02] é a bem dizer um algoritmo híbrido, pois faz uso não só do conceito de subdivisão do espaço, mas também do conceito de vizinhança usado nos algoritmos de perseguição ou continuação da curva.

Além disso, o algoritmo permite também determinar as várias componentes da curva, pontos isolados inclusiv-

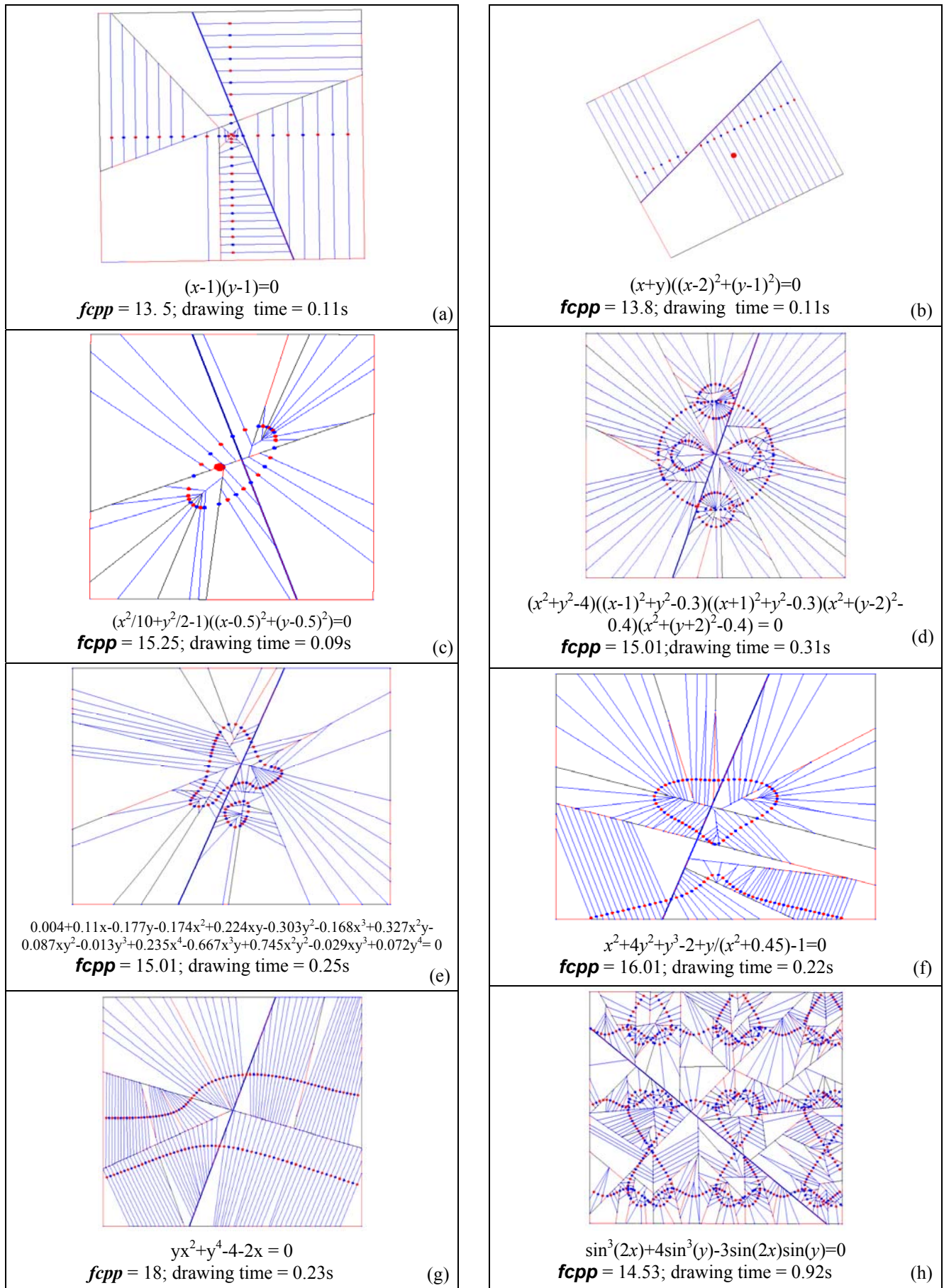


Figura 6: Resultados experimentais.

Note-se que, talvez, a maior contribuição deste artigo é precisamente resolver o problema da determinação dos pontos isolados.

O algoritmo mostrou-se relativamente rápido e pensamos que poderia ser incluído nos sistemas gráficos actuais. No entanto, a rapidez do método depende em grande parte da eficiência do método de cálculo da intersecção entre um segmento de recta e a curva. Por isso torna-se importante para o futuro encontrar algoritmos que permitam acelerar esse processo.

6. REFERENCES

- [Allgower90] Allgower, E. e Georg, K. Numerical Continuation Methods: An Introduction. Springer-Verlag, (1990)
- [Allgower91] Allgower, E., Gnutzmann, S.: Simplicial Pivoting for Mesh Generation of Implicitly Defined Surfaces. Computer Aided Geometric Design, 8 (1991), 305--325
- [Akai94] Akai, Terrence J.: Applied Numerical Methods for Engineers. John Wiley & Sons Inc, (1994)
- [Blinn82] Blinn J. F., A Generalization of Algebraic Surface Drawing, ACM Transactions on Graphics, 1(3), 1982, 235-256.
- [Bloomenthal88] Bloomenthal, J. Polygonisation of implicit surfaces Computer Aided Geometric Design, 5, (1988), 341-355.
- [Bloomenthal94] Bloomenthal, J. An Implicit Surface polygonizer. Graphics Gems, IV, (1994).
- [Chandler88] Chandler, R. A tracking algorithm for implicitly defined curves, IEEE Computer Graphics & Applications, 8(2), (1988), 83-89.
- [Lopes01] Lopes, H., Oliveira, J. e Figueiredo, L., Robust Adaptive Polygonal Approximation of Implicit Curves, Actas de SIBGRAP'2001, IEEE Computer Society.
- [Lorensen87] Lorensen, W. e Cline, W. Marching Cubes: A High Resolution 3D Surface Construction Algorithm, Computer Graphics, 21(4), 1987, 163-169.
- [Moller95] Moller, T. and Yagel, R., Efficient Rasterization of Implicit Functions, (1995), artigo publicado em (<http://citeseer.nj.nec.com/357413.html>).
- [Morgado02] Morgado, F. e Gomes, A., Fast Representation of Implicit Curves Through Space Subdivision, Actas do 1º Simpósio Ibero-Americano de Computação Gráfica (SIACG'2002), Guimarães, Portugal, 2002.
- [Snyder92] Snyder, J.: Interval arithmetic for computer graphics. Proceedings of ACM Siggraph, (1992), 121—130.
- [Triquet01] Triquet, F. *et al.* "Fast Polygonization of Implicit Surfaces", WSCG' 2001,2,(2001), 283-290.

BlobMaker: Free form Modelling with Variational Implicit Surfaces

Bruno Rodrigues de Araújo
IST/ IMMI INESC-ID
Rua Alves Redol, 1000-029 Lisboa
brar@immi.inesc.pt

Joaquim Armando Pires Jorge
Dep. Eng^a. Informática, IST
Av. Rovisco Pais, 1000 Lisboa
jorgej@acm.org

ABSTRACT

We present BlobMaker, a program for modelling surfaces using variational implicit surfaces. Our approach uses variational implicit surfaces as a geometrical representation for free-form shapes. We have implemented new modelling operations to support stroke (pen-based) input. To this end, we have built a complete modeller application using variational implicit surfaces. Users can create and manipulate shapes using sketches on a perspective or parallel view. The main operations are inflate, which creates 3D forms from a 2D stroke, merge, which creates a 3D shape from two blobs and oversketch, which allows users to redefine shapes using a single stroke to change their boundaries or to modify a surface by an implicit extrusion. We compare these techniques with those of other approaches published. Finally, we describe their implementation in BlobMaker. We have provided additional features such as copying, picking and dragging to offer a natural user interface suitable free-form modelling.

Keywords

Stroke based modelling application, variational implicit surfaces

1. INTRODUCTION

Modelling applications have become essential tools in the animation and manufacturing industries and now play a crucial role in the design workflow. However, CAD interfaces have not evolved much past the WIMP (Windows, Icons, Mouse and Pointing) to match the spectacular increase in modelling power of these systems. Therefore, CAD systems have become very complex, needing high technical knowledge and long learning periods. These problems have made them difficult to use for traditional designers. While WIMP interfaces represent a considerable improvement over command line applications, menu-based interactions do not map naturally to pen-and-pencil drawing modes. Analysing the actual command usage in a representative commercial CAD application, we can verify that 75% of user time is spent in menu navigation, picking and selection instead of model input [Sanchis02]. While this approach was acceptable for input via mouse and keyboard the emergence of pen-based systems mandates adopting better input methods.

Although pen-based input devices are supported by many modern applications, they are not used as well as they could for 3D modelling in contrast to 2D sketching programs. In the later case paper and pencil metaphor can be used to leverage both ergonomic and human-learned skills offering advantages in drawing speed, and rich expression afforded by pen movements and pressure. Applications such as Corel Painter [Corel] constitute a good example of this. However, conventional 3D modelling applications restrict pen-based input to entering point data. Examples of this are 3D Studio

Max [3DMax], Lighthwave 3D [Lighthwave3D], Maya [Maya, SoftImage] and CAD systems such as Dassault Systems' Catia [CATIA]. Even though many of these applications allow some type of free-form modelling, they require detailed knowledge about the geometric representations such as NURBS [Farin99] and the intricacies of their mathematical formulations, which makes them difficult to use by traditional designers.

In the next section, we survey previous work in free-form modelling interfaces. Then we present a description of variational implicit surfaces. In sections four and five, we describe in detail our modelling operations based in sketch input. Then we present an implementation of our techniques as embodied in BlobMaker. Finally, we draw conclusions and highlight possible improvements for future work.

2. GESTURE-BASED MODELLING INTERFACES

Ivan Sutherland's SketchPad [Sutherland63] was the first graphical user interface to allow precise drawing using a calligraphic interface, modelling hierarchy and constraints. In the 1970's, several research works followed up on these ideas through sketch-based recognition and tablet systems [Negroponte73, Herot76].

In the early 1990's, the first generation of pen computer fostered the emergence of new interfaces based on sketch input. The SKETCH [Zelevnik96] and SKETCH-N-MAKE [Bloomenthal98] systems, combined gesture and geometric recognition to allow creating and modifying 3D models. SKETCH defines a specific gesture grammar language to allow the creation of simple 3D primitives in

an orthogonal view; for example, three concurrent lines define a cube. This gesture language allows the user to specify CSG operations and to define quasi-free form shapes (such as ducts) through extrusion.

Several works have introduced gestures for constructing complex 3D models based on line reconstruction algorithms. For example, GIDES [Pereira00] allows users to sketch on an orthogonal view, which combines with a suggestive interface to reduce the command set. The system provides a language similar to SKETCH to create complex models. It uses specific commands to allow constrained positioning between elements and construction lines to define specific locations, to constrain the output of the recognizer and allow rigorous drawing with imprecise sketches. CHATEAU [Igarashi01] provides another good example of a suggestive interface. Like GIDES and SKETCH COSMO [Michalik02] it uses a gesture language to specify extrusions. While these systems offer support for reliable extrusions, we need more elaborate functions to support “true” free-form modelling.

REFER[Contero01] and SKETCHUP [Sketchup3D] provide other examples of modelling using line drawings for architectural applications. Both systems feature recognition and reconstruction of models from straight lines. One serious problem with line drawings is the Necker ambiguity [Necker32] characteristic of 3D wire-frame models. The SKETCHUP system, offers new operations for interaction such as face and edge dragging that are much simpler than conventional CSG methods.

The different approaches presented here offer good solutions for the construction of complex models but are limited to extrusions or their CSG combinations, which offer a poor substitute for free-form shapes. Thus, they are unable to model “soft” forms such as a human head or biological shapes (animals) or surfaces on a car body.

3. SKETCH BASED FREE FORM MODELLING

Brian Wyvill [Wyvill98] introduced the BlobTree, which is an implicit surface model based on skeleton primitives to describe soft objects. One prototype system described in this work uses sketches for defining skeleton primitives. Skeletons define blobs using a specific language, for example, a line defines a cylinder and a point a sphere. The interesting point of BlobTree, is the combination of implicit surfaces with CSG operations, presenting a mixture between line-based interaction and a “real” free form approach. However, this method is still too similar in both its virtues and limitations to the gesture-based systems described above.

A more familiar approach for free form editing is to sketch in 2D the contours of 3D shapes. This is more natural than using extrusions. The first work using contours for free-form modelling was the Teddy system [Igarashi99]. Teddy presents a very simple interface that combines extrusions with contour-based shape creation. The system offers several operations to modify the start-

ing shape, which is normally a blob created by inflation of a 2D contour. This system had a great impact in the Computer Graphics community due to its simplicity. It provides simple primitives to extrude, bend, cut or smooth shapes. Geometric representations in Teddy use a triangulated mesh that can be modified through stroke input. Sketches are projected on the mesh according to the current view. However, the system is only able to construct one object and does not support hierarchy.

In recent years, several projects have followed the steps of Igarashi. One of the most interesting is Karpenko’s [Karpenko02], which was the first to adopt a mathematical implicit representation. Like Igarashi, Karpenko presents a simple interface for free form modelling. Notably, she models geometrical objects through variational implicit surfaces [Turk99] instead of polygonal meshes. We present these in detail in the next section. This application organizes the modelling scene in a tree hierarchy, allowing constrained move operations between tree nodes and creating distinct objects in the same scene. The main operations offered by the interface are merge, which allows combining two blobs and oversketch for redefining the boundary of a blob. Karpenko’s modeller presents simple navigation facilities to allow users to rotate and translate objects. Teddy did not support these features since the whole scene contains a single object. Another interesting aspect is the use of guidance strokes to help the merging process. However, guidance strokes are a symptom that the merging operation could be improved and further simplified towards a more natural interaction.

4. VARIATIONAL IMPLICIT SURFACES

Variational implicit surfaces (VIS) were introduced by Turk [Turk99, Turk01, Turk02]. VIS are smooth and respect a set of constraint points. VIS are always closed and can have arbitrary topology. They are implicitly defined by a mathematical function f , where the surface is the set of 3D points which verify $f(X)=0$. Their main difference to other implicit models such as meta-balls [Nishimura85] and BlobTree, lies in that the surface has to obey constraint points specified by the user. VIS are not algebraic surfaces and are based in a problem similar to the thin-plate interpolation. This approach interpolates a cloud of points. To find the implicit function that respects all the constraint points with a minimum of curvature, $f(x)$ is defined such as to minimize (1):

$$E = \int_{\Omega} f''_{xx}(X) + 2f''_{xy}(X) + f''_{yy}(X) dX \quad (1)$$

There are several approaches to solve this problem. Turk decomposes $f(x)$ into a linear combination of radial basis functions ϕ centred on the constraints, using $\phi(X) = |X|^3$. The interpolated function, which satisfies the condition presented in (1) and defines the variational implicit surface, is presented in (2):

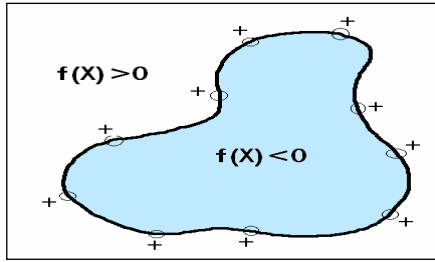


Figure 1: Example of boundary (o) and normal (+) constraints in variational implicit surfaces

$$f(X) = \sum_{j=1}^n d_j \phi(X - c_j) + P(X) \quad (2)$$

Where c_j are the locations of the constraints on the surface, d_j are the weight of each constraint, $P(X)$ is a one-degree polynomial which can be omitted if the number of constraints is greater than eight [TURK01].

For the calculation of the weights d_j , we know the value h_j of the height field for each constraint such as $f(c_j) = h_j$. Based on Equation (2), the following linear system is defined in (3):

$$M \cdot D = H \quad (3)$$

In (3), $D = [d_j]$ are the unknowns, $H = [h_j]$ are the height field values and M , a matrix defined as a function of ϕ , P and c_j . While the linear system can be solved using LU decomposition in $O(k^3)$ steps where k is the number of constraints, iterative methods can solve large-scale systems in $O(k^2)$.

To simplify creating VIS, Turk proposes a method based on four different types of constraints:

- Boundary constraints c_j that are placed in the boundary of the surface verifying $f(c_j) = h_j$ with $h_j = 0$
- Normal constraints c_j that are located outside the surface at a tiny distance of the boundary using the normal of the surface. These constraints verify $f(c_j) = h_j$ with $h_j = 1$
- Interior constraints c_j that are located arbitrary inside the surface verifying $f(c_j) = h_j$ with $h_j < 0$
- Exterior constraints c_j that are located arbitrary outside the surface verifying $f(c_j) = h_j$ with $h_j \geq 1$

It is possible to create variational implicit surfaces by specifying only boundary and normal constraints as shown in Figure 1. We can also use this principle for converting polygonal meshes into VIS [Yngve02].

The flexibility of this representation allows modelling complex and smooth models with arbitrary topology. We use it in our modeller, since it provides a compact and mathematically simple means of describing a surface using constraints, weights and a first order polynomial. It affords flexibility for the computation of normal and curvature information, while ensuring C^2 continuity.

5. FROM 2D TO 3D: INFLATION

In this section, we propose our method for the creation of 3D VIS based on user 2D stroke input, using an inflation process similar to [Karpenko02]. We present an overview of the algorithm, followed by the description of the relevant steps.

The inflation process takes a set of 2D points input by the user (stroke) and creates a 3D object matching the contour drawn by the user. The following list presents the different steps of the process:

- Filtering the 2D stroke : this step receives the input set of 2D points from the user and simplifies it
- Verification of filtered stroke : this step rejects incorrect parts of the stroke such as self-intersections
- Skeletonization of the 2D stroke: this step analyzes the entire stroke and creates a 2D skeleton with all the relevant topological information. The skeleton allows the reconstruction of the input stroke and the information about the thinness of the enclosed region in order to perform depth inflation
- Mapping the 2D stroke into a 3D contour: this step transforms the 2D stroke and skeleton information to a 3D virtual plane according to the actual definition of viewport and view parameters, computing both normal and depth information
- Creation of 3D implicit surface and its visualization: this step creates a VIS, representing the blob, which matches the contour's skeleton. The polygonization process creates a triangulated mesh together with surface normals at each vertex, which is suitable for rendering

5.1 Main contributions of our work

In the previous section, we have presented two different methods from Igarashi and Karpenko to create 3D surfaces from 2D strokes. Both approaches have advantages and limitations. In Teddy, the inflation process is oriented to the triangulated mesh. This approach restricts the possible operations on objects to local mesh modifications. This makes it difficult to merge two meshes. Furthermore, triangular meshes make it difficult to model arbitrarily smooth shapes. To overcome this limitation, Igarashi introduced a smoothing operator that applies local subdivision algorithms to the mesh. Even though we use similar methods for skeletonization, Teddy takes a different approach to inflation. While both Igarashi and Karpenko follow the same approach for filtering and mapping the 2D stroke, which is dependent on screen resolution. They reject consecutive 2D points closer than a specified number (15) of pixels. This can result in important details being dropped from the resulting surface.

While our approach to inflation is similar to Karpenko's her method might present incorrect results due to her screen-dependent filtering method and incorrect technique for depth classification. Her approach only defines two constraints on the VIS to define shape thickness. This brings about several limitations. One of them is that the merging operator needs a re-sampling of the triangulation.

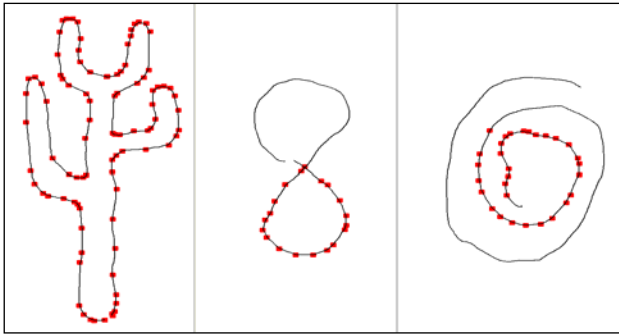


Figure 2: Example of filtering step, centre and right figures show self-intersecting strokes.

lated mesh. This increases the number of constraints on the VIS dramatically, thus incurring in additional computational costs for matrix solving. Another consequence is that the merge between a big blob and a very small one will be impossible without loss of detail. In our approach, all modelling operators use only the mathematic representation and skeleton information. The next sections describe in more detail our inflation process.

5.2 Filtering the input stroke

Since the sampling of stroke points from an electronic digitizer depends of the interrupt processing capability of the operating system, points on an input stroke are not evenly spaced. For example, some points can be repeated or their number can be unnecessarily large depending on the speed at which the stroke was drawn.

As discussed before, Igarashi and Karpenko folloz *pdf w an approach for stroke filtering which is dependent on screen resolution. In our approach, we implement a greedy algorithm similar to Douglas-Peucker filtering [Douglas73].

Figure 2 presents an input stroke and the result (red points) of the filtering; the input stroke on the left is compound by 2081 points and reduces to 92 points after the filtering step.

Finally, we analyze the filtered stroke, rejecting self-intersection parts. This step is necessary because the boundary of a shape cannot become self-intersecting when we project the stroke on a 3D virtual plane. In our approach, we retain the first non-intersecting loop found in the filtered stroke and drop the rest of the stroke as shown in Figure 2.

5.3 Skeletonization of 2D Strokes

The skeletonization step transforms the filtered stroke to reveal all the information of the stroke. We adopt the Chordal Axis Transform presented in [Prasad97] to accomplish this. This step applies a Constrained Delaunay Triangulation, using as constraints the edges of the filtered stroke. We generate three kinds of connectivity information for the triangulated polygon as illustrated in Figure 3, where triangles fall into one of three categories:

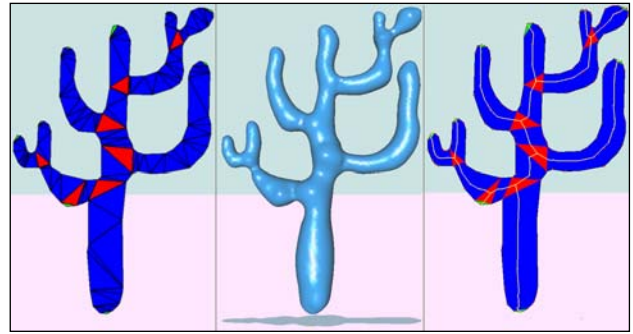


Figure 3: Chordal Axis Transform for skeletonization based on Constrained Delaunay Triangulation

- Terminal Triangle: triangles with one neighbour adjacent in the polygon (small outside triangles shown in green)
- Seed Triangles: two neighbours are adjacent to these triangles in the polygon (depicted in blue)
- Joint Triangles: triangles with three neighbours in the polygon (painted in red)

Then we transform the triangulated polygon into a skeleton where terminal triangles terminate limbs and joint triangles are connections in the hierarchy. Figure 4 presents an example of skeleton of a stroke that is similar to a hand. We can verify that the skeleton respects the shape; the red edges result from joint triangles. For each node in the skeleton, we save the distance between it and the border as depth information. The skeletonization procedure may generate some irrelevant branches (these appear in yellow in the figure). Since these are meaningless for the topology of the shape, they are rejected by a post-processing step of the skeletonization process.

Our process is similar to that followed by Igarashi except for the last step, where he derives a 3D mesh from the 2D triangulation, while we create a VIS from the skeleton information.

5.4 Projecting 2D points onto 3D virtual plane

At the end of the 2D analysis, the inflation proceeds with a mapping step. The goal of which is to map all 2D points into 3D scene coordinates. Since 3D Blobs are conceptually drawn on a 3D virtual plane, which is parallel to the planar drawing surface and their visualization uses a perspective projection, we need to apply the “in-

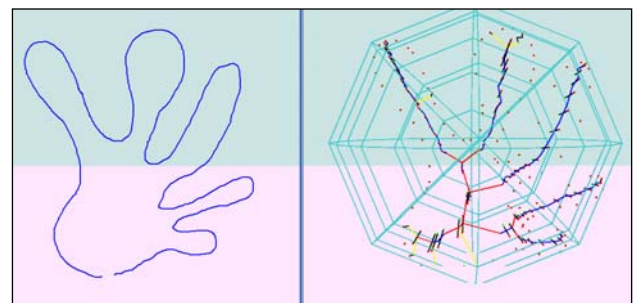


Figure 4: Example skeletonization of a stroke with depth information

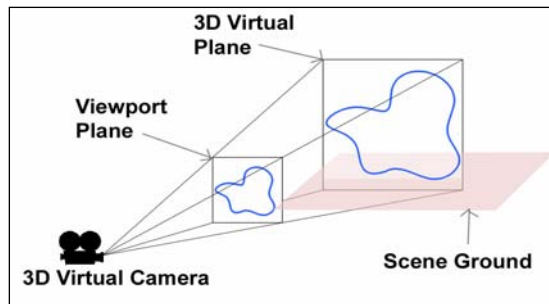


Figure 5: Projecting 2D stroke into 3D virtual plane based on virtual camera definition

verse projection” defined by the virtual camera to each 2D point in the filtered stroke. We map the skeleton onto the 3D virtual plane using the view camera projection and update depth information accordingly.

Figure 5 shows how we perform this mapping. As a result of this step, we obtain a 3D outline that defines the boundary of the blob and a mapped 3D skeleton in scene coordinates. This mapping step yields all information required for the implicit surface representation.

5.5 Creating Variational Implicit Surfaces

For the creation of the implicit representation, we use the variational implicit surface model, based on boundary and normal constraints as presented by Turk. Our approach uses the 3D mapped stroke as boundary constraints. Other boundary constraints use the depth information from the skeleton as shown in Figure 6 both to the left and to the right of the mapped stroke. The look-at vector of the 3D virtual camera defines the left and right direction. For each boundary constraint, we create a normal constraint by shifting the boundary by 0.05 in the normal direction.

After defining all constraint points and types, we have a linear system as presented in section 3. This is then solved, using a matrix resolution method such as LU factorization, to obtain a complete definition of a VIS. At the end of the inflation process, the implicit function is polygonized for visualization as shown in Figure 6.

6. MANIPULATION OF IMPLICIT SURFACES

This section describes possible modification operators on implicit surfaces for use in a modelling application. First, we present an overview of possible operators and discuss the limitations of implicit representation. The following sections describe solutions for the implementation of the merging and oversketching operators, which have been implemented in our demonstrator.

6.1 Operations between Implicit Surfaces

Our approach considers that the natural way of emulating paper and paper drawings is through oversketching. Similarly to Karpenko, we define two main operators, oversketching and merging. When we analyze the way in which designers specify free form surfaces, we verify easily that the main action to redefine the shape is by sketching over the boundary of the shape repeatedly until

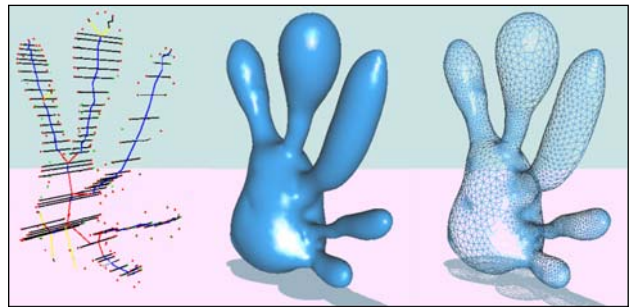


Figure 6: A skeleton and its corresponding polygonized implicit surface

the desired form is obtained. For this reason, our main modification operator is based on the oversketching concept.

The merge operator is an useful extended tool because it allows designers to work on partially defined shapes, which can later be combined to form a unique 3D shape. While the merge operator seems superficially similar to the traditional Boolean union, it actually has different semantics, since it uses the notion of blending blobs. The result remains smooth, thus guaranteeing C^2 continuity to the shape. Of course, this approach brings about some limitations, since sharp edges cannot be created through this method. However, we have to live with such a limitation since the mathematical representation of implicit surfaces is C^2 continuous by definition, which makes it impossible to represent non-continuous features such as sharp edges.

Mathematically, it is simple to define a model supporting Boolean operations using implicit surfaces. However, the C^2 continuity of the VIS model invalidates this issue. The following list presents the most common Boolean operations using implicit surfaces:

Precondition: $F1$ and $F2$ are mathematical functions that define an implicit surface

- Union : $F1 \cup F2 = \min (F1, F2)$
- Intersection : $F1 \cap F2 = \max (F1, F2)$
- Difference : $F1 / F2 = \max (F1, -F2)$

We can verify that the major obstacle for this representation is the use of non-continuous mathematical functions, such as the maximum and minimum. It is possible to support these operators through hybrid representations as Wyvill did in BlobTree [Wyvill99, Galin99]. However, these operations are not familiar for designers that follow a paper and pencil metaphor.

6.2 Using Skeleton Information

The key issue of our approach is to use the skeleton information generated during the inflation process of a blob. We extend this to allow manipulation of blobs. The main advantage is to forgo the use of visualization (mesh) representation for the implementation of the operators, which is one of the main shortcomings in [Karpenko02]. As seen in the previous section, the merge operator could be specified using purely mathematical

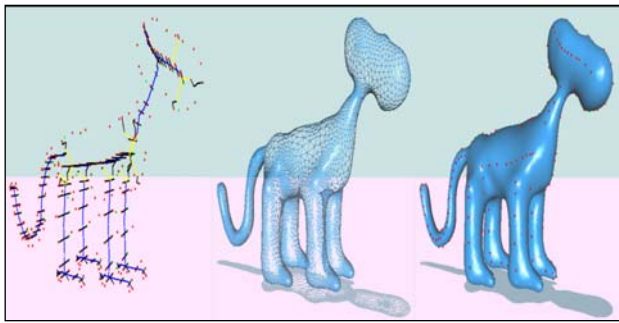


Figure 7: Example of skeleton resulting from several merge operations

means. However, this solution requires complete knowledge of all mathematical information such as the location of the critical points of the function. This requires us to identify the local, global minimum, maximum or saddle points of the implicit function. [Stander97] presents an approach for this. Using Morse Theory, he is able to extract all the topological information of the shape from the mathematic definition. However, this is both very difficult to implement and costly in computational processing time to guarantee that all the information has been extracted. This approach is not suitable for a real time modeller.

Karpenko presents a suitable solution, which combines the expressiveness of a mathematical formulation of surfaces with the flexibility of the triangulated mesh for visualization. However, this solution is only suitable for few steps. Repeated application of merge or oversketching operating, adds many unnecessary constraint points due to this resampling. This may easily scale to thousands of constraints after a few operations. Generating a VIS anew from these constraints becomes impractical with current hardware. Thus, this approach is not appropriate for interaction at real time rates. Another limitation is that this resampling automatically erases all features of the shape smaller than the sampling interval.

To overcome the limitation identified in her approach, we base our solution on the surface skeleton, its associated depth information and its VIS. Since skeletons can be merged or modified without using the mesh information and the VIS is recreated at each step, this guarantees that small features will be preserved by the modification operators without our needing to use the triangulated mesh. While the merge operator blends two skeletons, oversketching modifies or appends new data to the original skeleton. The interesting feature of this approach is that all strokes input for oversketching define a new skeleton, which we then merge to the original. After merging strokes, we drop all points that lie inside the merged stroke.

The following sections present a detailed description of the algorithms for merging and oversketching. After this step, we generate all the constraints for the new shape. These are enough to define all the characteristics of each shape. Then we perform a step to know which constraints will remain valid after merging. This is presented in Fig-

```

Construction of constraint set for merging
(FA, SA, FB, SB) {
  Initialize the result set to empty
  For each CA (boundary constraint) of FA {
    if ( FB(CA) >= 0 )
      then CA and the respective normal is
        inserted in the result set
    else CA and the respective normal
      constraints are rejected
  }
  For each CB (boundary constraint) of FB {
    if ( FA(CB) >= 0 )
      then CB and the respective normal is
        inserted in the result set
    else CB and the respective normal
      constraints are rejected
  }
  return the result set
}

```

Figure 8: Pseudo-code for merging two blobs

ure 8. Using inflation, we create a new blob. The skeletons of both the new and old blobs are merged. The resulting skeleton is then attached to the result blob.

6.3 Merging blobs

For the merging operator, let us consider the variational implicit functions to merge, where FA and skeleton SA define the first blob and FB, SB the second one. Both use constraint points inferred from their respective skeletons.

While the results obtained by our implementation are visually similar to Karpenko's, we can handle situations not supported by her method. However, the resulting mathematical function of the merged blob uses less constraints points, then interactivity is better for modelling application since the cost of computation for implicit calculation is smaller. The main problem with Karpenko's approach lies in that it uses only two constraints to define depth during inflation. This requires sampling the triangular mesh to perform the merge, which limits the guarantee of fidelity of finer features of both blobs in the resulting shape. Moreover, her sampling method uses the vertices resulting from a Marching Cubes polygonization [Lorensen87, Bloomenthal94], which depends on the size of the subdivision and yields too many constraints on the merged shape. While our approach uses more constraints to define the object in the inflation process, the merged blob has fewer constraints. Figure 9

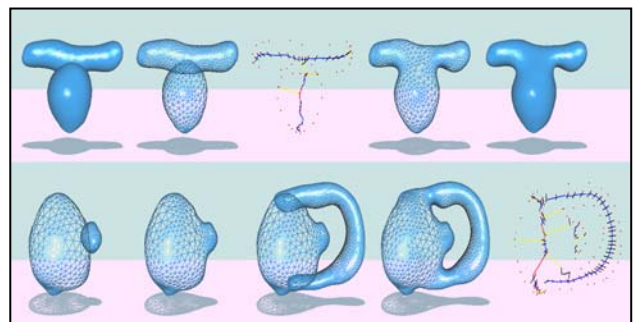


Figure 9: Example of merging operations with resulting skeleton and meshes

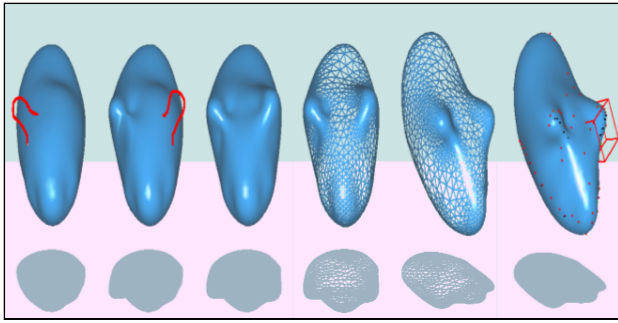


Figure 10: Example of oversketching operation

shows some examples of the merging operation presenting in both shaded and wire frame views.

6.4 Oversketching blobs

Another possibility is to create extrusions. These are the main modelling operator in Teddy. Two input strokes define the extrusion, the first defines the base area affected by the extrusion. The second one, defines the profile of the extrusion. The implementation of this solution presented some limitations in Teddy, corrected recently by [Wang03]. We will show that using the skeleton information for the profile stroke of the extrusion and its area for the base allows specifying the extrusion with only one stroke. Our approach applies skeletonization to the oversketching stroke. The volume defined by the skeleton specifies the base of the extrusion. This information helps to distinguish a boundary redefinition from an extrusion. The oversketching operation involves six different steps. First, we apply stroke filtering, followed by 2D skeletonization as in the inflation process. Then we project the stroke and its 2D skeleton on a plane cutting the surface perpendicularly to the look-at vector of the 3D virtual camera. Next, we identify and reject all constraints of the blob located inside the oversketching area. After, we insert the new constraints defined by the oversketching skeleton. Finally, we use the resulting skeleton to create the new blob.

6.4.1 Projecting oversketch strokes

After filtering the input stroke and creating its skeleton, 2D input points need to be transformed to 3D coordinates. This process is different from the initial step of the

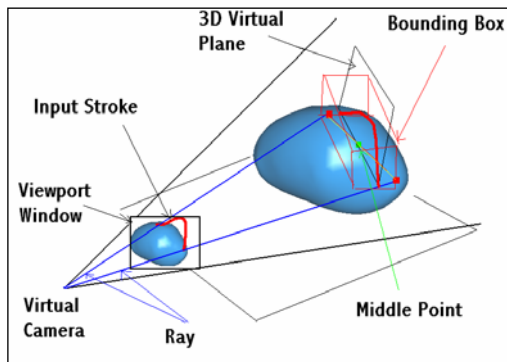


Figure 11: Mapping 2D stroke for oversketching

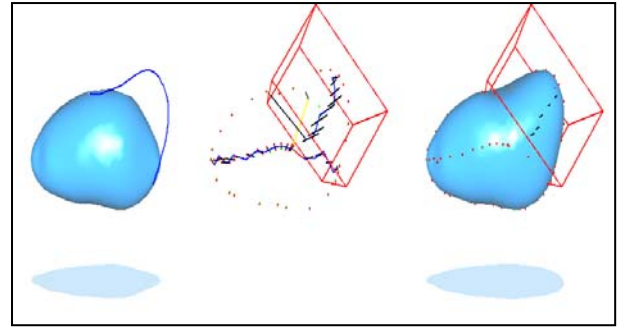


Figure 12: Example of boundary redefinition with oversketching

inflation, since the stroke must be projected on the *contour plane* of the target surface, as shown in Figure 11. The 3D virtual plane is defined using the first and last point of the input stroke. Both points are projected using a function, which computes the intersection between a ray defined using the 2D endpoint coordinate and the position of the 3D virtual camera. The 2D coordinates in the viewport represent the position of the stroke endpoint in the near plane of the camera. We calculate the 3D coordinates of the near plane of the camera according to the size of windows, the fov of the camera and the distance of the near plane to the camera position.

The ray intersections yield the start and end points of the stroke as projected in the surface on the contour plane. We define the 3D contour plane by the middle point between these two and it is perpendicular to the camera look-at vector. We then project the points of the filtered stroke and its skeleton onto this plane. Another possible approach is to define the 3D plane normal to the surface. We tried this approach and computed this plane, using a vector perpendicular to the true normal of the surface at the middle point. This solution is used by [Wang03] as a correction to the Teddy system for the projection of the profile stroke of the extrusion. However, this can lead to incorrect results as users conceptually draw on a plane perpendicular to the view direction.

6.4.2 Constructing the oversketched blob

For the construction of the new blob, we have to define its corresponding set of constraints. First, we start by verifying each constraint of the original blob, by checking to see if they are inside the bounding box of the pro-

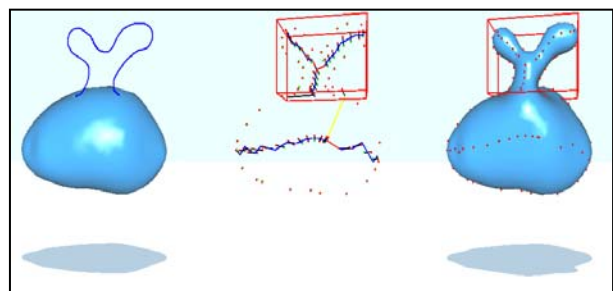


Figure 13: Example of implicit extrusion using oversketching

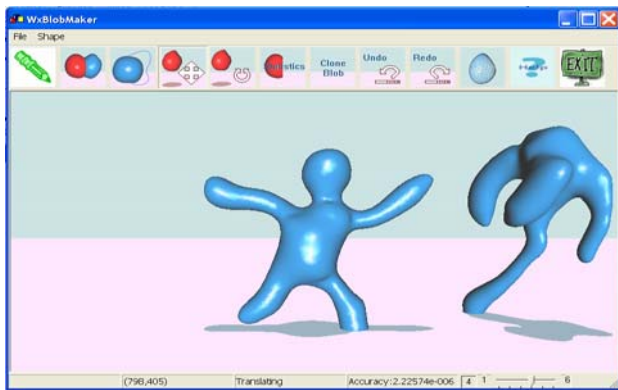


Figure 14: BlobMaker interface

jected skeleton. We reject constraints located inside the bounding box. All others will remain in the resulting blob. This test is very simple since we only need to verify whether a given 3D point is inside an oriented bounding box.

The skeleton of the oversketching stroke generates new constraints for the blob, following a different strategy depending on whether we are in creating an extrusion or redefining the boundary of a shape. In case of boundary redefinition, the stroke does not generate any new depth constraints. Only boundary constraints are defined from the input stroke. The extrusion operation creates boundary constraints from both the stroke and the original skeleton. For each boundary constraint, we create a normal constraint using the normal vector of the projected stroke in the 3D virtual plane. This is then combined with the real normal to the surface computed at the first and last point of stroke as projected over the surface.

Figures 12 and 13 show two examples of oversketching; the first presents a boundary redefinition and the second an extrusion.

We can verify that the skeleton associated the boundary redefinition does not generate any constraints in depth (red points near black branch of the skeleton in the figures). However, in the case of extrusion, we insert additional constraints to define the depth. The case for this distinction is simple to state. If the depth constraints for extrusion were to disappear, the depth of the blob in the oversketched volume would only be influenced by the depth constraints of the original blob. This would be incorrect, because the result of oversketching a shape boundary may yield different topology features, thus defining an implied extrusion as seen in Figure 13.

We assess the need for additional constraints for extrusion operator by looking at the characteristics of the input stroke. We measure the importance of the stroke by computing the ratio of the distance between its first and last points to the height of the volume it influences. In Figure 12, the base of the stroke is larger than the length of its skeleton; the resulting operation will be a boundary redefinition. On the other hand, in Figure 13, since the base is smaller than its length, the operation will be an extrusion.

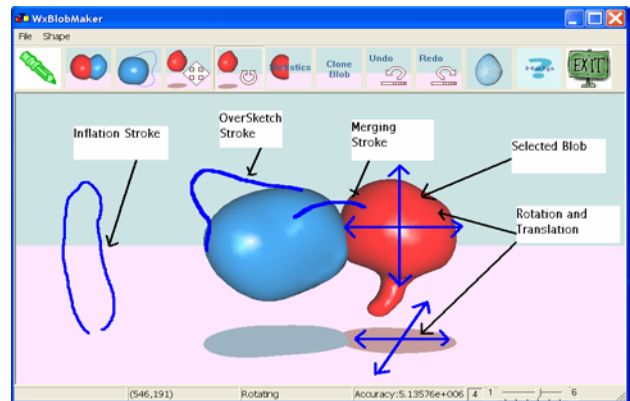


Figure 15: Stroke types in BlobMaker

Finally, we define the constraints for the oversketched blob. Then we generate the new blob and attach the merged skeleton to it.

7. BLOBMAKER

BlobMaker is an application for free form modelling based on variational implicit surfaces. It allows the user to create 3D blobs using the inflation process described in section 5. The modeller allows the user to modify shapes using the merging and oversketching operators presented in section 6. Visualization and rendering of implicit surfaces are based on OpenGL API using triangular meshes.

Our approach provides different tools to support creating and editing blobs. Users can translate, rotate and copy objects, with support for unlimited Undo and Redo. It is possible to save and retrieve Blobs to/from STL [3DSys88] and VRML [VRML97] formats. Users can visualize Blobs in several different modes such as Wireframe, Polygonal with Gouraud shading using OpenGL, or more faithful rendering modes such as Ray Tracing and Phong Shading. Users can also control mesh quality either through our approach or through Marching Tetrahedra [Bloomenthal94].

7.1 Interface

The BlobMaker interface divides the screen into two areas as shown in Figure 14. The first area, on the top of the window, is a toolbar that allows the user to select the following tools: drawing, merging, oversketching, translation, rotation, copy, undo/redo functionality and wireframe rendering mode. The second area represents the working space where the user can draw, manipulate and visualize the 3D free form shapes. The working space offers a perspective view of the modelling scene. The virtual camera defines the user point of view, which is centred in the zero-plane XY. To ease the depth perception and positioning relationship between blobs, the zero-plane XZ is drawn as the ground of the scene in a rose colour. This virtual camera is used for the projection of the 2D viewport based input inserted by the user.

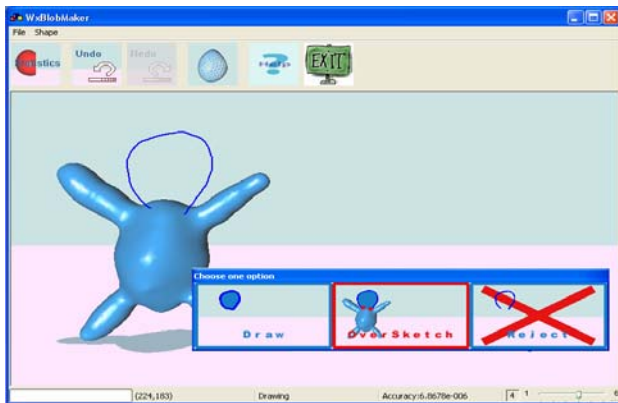


Figure 16: Expectation list for an ambiguous stroke

7.2 Stroke-based interaction

Figure 15 presents the different kinds of user input possible during interaction with the modeller. All operations are selected using the toolbar area, which removes any ambiguity from stroke semantics; i.e. the distinction between a merging, inflation or oversketch stroke is made explicitly through menu selection.

When the user selects the drawing operation, subsequent 2D input will be interpreted as the boundary of a 3D shape. Of course, some strokes are invalid in this context, such as straight lines. Input strokes can be closed or open. However, open curves must converge to a well-defined closed boundary. To detect this feature, begin and end segments of an input stroke must define intersecting straight lines that “close” the stroke. For the merge operation, the user needs to draw an arbitrary curve, linking two distinct blobs. The system uses begin and end points to pick each blob. Similarly, we use end-point to picking and select objects as targets for oversketching, translation, rotation and copy operations.

Since points are specified on a planar viewport, depth positioning can be ambiguous. Our modeller solves this problem using the “ground” visualization. The user can select a blob by its shadow, which is the planar projection of the blob onto the ground. This feature makes it easy to do relative positioning of shapes and offers two different ways to select a blob: either by clicking on the rendered projection of the blob or on its shadow.

7.3 Improving interaction using stroke analysis

By using skeleton and stroke information, we can improve interactions and avoid using the toolbar. We have found from experience that users prefer to draw, since this matches best the affordances of the stylus. Consequently, they sometimes forget to select the “right” command beforehand. This is the most common source of errors when using the interface. We have implemented a second mode in the application where the toolbar only offers undo/redo functionality and switching between wireframe and shaded views.

In this mode, some strokes are ambiguous, for example oversketching gestures can sometimes initiate inflation. To solve this, we use dynamic expectation lists as intro-

duced by [Pereira00] to present all possible interpretations of an ambiguous stroke to the user as exemplified by Figure 16. If the stroke is not ambiguous, the operation takes effect immediately. Thus, simple actions can yield different semantics. A click on a blob selects that object. Subsequent clicks create copies of the selected object. In addition, a straight line or an arc joining two blobs define a merge operation. A stroke whose endpoints lie on the same blob defines an oversketch operation on that blob. A closed stroke or a similar curve creates a new shape. In this manner, all the main interactions can be sketch-based. This mechanism can also avoid useless undo/redo commands due to erroneous use of buttons in the toolbar.

8. CONCLUSIONS AND FUTURE WORK

The proposed interface and operations are suitable for a complete free form modelling system. We feel that our sketch-based approach provides a natural interface for traditional designers not familiar with either geometrical constraints or the internal representation details of NURBS. In this work, variational implicit surfaces limit free forms to closed surfaces and smooth shapes. In the future, we would like to extend our system using hierarchical CSG combinations of VIS and support discontinuous representations using cuts as in Teddy. Another interesting development is to add primitives for NURBS creation and manipulation, since these are the more commonly used geometric representation in CAD and manufacturing. However, this is a non-trivial task, especially if we are to derive compact constrained representations from free-form primitives. Following Igarashi, we would also like to add non-photorealistic rendering of objects to better match the pencil-and-paper sketching metaphor. Work is already underway on this front.

In this paper, we have presented a stroke based modeller application for 3D free form modelling using variational implicit surfaces. The different operations proposed for free-form manipulation have shown oversketching to be a powerful and suitable tool for 3D modelling. Informal evaluations show great promise for this tool as being more adequate for traditional designers, who are familiar to the pencil and paper metaphor. This too, should be assessed by extensive usability evaluations in the future.

ACKNOWLEDGMENTS

This work was supported in part by European Commission through grant # IST-2000-28169 (SmartSketches) and by the Portuguese Science Foundation under grant POSI/34672/SRI/2000.

REFERENCES

- [3DMax] www.discreet.com
- [3DSystems88] Stereolithography Interface Specification, 3D Systems Inc., 1988.
- [Bloomenthal94] J. Bloomenthal, An Implicit Surface Polygonizer, *Graphics Gems IV* (P. Heckbert, ed.), Academic Press, New York, 1994.

- [Bloomenthal98] K. Bloomenthal, R.C. Zeleznik et al.: SKETCH-N-MAKE: Automated machining of CAD sketches. *Proc. of ASME DETC'98*, 1-11, 1998.
- [CATIA] www.catia.com
- [Contero01] M. Contero, F. Naya et al., Calligraphic Interfaces and Geometric Reconstruction", *12th ADM International Conference on Design Tools*, September 2001.
- [Corel] www.corel.com
- [Douglas73] D. Douglas and T. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Canadian Cartographer* 10, 2, 112-122, 1973.
- [Farin99] G. Farin, Curves and Surfaces for CAGD, A Practical Guide, *Academic Press, New York*, 5th Edition p. 227, 1999.
- [Galin99] E. Galin, A. Leclercq and S. Akkouche, Blob-Tree Metamorphosis, *Implicit Surfaces'99 Conference*, 4: 9-16, Bordeaux, France, September 1999.
- [Herot76] C. Herot, Graphical input through machine recognition of sketches, *ACM SIGGRAPH Computer Graphics*, vol. 10, n 2, 97-102, 1976.
- [Igarashi01] T. Igarashi, J. F. Hughes, 3D drawing: A suggestive interface for 3D drawing, *Proc. of the 14th annual ACM symposium on User interface software and technology*, Orlando, Florida, November 2001.
- [Igarashi99] T. Igarashi, S. Matsuoka et. al. Teddy: A sketching interface for 3D freeform design. *Proc. of SIGGRAPH 99*, 409-416, August 1999,
- [Karpenko02] O. Karpenko, J. F. Hughes, R. Raskar, Free form sketching with variational implicit surfaces. *Eurographics 2002*.
- [Lighthwave3D] www.newtek.com
- [Lorensen87] W. E. Lorensen and H. E. Cline. Marching Cubes: A high resolution 3D surface construction algorithm. *Computer Graphics*. 21, 4(1987) 163-169, 1987
- [Maya] www.aliaswavefront.com
- [Michalik02] P. Michalik, D. Kim, B. Bruderlin: Sketch- and Constraint-based Design of B-spline Surfaces. In *Proceedings of International Conference on Solid Modelling* 2002.
- [Necker32] L. A. Necker, Observations on some Remarkable Phenomena seen in Switzerland: and an Optical Phenomenon which Occurs on Viewing of a Crystal or Geometrical Solid, in *Philosophical Magazine*, 3rd series; 1:329-343, 1832.
- [Negroponte73] N. Negroponte, Recent advances in sketch recognition, *Proceedings of the AFIPS 1973, National Computer Conference*, 663-675, 1973.
- [Nishimura85] H. Nishimura, M. Hirai et al. Object modelling by distribution function and a method of image generation, *Trans. IEICEJ.*, 68:718, 1985.
- [Pereira00] J. P. Pereira, J. Jorge, V Branco and F. Nunes Ferreira, Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons, WSCG2000, Czech Republic, February 2000.
- [Prasad97] L. Prasad. Morphological analysis of shapes. *CNLS Newsletter*, n° 139, 1-18, July 1997.
- [Sanchis02] F Naya Sanchis, J. A Jorge et al, Direct Modelling: from Sketches to 3D Models", 1st Ibero-American Symposium on Computer Graphics, Guimarães (SIACG02), July 2002.
- [Sketchup3D] www.sketchup.com
- [SoftImage] www.softimage.com
- [Stander97] B. Stander, J. C. Hart. Guaranteeing the topology of an implicit surface polygonization, *Proc. SIGGRAPH97*, 279-286, August 1997.
- [Sutherland63] I.E. Sutherland, Sketchpad: a man-machine graphical communication system, *Proc. Spring Joint Computer Conference, AFIPS*, 329-346, 1963.
- [Turk99] G. Turk, J. O'Brien, Shape Transformation Using Variational Implicit Functions, *SIGGRAPH 99*, 335-342, August 1999.
- [Turk01] G. Turk, H. Quynh Dinh, J. O'Brien et al., Implicit Surfaces that Interpolate, *Shape Modelling International 2001*, Genova, Italy, 62-71, May 2001.
- [Turk02] G. Turk, J. F. O'Brien, Modelling with Implicit Surfaces that Interpolate, *ACM Transactions on Graphics*, Vol. 21, No. 4, 855-873, October 2002.
- [VRML97] The Virtual Reality Modeling Language, International Standard ISO/IEC 14772-1:1997
- [Wang03] C. C. L. Wang, M. M. F. Yuen, Freeform extrusion by sketched input, *Computers & Graphics*, vol. 27, no. 2, 255-263, April 2003.
- [Wyvill98] B. Wyvill, A. Guy and E. Galin, The Blob-Tree warping, blending and Boolean operations in an implicit surface modelling system, *Computer Science Technical Reports*, 1998-618-09, March 1998.
- [Wyvill99] B. Wyvill, A. Guy and E. Galin, Extending the CSG Tree (Warping, Blending and Boolean Operations in an Implicit Surface Modelling System), *Computer Graphics Forum*, 18(2), 149-158, June 1999.
- [Yngve02] G. Yngve, G. Turk, Robust Creation of Implicit Surfaces from Polygonal Meshes, *IEEE Transactions on Visualization and Computer Graphics*, vol. 8, no. 4, 346-359, October-December 2002.
- [Zeleznik96] R.C. Zeleznik, K. Herndon et. al. SKETCH: An interface for sketching 3D scenes. *SIGGRAPH 96 Conference Proceeding*, 163-170, 1996.

Web-based direct manipulation of feature models

Rafael Bidarra

André van Bunnik

Willem F. Bronsvoort

Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Mekelweg 4, NL-2628 CD Delft, The Netherlands
<http://graphics.tudelft.nl>

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

Abstract

Providing advanced 3D interactive facilities to users of a client-server collaborative modeling system presents a great challenge when thin clients are involved, mainly due to their lack of both a proper CAD model and the adequate modeling and solving functionality. This paper presents a new approach that provides a convenient representation of feature model data suitable for direct manipulation of feature models at such clients. It combines all advantages of a thin client approach with the sort of 3D direct manipulation facilities usually only found in powerful standalone CAD systems.

Keywords

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

1. INTRODUCTION

Current CAD systems, holding a sizeable modeling kernel which maintains an unabridged CAD model, provide many advanced interactive facilities for model manipulation. Requirement for this, however, is that they run on powerful, typically standalone, workstations.

Current demands for supporting design collaboration, on the other hand, require an efficient networked environment in which geographically distributed members of a development team can work together on the design of a part. In an ideal collaborative modeling framework, several team members should be able to remotely browse and manipulate a model, via Internet, *as if* they were working directly at a powerful CAD station. A web-based system, for example, would greatly facilitate this, by providing access to all sorts of product information in a uniform, simple and familiar framework. The above mentioned characteristics of current CAD systems prevent them from matching these demands.

A number of commercial tools that are now emerging provide some limited support for collaborative design activities. For example, tools for collaborative model annotation and visualization are now becoming available, providing concepts such as interactive 3D visualization, shared cameras and telepointers. However, such tools are primarily focused on visualization and inspection, basically using polygon mesh models, and do not support real modeling activities. In other words, they are valuable assistants for teamwork, but no real modeling systems.

Meanwhile, new prototype systems are being developed which directly concentrate on collaborative modeling facilities. In such systems, mostly following a client-server architecture, a crucial role is played by their complex concurrency and synchronization mechanisms.

Current commercial client-server modeling systems which offer some real collaborative modeling facilities as, for example, OneSpace [CoCreate03] and IX SPeeD [ImpactXoft03], use considerably fat clients, requiring heavy data synchronization among clients, and are severely constrained by the model format into which they convert shared CAD models.

Thin client web-based approaches, in contrast, are gaining particular attractiveness, one of the main reasons being that they usually provide a more efficient solution to data synchronization problems by using a single, server-based central model. In addition, directly loading the client application via Internet avoids complex installation and maintenance procedures, and therefore increases portability. Typically, in such systems, a development team member should be allowed to graphically specify a modeling operation, appreciate its consequences and, upon approval, issue it for execution at the server.

There are two important characteristics of thin clients in such systems which make user interactivity with the model particularly challenging.

First, they lack a real modeling kernel, and cannot therefore locally execute actual modeling operations. Instead, because such operations are executed only at the server, this is required, after each operation, (i) to export to the

clients the necessary model data (indispensable for visualization and user interaction), and (ii) to guarantee that such data is always kept up-to-date.

Second, these thin clients lack a comprehensive constraint solver. As a result, it is in general not possible to locally anticipate all consequences a given operation may have in the whole model. For example, when several features are related through geometric constraints, displacing one of them will typically affect a few others, but the overall result can only be precisely determined by means of a constraint solver.

Summarizing, a careful choice of client model data is required in order to provide thin clients with proper user interaction mechanisms.

webSPIFF, a web-based, collaborative feature modeling prototype system developed at Delft University of Technology, offers such a thin client framework. A complete description of its client-server architecture and functionality can be found in [Bidarra02]. In particular, the reader is referred to this reference for all aspects related to its consistency management, data synchronization and consistency maintenance facilities. In Section 3 only a short overview of the system is provided.

Model data used so far by the webSPIFF clients described in [Bidarra02] was mainly aimed at providing its users with (i) interactive visualization of the model, (ii) interactive selection of feature faces during operation specification, and (iii) textual information on each feature's parameter values. The main limitation of such data is that it can only be modified by means of updates received from the server, never directly by the client itself.

The goal of this project is twofold: to extend the model data at the webSPIFF clients (i) with a feature representation suitable for direct manipulation (Section 3) and (ii) with advanced interaction mechanisms supporting such feature direct manipulation (Section 4).

2. INTERACTIVE FACILITIES IN 3D WEB SYSTEMS

In this section interactive facilities offered by a few web-based 3D systems are briefly surveyed. The reader is referred to the Web3D Repository at the Web3D Consortium site [Web3D03] for an overview of other similar systems.

Kaon

The Kaon Composer [Kaon03] is a Java applet aimed at supporting virtual product presentations via Internet. It uses Kaon's Master Model native format to provide interactive visualization for zooming, panning and rotating a 3D mesh model directly in the web browser. Pre-defined regions of the model can be made sensitive to actions, as for example displaying attached annotations or triggering an animation. In addition, queries on dimensions can be also interactively performed by clicking and dragging on the model.

RealityWave

RealityWave [RealityWave03] developed VizStream Platform, a client-server technology aimed at supporting collaborative browsing and visualization of 3D models by loading a simple viewer in a web browser. In addition to the same functionality mentioned above for Kaon, VizStream provides also inspection facilities as, for example, clipping the 3D model by means of an interactively adjustable clipping-plane. The user can also attach markup to regions of his choice on the visualized model. Finally, the possibility of selecting which (and how) components of the model are visualized is also provided.

Art of Illusion

Art of Illusion [ArtOfIllusion03] is an open source studio application integrating modeling and rendering functionality. Although it is not strictly speaking web-based, we include it here as, to the best of our knowledge, it is the first fully Java-implemented 3D modeling system available. Being a moderately small application, it offers advanced direct manipulation and complex modeling operations (including face lifting and Boolean operations), comparable to those found in many commercial programs.

3. CLIENT REPRESENTATIONS FOR FEATURE MODELS

The webSPIFF server has two main components: the SPIFF modeling system and the Session Manager. The SPIFF modeling system provides all feature modeling functionality, including multiple views on a part, advanced visualization [Bronsvort02] and validity maintenance of feature models [Bidarra00]. It maintains a central product model, which includes a cellular model for the geometric representation of a part, and canonical shapes representing the individual features in each view. The Session Manager provides functionality to start, join, leave and close a collaborative session, to coordinate the session, and to manage all communication between SPIFF and the clients. In particular, the Session Manager collects all operations requested by the various clients, and schedules them for execution at the SPIFF system.

webSPIFF clients operate locally as much as possible, e.g. regarding visualization of, and interaction with, their feature model, and only high-level messages, e.g. for specifying modeling operations, as well as a limited amount of model data necessary for updating the client information, are sent over the network. As soon as real feature model computations are required, such as for executing modeling operations, conversion between feature views and 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 characteristic of this architecture is that by using a central product model, inconsistencies are avoided among multiple versions of the model data at different clients.

Both the clients and the Session Manager of webSPIFF were implemented in Java, using its Remote Method In-

vocation (RMI) facilities for communication, and Java3D for model visualization.

In order to support user interaction during collaborative modeling sessions, each client receives from the server the necessary model data. This data has to be carefully derived from the feature model, in order to satisfy two somehow conflicting goals: (i) it should contain all aspects of the feature model which are relevant for direct manipulation purposes; (ii) it should be compact enough to be quickly updated in all clients whenever the model is modified at the server.

In this section, a combination of model data is presented that fulfils these goals. In particular, the new notion of *feature skeletons* is presented, and examples are given of how they represent the relations among feature instances in a model. The interactive facilities provided by feature skeletons will be dealt with in Section 4.

3.1. Graphical data

Graphical data consists of feature model images that are rendered at the webSPIFF server in GIF format, and displayed in camera windows at the clients. These images provide very powerful visualizations of a feature model [Bronsvort02]. Many visualization options can be specified. For example, selected features may be visualized with shaded faces, and the rest of the model as a wire frame or with visible lines only. Also, additional feature information, such as closure faces of holes, can be visualized. A separate image is needed for each camera, and it must be updated every time the model or the camera settings are changed.

These model images provide the camera background on top of which other visualization and interaction techniques are available at the webSPIFF clients.

3.2. Geometric data

webSPIFF clients dispose of two representations of the model geometry: the visualization model and the selection model. Each one has a specific purpose in the camera windows at the clients [Bidarra01]. Both the visualization model and the selection model are generated by the webSPIFF server in VRML format and loaded by a client into its camera's scene graph.

The *visualization model* represents the global shape of the product model. It is used at the clients for interactively modifying the camera viewing parameters (e.g. rotating and zooming). All cameras on a particular client use the same local visualization model, but each camera displays it with its own viewing parameters.

The *selection model* is a collection of objects representing the canonical shapes of all features in a given view of the product. Its purpose is to support interactive selection of feature faces on a feature model image, during the specification of a modeling operation. Again, the selection model is identical for all cameras on a client, each applying its own viewing parameters.

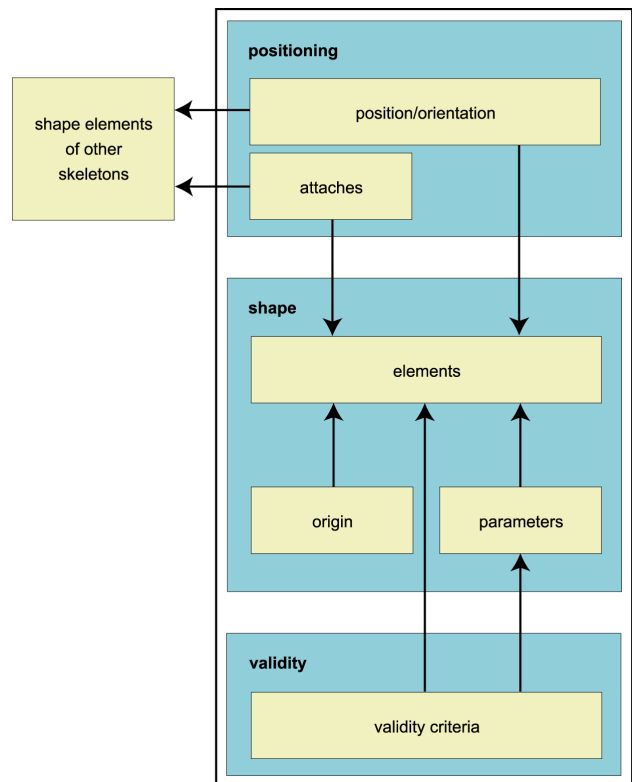


Figure 1 - Generic structure of a feature skeleton

3.3. Feature skeletons

A feature skeleton is a parametric representation of a feature instance which is linked to a simplified geometric model of its shape, in such a way that the latter can be modified by interactively manipulating the former.

Since skeletons are meant to represent feature instances at the client, the structure of skeletons bears resemblance with the generic structure of a feature class, as described in [Bidarra00]. This structure is read from the server during client initialization, for each class in the feature library, after which the client is able to instantiate the skeleton of every feature instance in a feature model.

All skeletons consist of three main components: a *shape* component, a *positioning* component and a *validity* component; see Figure 1. The shape component describes the feature shape in terms of (i) a number of so-called *shape elements* (e.g. the axis reference and the top, bottom and side faces of a cylinder shape), (ii) a number of *parameters* (e.g. the radius and the height of a cylinder shape), and (iii) an *origin*, specified as the intersection point of some shape elements (e.g., for the cylinder shape, the intersection of the axis and the top face).

Each shape parameter conveys a relation between two shape elements. For example, the height of a cylinder shape expresses the distance between its top and bottom faces. Although several feature classes may be based on the same shape type (e.g. block or cylinder), each feature class uses its shape in a different way. This is also reflected in the corresponding feature skeletons, and spe-

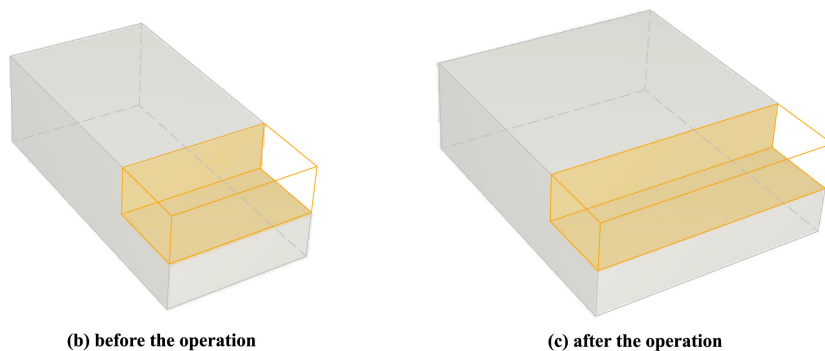
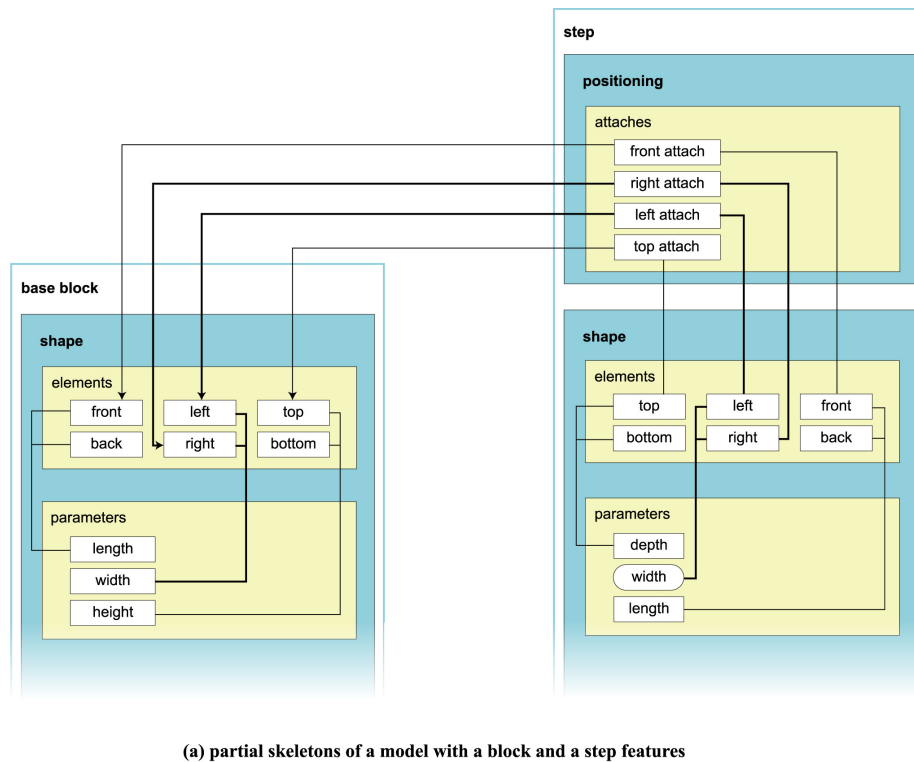


Figure 2 - Propagation of dimension modifications between dependent features

cifically, in the way the skeleton parameters may be adjusted by the user. For example, the skeletons of a blind hole and of a through hole both have a similar cylinder shape component. However, the blind hole skeleton provides two adjustable parameters, radius and depth, whereas for the through hole skeleton only the radius parameter is adjustable, its actual height being derived from the attaches of the through hole.

In short, skeleton parameters may be either adjustable or derived, and these settings are specified in each feature class, together with its own attach and positioning scheme, which will now be described.

The positioning component of a skeleton describes the geometric relations of a feature with the rest of the model. Such relations represent the attach and geometric constraints used at the server to hierarchically structure the actual feature model. An attach constraint is a kind of coplanar geometric constraint which takes into account

the nature of the two features it relates in order to determine the orientation of the attach. Examples of geometric constraints are *distance-face-face* and *angle-face-face* constraints between two planar faces. Because the clients do not dispose of a geometric constraint solver, recording such relations amounts to permanently maintaining the relative position, orientation and dimensions of each feature in terms of the features to which it is explicitly related. Basically, a skeleton achieves this by relating the parameters and the origin of its shape to elements of other features by means of geometric transformations.

Among other things, this information is crucial to know which other features a given feature depends on, and thus, to allow for tracking the correct propagation of changes when any of those features is modified. For example, Figure 2 presents a model with a rectangular step attached to a base block, together with the respective skeletons. When the block width parameter (relating its left and

right faces) is increased, the step width should be increased as well, because its left and right attaches refer to those block faces. As will be explained in the next section, when a user is modifying a feature, its dependent features are also highlighted in the camera, so that possible modifications in their derived parameters become apparent to him as well.

The validity criteria referred to in Figure 1 reflect the validation constraints specified by a feature class for each of its instances. An example of these are dimension constraints, which prescribe a specific range for the value of a given feature parameter. Such criteria can be profitably used during direct manipulation of the model, to prevent the user from performing feature modifications which would turn the model invalid. It should be noted, however, that not all advanced validity criteria specified in feature classes can be maintained and assessed remotely at the clients. This is, for example, the case of most topology-related validity criteria involved in feature interaction management, which can only be properly maintained on the central model at the server [Bidarra00].

Summarizing, feature skeletons provide a compact parametric representation of the features in a model, and their relations. As a result of their integration with the feature geometry stored in the selection model, webSPIFF clients are able to support direct manipulation of the feature model, as will be discussed in the next section.

4. INTERACTIVE FEATURE EDITING FACILITIES

Feature skeletons, as described in the previous section, contain all elements required to provide interactive feature editing facilities to webSPIFF clients. To achieve this, a number of methods has been implemented at the clients that visualize a feature skeleton, including all its adjustable parameters, and allow for their interactive modification. As explained in Section 1, the main goal of this functionality is to achieve that webSPIFF users specify their modeling operations in an interactive manner and be given as much insight as possible into their result, without recurring to the server.

Direct manipulation on a skeleton is performed by means of handles, each of which is associated to exactly one parameter, so that dragging a handler adjusts the value of the corresponding parameter.

There are three sorts of handles: (i) for attaches and references, (ii) for positioning and (iii) for shape parameters. Handles for attaches and references are aimed at selecting the shape elements of other skeletons to be used in attaches and positioning references. These handles describe thus how the feature skeleton relates to those of the other features in the feature model. Handles for positioning are aimed at setting the value for positioning parameters. These handles determine the distances and angles used in skeleton position and orientation parameters. Handles for shape parameters are aimed at setting the values for the adjustable parameters of a feature skeleton.

The basic procedure to interactively modify a feature in the model is straightforward. First, the user selects the

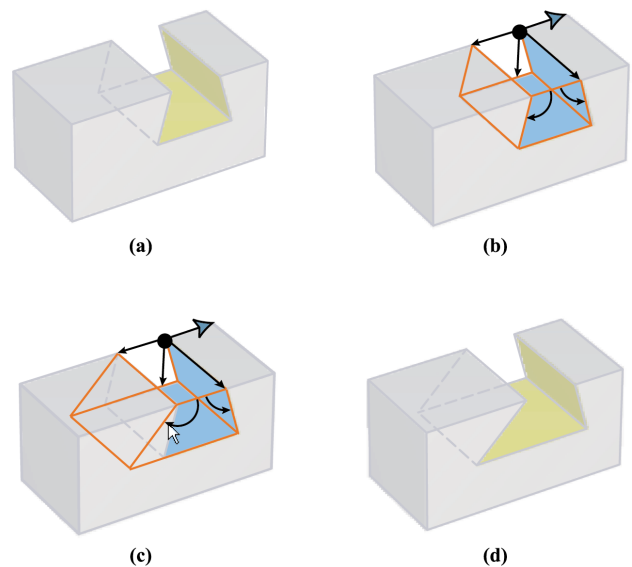


Figure 3 – Modifying a feature at the client

feature by clicking it on a camera, showing a model image. If more than one feature is located behind the selected camera position, then clicking on the same position will scroll through all intersected features. The selected feature is highlighted by displaying its canonical shape (from the selection model) and, on top of it, the wireframe representing its skeleton. This wireframe contains the different sorts of handles mentioned above, which can be used to modify the feature. Dragging a handle results in the immediate modification of the skeleton. As the shaded image does not change, it is clearly visible what is the effect of changing the parameter whose handle is being dragged. When the user has finished manipulating the feature, the specified modify operation, containing the new parameters, is sent to the server, where it is executed on the central model. As a result, new model data is generated and sent back to the client, where it is visualized.

The example in Figure 3 illustrates this with a model containing a block and a trapezoidal slot, for the modification of one of the angle parameters of the slot (Figure 3.a). First, the slot is selected and its skeleton is displayed, together with its handles (Figure 3.b). Subsequently, the handle of the angle parameter between the left and top faces is dragged to change its value. The original feature model depicted in the model image remains unchanged, so that the difference between the original angle and the new angle shown in the skeleton is clearly perceptible (Figure 3.c). Finally, the operation is submitted for execution at the server, after which the resulting model is displayed at the client's camera (Figure 3.d).

Adding a new feature to the current feature model is in many regards analogous to modifying an existing one. It only requires an initial step, in order to first choose the feature class of the new feature instance and to select its attach faces. Once this is done, the skeleton of the new instance is displayed, with its parameters set to their de-

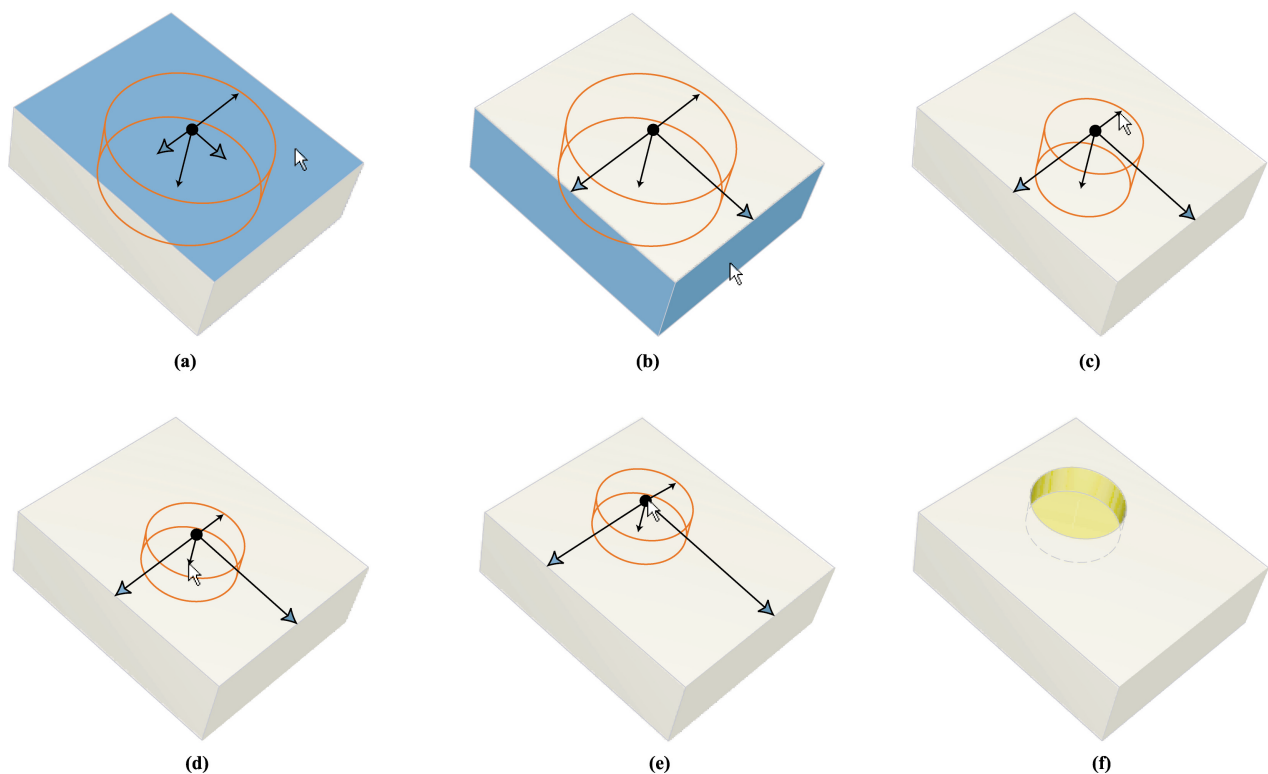


Figure 4 - Adding a new feature to the model

fault values. The user is then required to interactively select the references required for positioning the new feature, after which its shape parameters can be adjusted as desired.

Figure 4 provides a simple example of this, illustrating how a blind hole can be attached to a base block. As soon as the desired attach face is selected on the block, the skeleton of the blind hole is displayed, exhibiting its default shape parameter values (Figure 4.a). Subsequently, the user selects the required pair of (non-parallel) reference faces relative to which the blind hole origin is positioned (Figure 4.b), in this case, the front and left face of the block. The parameters of the hole can then be adjusted by dragging the corresponding handles: dragging the radius handle closer to the origin decreases the radius of the hole (Figure 4.c), dragging the depth handle closer to the origin decreases its depth (Figure 4.d). By dragging the blind hole origin to another position on the attach face, the entire skeleton is displaced, and the parameters representing the distance to the reference faces are modified accordingly (Figure 4.e). When the user is satisfied with the settings of the blind hole, he can submit the operation for execution at the server, after which a new model image of the resulting model is displayed at his camera (Figure 4.f).

5. CONCLUSIONS

A novel approach has been presented that enables thin clients in a client-server collaborative modeling environ-

ment to provide their users with direct manipulation facilities on a feature model. These are made possible by the use of feature skeletons, a compact parametric representation of the features in a model and their relations, which are maintained at the system's thin clients. Feature skeletons can be visualized and manipulated in real-time, allowing users to locally specify a modeling operation in an interactive manner, and giving them insight into its results, prior to recurring to the modeling server for its actual execution.

Future work in this project will have to clarify whether it is always possible to determine in advance which validity criteria can be handled at the client and which do not.

6. REFERENCES

- [ArtOfIllusion03] Eastman, P. Art of Illusion. www.artofillusion.org, May 2003.
- [Bidarra00] Bidarra, R. and Bronsvort, W.F. Semantic feature modelling. *Computer-Aided Design*, **32**(3): 201–225, 2000.
- [Bidarra01] Bidarra, R., van den Berg, E. and Bronsvort, W.F. Interactive facilities for collaborative feature modeling on the web. Proceedings of 10th EPCG, Madeira, J., Marques, J.S., Dias, M.S., and Jorge, J.A. (Eds.), pp. 43-52, October 2001.
- [Bidarra02] Bidarra, R., van den Berg, E. and Bronsvort, W.F. A Collaborative feature modeling system.

Journal of Computing and Information Science in Engineering **2**(3): 192-198, 2002.

[Bronsvoort02] Bronsvoort, W.F., Bidarra, R. and Noort, A. Feature model visualization. *Computer Graphics Forum* **21**(4): 661-673, 2002.

[CoCreate03] CoCreate Software Inc., Fort Collins, CO, USA. www.cocreate.com, April 2003.

[ImpactXoft03] ImpactXoft, San Jose, CA, USA, www.impactxoft.com, January 2003.

[Kaon03] Kaon Interactive Inc. Maynard, MA, USA, www.kaon.com, April 2003.

[RealityWave03] RealityWave Inc. Cambridge, MA, USA, www.realitywave.com, April 2003.

[Web3D03] The Web3D Repository. www.web3d.org/vrml/vrml.htm, May 2003.

Sessão 2

MULTIMÉDIA E VISUALIZAÇÃO GRÁFICA

VGLib2D - Biblioteca de classes para a visualização gráfica de grandes volumes de dados em 2D

Sofia Gameiro

Centro de Computação Gráfica

Departamento de Coimbra

Centro de Empresas de Taveiro

Estrada de Condeixa

3040 – 912 Coimbra | PORTUGAL

{ Sofia.Gameiro, Luis.Almeida}@coimbra.ccg.pt

Luís Almeida

Adérito Marcos

CCG / Universidade do Minho

Escola de Engenharia

Departamento de Sistemas de Informação

Campus de Azurém

4800-058 Guimarães, Portugal

marcos@dsi.uminho.pt

Resumo

Este artigo apresenta a implementação de uma biblioteca de classes, em linguagem C#, recorrendo às recentes tecnologias .Net, destinada à criação de documentos/imagens em formato SVG (Scalable Vector Graphics), para a representação, visualização e impressão de grandes volumes de dados em duas dimensões (2D).

O objectivo é tirar partido das vantagens do SVG para efeitos de implementação de aplicações, por exemplo baseadas em Web, onde seja necessária a representação gráfica de grandes volumes de informação, de forma dinâmica e em tempo real, bem como a sua disponibilização imediata para visualização, impressão e download.

Palavras-chave

SVG (Scalable Vector Graphics), XML (Extensible Markup Language), Visualização de Informação 2D, C#, .Net

1. INTRODUÇÃO

A *World Wide Web* constitui, sem sombra de dúvida, o maior repositório e veículo de circulação de informação da actualidade.

A facilidade de acesso, o baixo custo de utilização e a facilidade da navegação tornaram-na no instrumento de pesquisa de informação mais usado em todo o mundo.

Contribui fortemente para este fenómeno a constante evolução das formas de apresentação e disponibilização de informação de todo o tipo e formato. Tal evolução abrange aspectos importantes como o *design* e aparência da apresentação da informação, a facilidade de navegação entre conteúdos, a possibilidade de animação e interacção com a mesma, a compactação e rapidez de *download*, entre outros. Paralelamente, há ainda que garantir a possibilidade de pesquisar essa informação, seja qual for o tipo de formato em que esta se encontra.

No que respeita a informação textual, tanto a sua visualização como pesquisa estão bastante optimizadas. A informação gráfica, no entanto, não é tão facilmente pesquisável ou visualizável. Tal deve-se a algumas limitações ainda existentes na actual tecnologia *Web*, no que respeita à apresentação de imagens 2D, não animada ou vídeo, que consistem, por exemplo, no facto de a maior parte do material gráfico ser apresentado num dos três formatos de varrimento seguintes: GIF, JPEG, PNG, os quais, para além de distorcerem qualquer factor de escala diferente de 1:1, não são tão facilmente pesquisáveis [Probets,01].

O SVG é uma tecnologia XML que veio contornar esta situação: para além de permitir a representação de informação gráfica em formato compacto, genérico e portátil [Eisenberg, 02], a sua natureza textual (implícita

à sua representação XML) facilita a pesquisa da informação representada [Probets,01].

Assim sendo, a biblioteca implementada - VGLib2D - visa a criação de documentos e imagens SVG de forma apropriada à representação e visualização de grandes volumes de dados em 2D, uma vez que se considerou que o SVG seria, por estas e outras razões explicitadas mais adiante, o formato mais adequado para a representação de informação gráfica com forte componente vectorial ou representada por meio de vectores.

Nas restantes secções deste artigo faz-se uma breve referência às tecnologias usadas, abordando-se de seguida a tecnologia SVG – características e potencialidades - e alguns aspectos técnicos no que respeita à implementação da biblioteca. Finalmente apresentar-se-á um contexto de aplicação da biblioteca implementada e far-se-á referência a trabalho futuro.

2. TECNOLOGIAS UTILIZADAS

Como plataforma de desenvolvimento optou-se pela plataforma .Net [Microsoft, 03], um conjunto de tecnologias que pretende simplificar o desenvolvimento de aplicações *Web* e para plataforma Microsoft Windows [Duthie, 02]. Tecnicamente actuais, componenciais, e, ao basear-se num conjunto de *XML Web Services* (pequenas aplicações reusáveis escritas em XML, uma linguagem dirigida para a unificação da representação e circulação de informação), estas tecnologias revelaram-se altamente integráveis.

Neste âmbito, a biblioteca de classes foi implementada em C#, uma linguagem de programação da Microsoft®, relativamente recente, desenhada especialmente para a plataforma .Net. Baseada no paradigma de programação para objectos, a linguagem C# é derivada do C e do C++,

apresentando grande semelhança com o C++ e a Java. Para além disso, o C# baseia-se numa vasta biblioteca de classes presente no .Net Framework, promovendo a robustez, consistência e eficiência do *software*. [Drayton, 02]

Apesar da plataforma .Net não ter *namespaces* (conjunto de classes específicas) para trabalhar o SVG, foi possível recorrer ao suporte disponível para XML, o qual garante a criação de potencialmente qualquer tipo de imagens usando SVG, a partir de qualquer tipo de informação disponível [Wahlin, 02].

Toda a programação foi efectuada recorrendo ao *Visual Studio .Net* da Microsoft®.

3. A TECNOLOGIA SVG

3.1. Introdução

Uma das mais excitantes tecnologias XML actuais é o *Scalable Vector Graphics* (SVG), que, fornecendo uma sintaxe baseada em XML, pode ser usada para a criação e visualização de imagens, desde figuras a *reports* e gráficos, numa variedade de dispositivos [Wahlin, 02].

O SVG permite o uso de elementos e atributos específicos, definidos no *SVG Document Type Definition* (DTD), publicado no site da *World Wide Web Consortium* - W3C ([W3C, 00a]), para a criação de imagens. Apesar de haver uma vasta lista de elementos e atributos a aprender, é relativamente fácil trabalhar com SVG, e editores visuais de companhias tais como a Adobe® e a JASC podem exportar imagens em formato SVG [Wahlin, 02]. Para a visualização num *browser* é necessário um *plug-in* próprio, sendo o mais difundido o Adobe® SVG Viewer ([Adobe, 01]).

No texto 1, e correspondente figura 1, é apresentado um exemplo dos elementos básicos usados para desenhar formas e texto, criando uma imagem simples [Wahlin, 02]:

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C// DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-
20010904/DTD/svg10.dtd">
<svg width="300" height="150">
<rect x="31" y="36" width="240" height="59"
style="fill:#02027a;stroke:#000000;stroke-
width:1"/>
<text x="56px" y="67px" style="fill:#ffffff;
font-family:Arial; font-size:18"> My First SVG
Document </text>
</svg>
```

Texto 1: Exemplo de um documento SVG



My First SVG Document

Figura 1 : Visualização gráfica do documento usando o *Adobe SVG Viewer*

3.2. Principais características do SVG

Há quem afirme que a forma de representação gráfica na Internet está a ser revolucionada pelo formato SVG. O sucesso deste formato deve-se a algumas das suas características [Probets, 01] [Eisenberg, 02] [Mouza, 02] [Marriott, 02], as quais foram decisivas para a sua eleição no que respeita à implementação da VGLib2D, nomeadamente:

- **Formato vectorial;**
- **Sintaxe XML;**
- **Ficheiro de texto**, permitindo que motores de busca indexem ficheiros SVG ou componentes SVG embebidos em ficheiros HTML (*HyperText Markup Language*);
- **Independente de plataforma e dispositivo;**
- **Permite interacção**, ao possuir um número de procedimentos orientados ao evento, que permitem um certo grau de interacção com o utilizador;
- **Integração com CSS (*Cascade Style Sheets*) e XSL(*Extensible Stylesheet Language*);**
- **Flexibilidade**, no que respeita aos atributos dos elementos, permitindo uma separação entre a apresentação gráfica e a estrutura geométrica dos elementos;
- **Interface DOM (*Document Object Model*)**, graças à qual se pode modificar o documento do lado do servidor. Quando isto acontece o *browser* SVG transmite de imediato a alteração para o *display* gráfico;
- **Permite animação.**

Apesar de tudo o SVG apresenta também algumas limitações que é preciso ter em conta:

- **Não suporta adaptação do lado do cliente de acordo com diferentes condições de visualização.** Esta é uma desvantagem importante se tivermos em consideração que o uso de diferentes tipos de dispositivos é cada vez mais frequente, indo desde pequenos ecrãs de PDAs (*Personal Data Assistants*) e telefones móveis até aos grandes dispositivos de salas de conferências;
- **Formato textual do ficheiro pode, em alguns casos, resultar em ficheiros de grandes dimensões.** Tal problema pode ser ultrapassado desde que se tenham em conta alguns cuidados a nível da implementação que visam garantir a optimização dos ficheiros. Este aspecto será abordado mais adiante.

3.3. Áreas aplicacionais do SVG

O impacto do SVG está a ser de tal forma marcante, aos mais variados níveis, que são várias, e muito díspares, as áreas de conhecimento a recorrerem a esta nova tecnologia para efeitos de representação e visualização gráfica de dados.

Referenciam-se aqui apenas quatro exemplos, seleccionados entre os muitos disponíveis, que se consideram significativos e ilustradores da flexibilidade de aplicação deste formato, bem como do seu sucesso e difusão a nível da computação gráfica.

Pretende-se com esta abordagem fazer uma aproximação ao estado da arte da tecnologia SVG, bem como apresentar de forma mais concreta, recorrendo a exemplos reais, as potencialidades deste formato.

3.3.1. SVG e cartografia

Uma das áreas na qual o impacto do formato SVG está a ser mais marcante é a da cartografia. De facto a cartografia representa uma aplicação perfeita para o SVG: os mapas são, por definição, representações vectoriais de áreas 2D, distribuídas por camadas. A especificação SVG contém esse mesmo conceito de camadas ou agrupamentos, que é essencial para a representação de informação usando os Sistemas de Informação Geográfica (SIG).

Os mapas GIF ou JPEG usados até muito recentemente, apesar de não apresentarem problemas de compatibilidade com as ferramentas *Web* e ambientes de trabalho, não oferecem potencialidades de interacção, *zoom*, controlo de *layers*, entre outras. O SVG veio resolver todos esses problemas: não sendo um substituto para um Sistema de Informação Geográfica completo, o SVG personifica uma nova forma de representar informação geográfica de qualidade na *Web* [Seff, 02].

Em [Custom, 03] pode ser encontrado um exemplo de aplicação (ver figura 2):

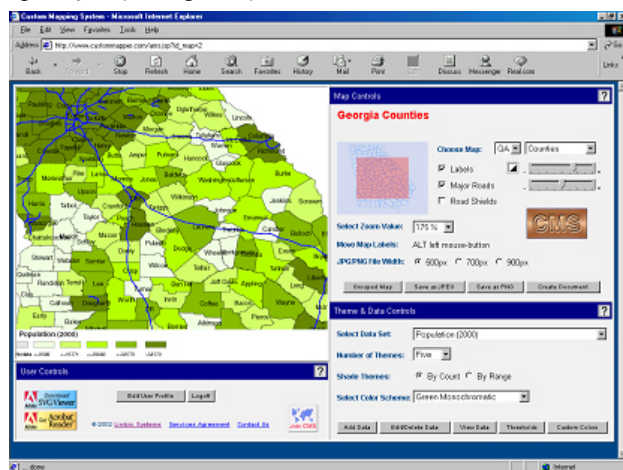


Figura 2 : Custom Mapping System v1.1, desenvolvido pela Limbic Systems, Inc

3.3.2. Chemical Markup Language (CML™)

A *Chemical Markup Language (CML™)* [CML, 03] é uma nova aproximação para a gestão de informação molecular que tira partido das tecnologias XML. Sendo capaz de conter estruturas de informação extremamente complexas, actua desta forma como um mecanismo de intercâmbio ou de arquivo.

Em [Adobe, 03] pode encontrar-se o seguinte exemplo, no qual informação armazenada em CML™ é

manipulada via uma XSL (*Extensible Stylesheet Language*) de forma a criar um SVG passível de visualização e interacção (ver figura 3):

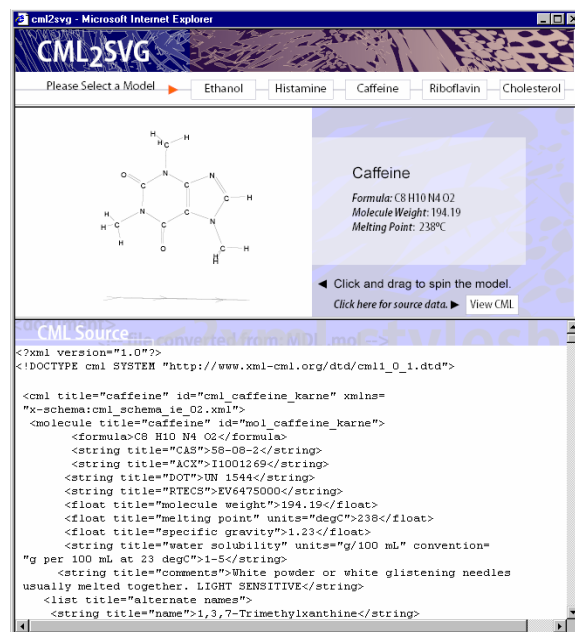


Figura 3 : Visualização 3D de uma molécula de cafeína em formato SVG, obtida a partir do formato CML™

3.3.3. SVG e Dispositivos Móveis

Dispositivos móveis tais como os PDAs, *PalmPilots*, *SmartPhones*, entre outros, ajudam na gestão e optimização de sequências de actividades a realizar pelos seus utilizadores.

Neste âmbito, recorreu-se ao formato SVG para a representação e visualização de toda a informação gráfica associada a essas actividades e tarefas, cuja representação fiel compreende a interacção de uma série de técnicas de visualização tais como a representação espacial do local onde ocorrem as actividades, representação temporal das tarefas, representação textual das mesmas através de listas, texto contínuo e hierarquias, representação de prioridades através do uso de cores, entre outros [Bieber, 03].

3.3.4. Bloco de notas de laboratório

O último exemplo apresentado será talvez o mais surpreendente no que respeita à demonstração da aplicabilidade do formato SVG.

Trata-se de um novo conceito, o “*augmented laboratory notebook*”: um espaço de trabalho que pretende fazer a ligação entre informação em formato papel e digital, em contexto laboratorial.

Sem se pretender aprofundar os aspectos técnicos inerentes aos protótipos implementados, importa referir que cada bloco de notas consiste num conjunto de folhas e que cada folha contém várias camadas distintas de informação. O SVG foi o formato seleccionado para toda a representação e posterior visualização da informação gráfica inerente ao uso deste dispositivo, que os autores acreditam poder vir a atingir as proporções de um

autêntico conjunto de ferramentas de realidade aumentada para trabalho sobre papel [Mackay, 02] (ver figura 4).

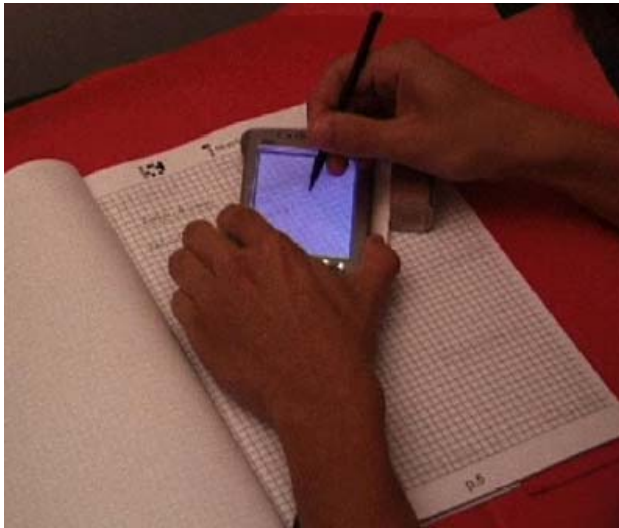


Figura 4 : Um dos protótipos implementados, o *a-book* [Mackay, 02].

4. A BIBLIOTECA

4.1. Introdução

Como já foi anteriormente referido, a plataforma .Net não contém *namespaces* para trabalhar o SVG. Esta é, de momento, uma limitação inerente, também, às restantes plataformas e linguagens de desenvolvimento. Tal significa que toda a implementação respeitante à geração de ficheiros em formato SVG é ainda trabalhada a muito baixo nível, o que é de certa forma preocupante, se se tiver em conta o estado da arte deste formato, bem como a quantidade de aplicações actualmente desenvolvidas com suporte SVG.

Foi precisamente com o intuito de colmatar esta lacuna que se optou pela implementação da VGLib2D: uma biblioteca de classes que permite a criação e escrita de ficheiros SVG. Mais do que isso, a VGLib2D pretende tirar partido das potencialidades do SVG, atrás citadas, de forma a possibilitar a representação gráfica, o *download* e impressão de grandes volumes de dados, com a rapidez e qualidade desejadas.

A VGLib2D compreende classes de mais baixo nível para a criação e definição de propriedades de elementos básicos do SVG, tais como *SVGCircle*, para a criação e definição de um círculo, *SVGLine*, para a criação e definição de uma linha, etc., bem como classes de mais alto nível, que funcionam como *interface* para a criação de documentos SVG: escrita de cabeçalho (*header*), escrita de elementos, de comentários, etc., e que se baseiam nas classes de mais baixo nível referidas (ver figura 5). Pretende-se com esta divisão a separação visível do núcleo relativo ao SVG propriamente dito (elementos e atributos SVG), da *interface* da biblioteca, acessível ao programador e dirigida para a criação e gestão de documentos que contêm a definição de todas as entidades gráficas a representar.

A VGLib2D foi implementada para funcionar de forma equiparada aos *namespaces* do .Net Framework e garantindo a interoperabilidade e comunicação com os mesmos.

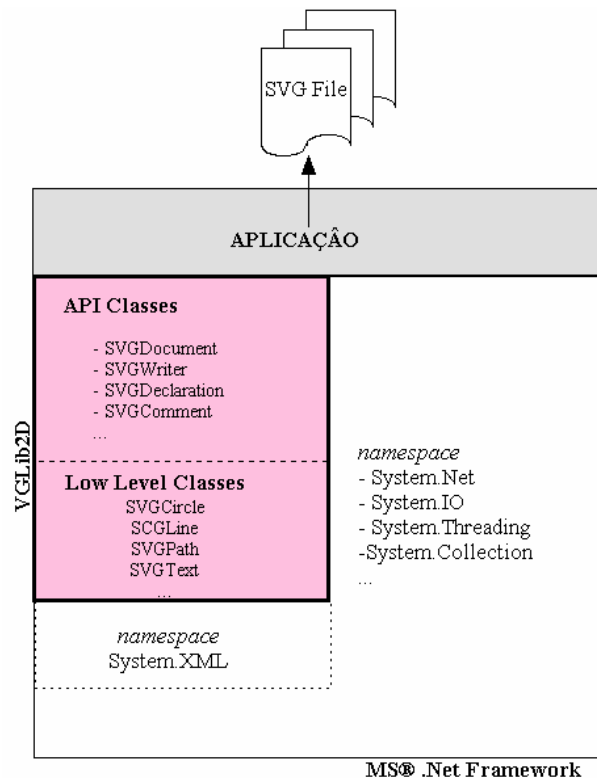


Figura 5 : VGLib2D – Arquitectura e Comunicação com o .Net Framework

4.2. Estratégia de implementação

Para efeito de implementação partiu-se do *namespace* System.Xml presente no .Net Framework Software Development Kit, e incluído na biblioteca de classes FCL (*Framework Class Library*) da Microsoft®.

Este *namespace* fornece suporte para a gestão de documentos XML, de acordo com um conjunto de standards definidos pelo *World Wide Web Consortium* (3WC). As suas classes implementam objectos que respeitam a especificação XML1.0([W3C, 98]) e o DOM Core Level 1 e Core Level 2 ([W3C, 00b]) [Drayton, 02]. Pretendeu-se com isto garantir a validação do formato XML dos ficheiros e a criação de um DOM coerente e estruturado que permita edições posteriores, com vista à obtenção de interactividade, entre outros.

Para validação dos ficheiros SVG criados utilizou-se o software XMLSpy®[XMLSpy, 03].

Uma vez que um ficheiro SVG mais não é do que uma variação de um ficheiro XML, e tendo em conta que partimos do *namespace* System.Xml, considerou-se que a arquitectura da VGLib2D deveria reflectir, até certo ponto, a arquitectura do *namespace* referido. Assim sendo, as classes de *interface* tais como *SVGDocument*, *SVGDeclaration*, *SVGWriter*, etc.... apresentam os mesmos métodos que se podem encontrar nas classes paralelas pertencentes ao *namespace* System.Xml,

acrescidas de outros métodos específicos dirigidos ao formato SVG e suas particularidades. É o caso de métodos tais como *SetWidth*, *SetHeight*, *SetUnits*, *DrawLine*, *DrawCircle*, entre muitos outros. Pretende-se assim estender a filosofia de utilização inerente aos *namespaces* do *.Net Framework* à *VGLib2D*, assegurando desta forma a uniformidade da *interface*: um programador experiente na plataforma *.Net* não deverá ter quaisquer dificuldades no uso da biblioteca implementada.

A biblioteca implementada visou conseguir o máximo grau de abstracção e uma hierarquia consistente, de forma a garantir a sua generalização a aplicações diversificadas.

Para a criação dos ficheiros SVG existem algumas questões de optimização que se devem ter em conta, tais como o uso preferencial de alguns atributos, como sejam *path* em substituição de múltiplas utilizações do atributo *line* ou *polyline*, o *H* e o *V* para linhas horizontais e verticais, entre outros. Considerações semelhantes aplicam-se a curvas [Proberts,01]. Após algum debate, decidiu-se que caberia ao utilizador a gestão das questões de optimização dos ficheiros, através da selecção correcta dos métodos referentes aos diferentes atributos a utilizar. No entanto a *VGLib2D* possui os mecanismos que proporcionam essa optimização.

4.3. Características

A *VGLib2D* possibilita a criação de documentos SVG com as mais variadas dimensões, definidas em todas as unidades métricas consideradas válidas pela especificação deste formato.

Para além de permitir a associação de folhas de estilo e a definição de estilos gerais para o documento, estilos estes automaticamente adoptados pelos elementos criados que os referenciem, a *VGLib2D* permite também a definição de elementos gráficos com o seu estilo próprio, independente dos estilos gerais do documento.

Todos os elementos ou entidades que fazem parte da especificação do formato SVG são considerados na *VGLib2D*. Sobre estes elementos é possível a definição e edição dos seus atributos e a aplicação de operações geométricas tais como translações, rotações e escalamentos.

Sem se pretender entrar em grande detalhe, exemplos de elementos suportados e respectivos atributos são o caso das linhas, com atributos como a espessura, cor, opacidade e estilo (contínuo, interrompido,...); do texto, com atributos como o tipo, tamanho, espessura e estilo de letra, alinhamento, espaçamento entre letras e palavras, orientação (vertical, horizontal, inclinada segundo determinado ângulo, com leitura da direita para a esquerda, de baixo para cima, etc.); entre tantos outros cuja enumeração exaustiva não é viável no contexto deste artigo.

A *interface* para a definição dos elementos gráficos é intuitiva e devidamente documentada, obedecendo todas as funções a uma sintaxe simples e evidente, como é o caso do seguinte exemplo, que permite a definição de uma recta/linha contínua:

- *DrawLine (SVGWriter filePointer, string id, double x1, double y1, double x2, double y2, int lineWidth, string color);*

Neste caso temos que *filePointer* é o ponteiro para o ficheiro SVG criado, *id* é o identificador da linha criada, *x1*, *y1*, *x2* e *y2* são as coordenadas da linha e *lineWidth* e *color* são, respectivamente, a espessura e a cor da linha.

Caso se pretendesse aplicar, na criação da linha, um estilo pré definido, invocaríamos antes a seguinte função, em que *style* é o identificador desse estilo:

- *DrawLine (SVGWriter filePointer, string id, double x1, double y1, double x2, double y2, string style);*

Finalmente a *VGLib2D* possibilita ainda a inserção de elementos em formato de varrimento, a definição de padrões e gradientes, de curvas de Bézier, e o uso de filtros e a definição de áreas geométricas com ou sem preenchimento.

Até ao momento não foram ainda contempladas classes ou funcionalidades destinadas a animação ou interacção com o utilizador em tempo real. A biblioteca foi, no entanto, architectada de forma a possibilitar esta implementação futura.

4.4. Contextos de aplicação

A *VGLib2D* permite o desenvolvimento de todo e qualquer tipo de aplicações que considerem o recurso ao SVG para a visualização de informação, desde que esta seja passível de representação/"tradução" gráfica 2D. Tal significa que pode ser utilizada no desenvolvimento de aplicações SVG em todas as áreas de aplicação atrás referidas, com a virtude de transportar, desta forma, toda a programação referente à geração de documentos SVG para um mais alto nível, que se optimizará progressivamente à medida que novas funcionalidades/classes forem implementadas.

A *VGLib2D* foi testada e refinada quando da sua utilização no desenvolvimento de uma aplicação *Web* para a gestão de Horários Técnicos de Comboios para a Refer.E.P., na qual se revelou crucial a nível da implementação da componente de *output* gráfico.

Um dos componentes requeridos para a aplicação implicava a visualização de Gráficos de Circulação, mapas técnicos onde aparece representada a circulação de um dado número de comboios ao longo de um intervalo horário e de uma sequência de estações. Estes mapas devem respeitar os horários em vigor, o período de validade dos comboios (período temporal de circulação destes comboios) e o seu regime de frequência (dias da semana em que circulam), entre muitos outros aspectos. Foi especificamente nesta componente que se aplicou a *VGLib2D*, com o objectivo concreto da geração dos Gráficos de Circulação em formato SVG.

Esta aplicação, acessível ao utilizador através de um simples *Web browser*, apresenta uma arquitectura Cliente-Servidor baseada em tecnologia Asp.Net e numa base de dados relacional SQL (*Structured Query*

Language), como se pode observar, de forma esquemática, na figura 6.

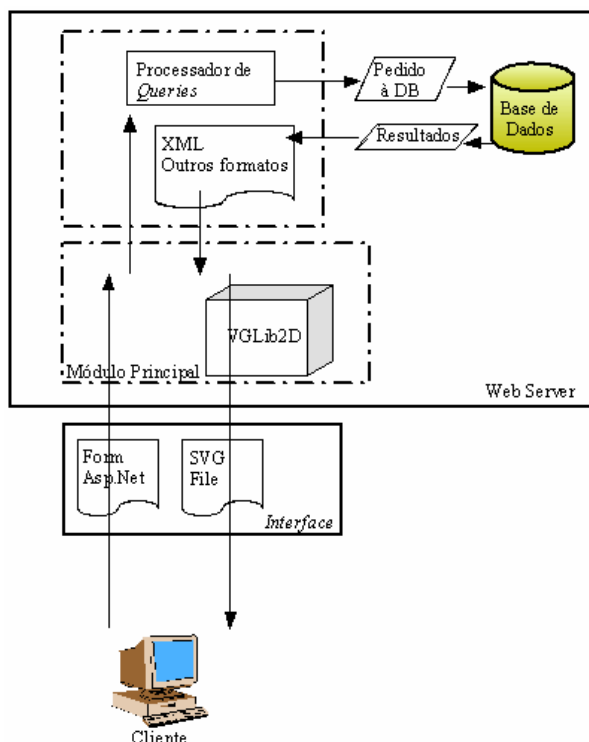


Figura 6 : Arquitectura do módulo referente aos Gráficos de Circulação da aplicação

A interface com o utilizador consiste assim num Formulário de Pesquisa (*form* ASP.NET) onde devem ser seleccionadas as linhas e estações que se pretendam consultar, intervalo horário, período de validade dos comboios e regime de frequência (ver figura 7).

Figura 7 : Formulário de pesquisa

A aplicação acede de seguida à base de dados através de *Data Queries* correspondentes aos dados introduzidos pelo utilizador, devolvendo resultados correspondentes aos comboios que circulam nos intervalos definidos pelo utilizador. Cada comboio é definido por um conjunto de informação genérica (primeiro número de identificação, segundo número de identificação, data de início de

circulação, data de final de circulação, data de inserção do comboio na base de dados, descrição, identificador operacional, identificador comercial, etc....), e um conjunto de nós ou estações correspondentes ao percurso efectuado (número de identificação da estação, número de identificação da linha, hora de chegada, hora de partida, tempo de paragem, etc....). Tanto a informação genérica como a relativa aos nós é bastante complexa, podendo o número de comboios ascender aos dois mil, nos casos em que o intervalo de pesquisa é extenso (por volta de um ano).

Toda a informação resultante é submetida a pré processamento, sendo depois “traduzida” e representada graficamente, de forma dinâmica, através do recurso à VGLib2D, ficando o ficheiro SVG resultante automaticamente disponível para visualização, impressão e *download*.

Um comboio, na sua representação gráfica, mais não é do que uma linha que percorre determinadas estações (eixo vertical) em determinado período temporal (eixo horizontal). Desta forma, recorreu-se a funções da VGLib2D para a criação de *polylines* e *paths*, para representar as trajetórias dos comboios ao longo das várias estações. Objectos adicionais de texto foram usados para as legendas, uma vez que todos os comboios têm de ter associada a informação genérica mencionada atrás. O uso de cores é também obrigatório na definição dos comboios, uma vez que esta funciona como elemento classificativo de diferentes tipos de comboios.

Convém salientar, no entanto, que o gráfico apresentado recorre apenas a uma parcela muito reduzida de elementos do SVG e usufrui de uma margem reduzidíssima das suas potencialidades.

Os Gráficos de Circulação, para além da visualização, estão especialmente orientados para a impressão. Este era, efectivamente, um dos requisitos primordiais a cumprir, uma vez que, devido à complexidade e ao elevado montante de informação muitas vezes contido em cada gráfico, a sua visualização e análise no ecrã mostrava-se insatisfatória. Foi assim preciso garantir a possibilidade de criação de ficheiros SVG de grandes dimensões, que podem ir desde o formato A4 (14,8cmx21cm) até ao A0 (84,1cmx118,9cm), passíveis de impressão em plotters específicas.

Apesar de ter sido possível a criação de ficheiros SVG com estas dimensões, a sua impressão só foi possível até ao formato A2(42cmx59,4cm) devido a limitações do *plug-in*, que obrigaram a que esta impressão fosse efectuada via *browser*. Prevê-se que esta limitação seja ultrapassada com a nova versão do *plug-in*, que promete disponibilizar funcionalidades de impressão.

Factores relevantes para o uso da VGLib2D foram o facto da aplicação desejada ser uma aplicação Web e de ter sido indicada como plataforma de desenvolvimento a plataforma .Net.

Houve, no entanto, uma outra intenção na escolha do SVG como formato a adoptar para a representação gráfica da informação: por um lado, com o SVG, e

recorrendo à utilização de grupos, é possível permitir ao utilizador opções de visualização interactiva da

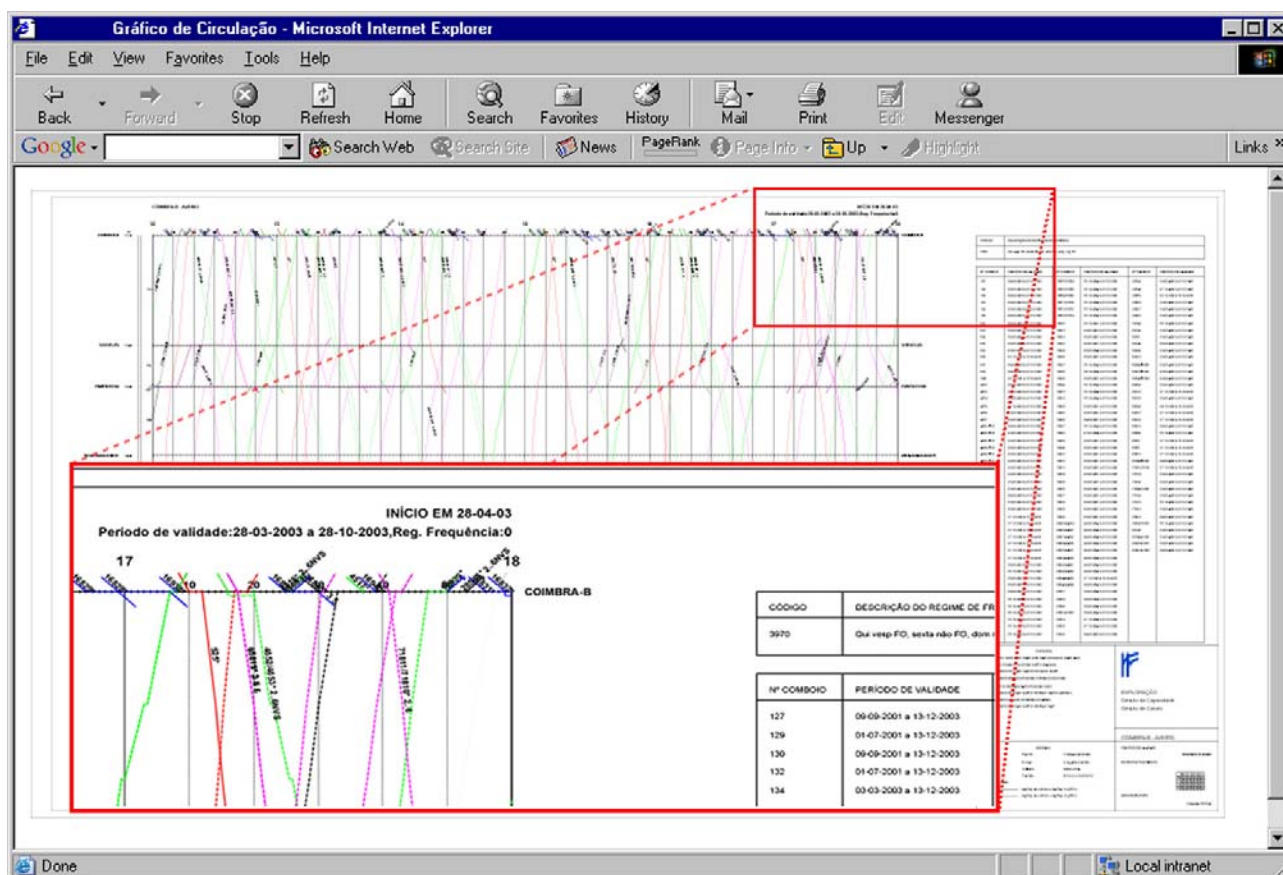


Figura 8 : Exemplo de um Gráfico de Circulação

informação tais como ocultar, evidenciar ou visualizar apenas determinados subgrupos de informação, do total que está contido no ficheiro SVG. Este tipo de funcionalidade pode ser equiparado ao uso de *layers*, técnica de visualização amplamente difundida em todo o tipo de *software* que trabalha com informação gráfica, e que permite a repartição da informação geométrica disponível em camadas isoladas, facilitando a sua leitura e interpretação. Por outro lado, ao possibilitar animação, o formato SVG permite que se venha a implementar, num futuro próximo, Gráficos de Circulação dinâmicos, em que o utilizador observa em tempo real a circulação dos comboios. Esta aproximação recai numa abordagem recente em que a noção tradicional de documentos como sendo maioritariamente compostos por entidades estáticas se tem vindo a modificar em direcção a uma nova perspectiva em que se envolvem constantemente artefactos informativos. São os chamados *Live Documents* : com as tecnologias emergentes, como é o caso do SVG, podem-se enriquecer os documentos informativos com componentes multimédia que possibilitam o acesso a informação adicional, interactividade e trabalho cooperativo [Weber, 02].

Um outro aspecto que levou ao uso da VGLib2D foi o facto de se estar a trabalhar com grandes volumes de informação e respectiva visualização gráfica: como já

foi referido, alguns Gráficos de Circulação chegam a apresentar informação referente a cerca de dois mil comboios. Tal significa ficheiros com um DOM bastante complexo e sobre os quais é necessário incidir todo um conjunto de medidas de optimização atrás referidas. Assim mesmo, em termos comparativos e a título de exemplo, importa salientar que o documento da figura 8, em dimensões consideradas legíveis (em mapas tão complexos e abarcando intervalos horários não inferiores a 12 horas, considera-se que um formato de papel inferior ao A2 não é legível) e com qualidade de impressão mínima (150dpi), em ficheiros de formatos de varrimento JPEG e TIFF apresenta, respectivamente, 1674KB e 41769KB, enquanto que em formato SVG apresenta 210KB. A escolha do SVG como o formato de *output* da VGLib2D mostrou ser, também neste aspecto, crucial.

De referir ainda que os ficheiros SVG permitem compactação no formato SVGZ recorrendo à ferramenta Gzip [Gzip, 03], mantendo-se passíveis de visualização no *plug-in*.

5. CONCLUSÃO E TRABALHO FUTURO

A VGLib2D revelou-se de grande utilidade para a implementação de aplicações baseadas em SVG e orientadas para a representação gráfica 2D de grandes volumes de informação.

Veio também colmatar a lacuna presente no *.Net Framework*, no que respeita à disponibilização de uma *interface* de mais alto nível para a geração de documentos SVG. O contexto de aplicação apresentado veio reforçar esta ideia, bem como evidenciar, mais uma vez, a grande flexibilidade do formato SVG e, consequentemente, cimentar a certeza de que este era o formato de *output* mais indicado a adoptar pela VGLib2D.

No que respeita a trabalho futuro, um dos aspectos de maior importância a considerar é o de dotar a VGLib2D de funcionalidades que suportem uma extensão ao SVG denominada de *Constraint Scalable Vector Graphics* (CSVG), que permite maior flexibilidade na descrição dos elementos gráficos. Com o CSVG, uma imagem pode conter objectos cujas posições e outras propriedades são especificadas em relação a outros objectos e não em termos absolutos. O CSVG permite retardar o *layout* dos objectos definidos no documento até ser efectuado o *rendering* pela aplicação cliente, o que resulta numa maior flexibilidade quanto ao uso de diferentes dispositivos de visualização [Badros, 01]. Ultrapassar-se-á, desta forma, um dos principais inconvenientes do SVG.

A mais longo prazo prevê-se a implementação de novas classes destinadas às funcionalidades de animação e interacção em tempo real.

6. AGRADECIMENTOS

Uma referência especial deve ser feita à Rede Ferroviária Nacional - REFER EP, à Meticube, Lda, e à noLimits consulting, S.A., com as quais o Centro de Computação Gráfica colaborou, implementando entre outras componentes a VGLib2D, para o desenvolvimento da plataforma de gestão de informação de material circulante actualmente em uso na REFER EP.

7. REFERÊNCIAS

- [Adobe, 01] Adobe® SVG Viewer 3, 2001. URL: <http://www.adobe.com/svg>
- [Adobe, 03] CML2SVG. URL: <http://www.adobe.com/svg/demos/cml2svg/html/index.html>
- [Bieber, 03] Gerald Bieber, Volker Leck. Scalable Vector Graphics for SaiMotion. *Computer Graphics Topics – Reports on Computer Graphics, 1/2003 Vol.15.* ~ http://www.inigraphics.net/publications/topics/2003/issue1/1_03a08.pdf
- [Badros, 01] Greg J. Badros, Jojada J. Tirtowidjojo, Kim Marriott, Bernd Meyer, Will Portnoy, Alan Borning. A constraint extension to scalable vector graphics. *Proceedings of the tenth international conference on World Wide Web*. April 2001, 489-498.
- [CML, 03] Chemical Markup Language (CML™). URL: <http://www.xml-cml.org/>
- [Custom, 03] Custom Mapping System V1.1. Limbic Systems, Inc. URL: <http://www.custommapper.com/index.jsp>
- [Drayton, 02] Peter Drayton, Ben Albahari, Ted Neward. C# in a nutshell – A desktop quick reference. O'Reilly & Associates, Inc. 2002.
- [Duthie, 02] G. Andrew Duthie, Matthew MacDonald. Asp.Net in a nutshell – A desktop quick reference. O'Reilly & Associates, Inc. 2002.
- [Eisenberg, 02] J. David Eisenberg. SVG Essentials. O'Reilly & Associates, Inc. 2002.
- [Gzip, 03] The gzip home page. <http://www.gzip.org/>
- [Mackay, 02] Wendy E. Mackay, Guillaume Pothier, Catherine Letondal, Kaare Bøegh, Hans Erik Sørensen. Interaction in the real world: The missing link: augmenting biology laboratory notebooks. *Proceedings of the 15th annual ACM symposium on User interface software and technology*. October 2002, 41-50.
- [Marriott, 02] Kim Marriott, Bernd Meyer, Laurent Tardif. XML Applications: Fast and efficient client-side adaptivity for SVG. *Proceedings of the eleventh international conference on World Wide Web*. May 2002, 496-507.
- [Microsoft, 03] Microsoft® .Net™ <http://www.microsoft.com/net/>
- [Mouza, 02] Cédric du Mouza, Philippe Rigaux. GIS and the internet: Web architectures for scalable moving object servers. *Proceedings of the tenth ACM international symposium on Advances in geographic information systems*. November 2002, 17-22.
- [Proberts, 01] Steve Proberts, Julius Mong, David Evans, David Brailsford. Hypermedia and Graphics 2: Vector graphics: from PostScript and Flash to SVG. *Proceedings of the 2001 ACM Symposium on Document engineering*. November 2001, 135-143.
- [Seff, 02] George Seff. SVG and GIS. 2002. URL: http://www.directionsmag.com/article.php?article_id=198
- [W3C, 00a] World Wide Web Consortium. DTD - W3C SVG 1.0 Specification - Candidate Recommendation 20001102. URL: <http://www.w3.org/TR/SVG/svgdtd.html>
- [W3C, 00b] World Wide Web Consortium. The Document Object Model. 2000 URL: <http://www.w3.org/DOM>
- [W3C, 98] World Wide Web Consortium. Extensible Markup Language (XML) 1.0 – W3C Recommendation 10 Feb 98. URL: <http://www.w3.org/TR/1998/REC-xml-19980210>
- [Wahlin, 02] Dan Wahlin. SVG: .Net Graphics Courtesy of XML. *XML & Web Services Magazine*. August/September, 2002.
- [WebDraw, 03] Jasc® WebDraw™. URL: <http://www.jasc.com/products/webdraw/>
- [Weber, 02] Anke Weber, Holger M. Kienle, Hausi A. Müller. Live documents with contextual, data-driven information components. *Proceedings of the 20th annual international conference on Computer documentation*. October 2002, 236-247.
- [XMLSpy, 03] XMLSpy <http://www.xmlspy.com/>

Vídeo Digital para Multimédia:

Boas Práticas de Aquisição e Processamento

Pedro Faria Lopes

ADETTI / ISCTE

Av. Forças Armadas, 1600 Lisboa

Pedro.Lopes@iscte.pt

Maria Vasconcelos Moreira

ADETTI

Av. Forças Armadas, 1600 Lisboa

Maria.Moreira@iscte.pt

Noel Santos

ADETTI

Av. Forças Armadas, 1600 Lisboa

Noel.Santos@iscte.pt

Sumário

Este artigo descreve as particularidades do vídeo digital em multimédia pretendendo contribuir para uma compilação de boas práticas na aquisição, processamento, codificação e integração de vídeo digital de qualidade para multimédia. Este trabalho resulta de investigação e desenvolvimento levado a cabo no contexto do Projecto EcoMedia.

Palavras-chave

Multimédia, vídeo digital, aquisição, processamento, enquadramento, imagens por segundo, IPS.

1. INTRODUÇÃO

No que se refere à utilização de vídeo digital em multimédia, 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, não há um evidente valor acrescentado na inclusão de vídeo clips.

Na realidade a sua utilização é realizada mais como um *gadget*: utiliza-se porque o vídeo digital em multimédia existe e tecnicamente é exequível, porque ter o vídeo é “mais giro” ou “tem mais piada”. Porquê? Mesmo em enciclopédias, com tendência para a “universalidade”, não é evidente a razão da utilização de determinados vídeo clips, qual a razão da sua escolha, em detrimento de outros ou de, simplesmente, utilizar texto e imagem.

Na maior parte dos casos e aplicações, fazem-se transcrições de conteúdos que na realidade são originalmente provenientes da indústria do cinema ou vídeo, recodificados num formato digital para ser incluído em multimédia, conteúdos que não foram pensados, de origem, para o produto multimédia final.

Em termos técnicos, para reduzir o débito binário por causa de limitações de largura de banda nos dispositivos de armazenamento, ou para incluir mais informação no mesmo espaço de armazenamento, são manipulados (normalmente reduzidos) a resolução da imagem, o número de imagens por segundo, a frequência de amostragem do som, o número de bits por amostra, e outros parâmetros, para garantir que uma dada limitação ou especificação técnica é cumprida. Na maior parte dos casos não

se respeita – ou não existe – uma coerência de conteúdo, quando deveria ser o conteúdo, e a melhor forma de o disponibilizar, que deveriam ser os motores.

Como consequência das pequenas resoluções de imagem, dos baixos ritmos de imagem por segundo, etc., o resultado final é pobre, a percepção do conteúdo da imagem e do movimento são, nos melhores casos, sofríveis.

Da análise salvaguardam-se algumas excepções, entre elas por exemplo o jogo *Eve* [Gabriel96] onde é bastante claro que a base tecnológica é posta ao serviço do conteúdo.

O que nos preocupa na criação e desenvolvimento de conteúdos multimédia maioritariamente, ou integralmente, baseados em vídeo digital é saber como é que se pode e deve planificar, adquirir, processar, codificar e integrar conteúdo de vídeo digital especificamente para multimédia.

Este artigo resulta do trabalho desenvolvido no contexto do projecto *EcoMedia*, projecto desenvolvido entre Janeiro de 2001 e Dezembro de 2002, co-financiado pelo programa Sapiens99 da Fundação para a Ciência e Tecnologia, projecto nº POSI/1999/CHS/34676.

O projecto *EcoMedia* baseia-se na utilização, reutilização e actualização rápida e barata de material em vídeo digital de qualidade para a integração num ambiente de jogo multimédia. Este artigo estabelece um conjunto de recomendações e procedimentos para a melhor utilização e optimização de vídeo e áudio digitais para aplicação em multimédia, em particular para aplicação no âmbito do ambiente de jogo produzido no Projecto *EcoMedia*. Algumas das práticas e ilações são apresentadas no contexto do projecto, sendo no entanto facilmente generalizáveis e

aplicáveis noutros contextos de vídeo digital para multimédia.

2. IMAGENS E MOVIMENTO

Num painel de discussão na conferência *International Conference on Media Futures*, Florença, Itália, de 8 a 9 de Maio, 2001, um dos intervenientes solicitava “material novo, não recodificação de material antigo”. Esta observação foi feita a um nível técnico e tecnológico, no contexto de formatos novos que suportam novas possibilidades que é necessário disponibilizar com conteúdo novo sob pena de se tornar obsoleto antes de ser utilizado.

Mas esta observação pode e deve ser lida de outro modo, no contexto do estabelecimento de um paralelo, por estudo, dedução e comparação da criação e evolução das indústrias do cinema e do vídeo. Explicam-se seguidamente alguns resultados desta análise comparativa.

Cinema, vídeo e vídeo para multimédia, aparentemente tratam ou são, qualquer um deles, “imagens em movimento”. Esta leitura faz esquecer a especificidade de cada meio, que se passa a explicar.

Tomando o exemplo de filmes como *2001, Uma Odisseia no Espaço* ou *Lawrence da Arábia*, ver estes filmes numa sala de cinema leva a uma certa leitura do conteúdo: porque a dimensão, o espaço fisicamente ocupado pela imagem condiciona a percepção do conteúdo; ver as mesmas imagens num ecrã de cinema ou num tubo de raios catódicos é diferente: no ecrã de cinema ficamos com a noção da dimensão do deserto e do espaço, na televisão essa noção perde-se facilmente ou não chega a existir. Se, inversamente, uma novela – um produto gerado para televisão – fosse projectada numa sala de cinema, rapidamente os espectadores ficavam cansados do conteúdo, eventualmente sem perceberem exactamente a causa.

Estas questões da não permutabilidade, ou o grau de permutabilidade, de formatos/conteúdos entre cinema e televisão, a diferença entre um filme (cinema) e um telefilme (televisão) ou a especificidade de publicidade para televisão, são questões conhecidas e tratadas na literatura dedicada às respectivas áreas. Se, por exemplo, analisarmos um livro da área da televisão [Compesi90] verifica-se que parte do conteúdo é dedicado à técnica e tecnologia inerentes ao formato. Mas tem também a análise e descrição de como tirar o melhor partido do formato, como obter determinados tipos de conteúdo, como realizar enquadramentos, etc., ie, como gerar conteúdo especificamente para televisão, não para cinema. Devido à menor dimensão física da imagem reproduzida em televisão deve-se, por exemplo, planificar o conteúdo à base de planos mais próximos do assunto, em enquadramentos de “grande plano” ou “plano médio”. É isto que explica porque é que num telefilme ou nas telenovelas se vêem tantas imagens à base de rostos em vez de planos mais abertos. A linguagem da televisão e do cinema são diferentes, do cinema a linguagem teve que se adaptar à especificidade da televisão.

Se analisarmos material relativo a vídeo digital em multimédia temos basicamente uma abordagem tecnológica

com a descrição dos detalhes de codificação, compressão, ..., [Williams99] sem menção à forma de criar conteúdo que tire partido do novo meio. Em vídeo digital para multimédia o tamanho físico da imagem é diferente do tamanho da imagem em televisão e em cinema pelo que, mais uma vez, a percepção da imagem se vai alterar.

Por outro lado, se em televisão e em cinema o ritmo de imagens por segundo é basicamente o mesmo (25 e 24 ips), para a definição de conteúdo de vídeo para multimédia é importante perceber o contexto da mudança do ritmo de imagens por segundo já que não é indiferente que o mesmo conteúdo – movimento – seja reproduzido a 10, 12, 15 ou 25 imagens por segundo.

O paralelo está criado e é neste contexto que o trabalho que desenvolvemos pretende contribuir para a clarificação e definição de metodologias de boas práticas de aquisição, processamento e integração de vídeo digital para multimédia, contribuindo em última instância para a criação de uma linguagem de comunicação multimédia.

3. REQUISITOS DE IMAGEM E MOVIMENTO

Num trabalho anterior [Lopes99] vimos como a percepção da qualidade do movimento depende da velocidade de reprodução, tal como se exemplifica na diferença entre a sequência original a 25 ips (Figura 1) e a sequência recalculada a 10 ips (Figura 2). No segundo caso deixa de haver percepção do movimento da rolha a saltar.

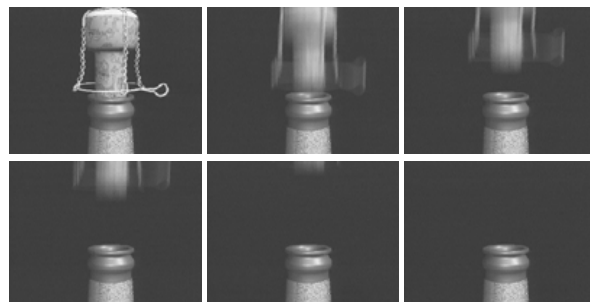


Figura 1. Movimento original a 25 IPS [Lopes90].

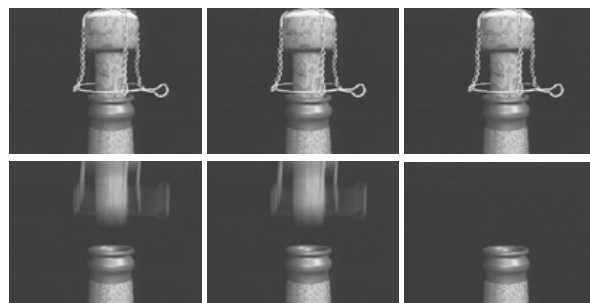


Figura 2. Reprodução de movimento a 10 IPS.

Este exemplo mostra que sequências rápidas só são reproduzidas fielmente quando se garante uma elevada taxa de imagens por segundo.

Ao estudarmos esta questão aplicada a pessoas em movimento, por exemplo marcha, constata-se que qualquer velocidade de reprodução que não seja 25 ips, mesmo em marcha lenta, já deturpa a percepção: o movimento é percebido como caricatural. Se o efeito for a reprodução

natural do movimento, como se pretende no projecto EcoMedia, então deve manter-se a reprodução a 25 ips e deve investir-se em soluções e tecnologia que permita garantir esta velocidade.

Analisando a abordagem seguida no jogo *Eve* [Gabriel96] verifica-se que o conteúdo e a concepção do jogo foram pensados de forma integrada de modo a minimizar os efeitos visíveis das limitações técnicas à data existentes em vídeo digital para multimédia (imagem pequena e ritmo baixo de reprodução).

O jogo inclui algumas abordagens técnicas convencionais para a integração de imagem real sobre um cenário virtual, como por exemplo filmando um actor sobre um fundo azul e aplicando técnicas de *chroma key* ou processamento de canal *alpha* (transparência) para criação de diferentes níveis (*layers*) onde são visíveis os problemas típicos de nível de detalhe dos limites da imagem do actor e onde se percebem algumas questões no tipo de movimento conseguido (mistura de “caminhar” com translação de um plano independente).

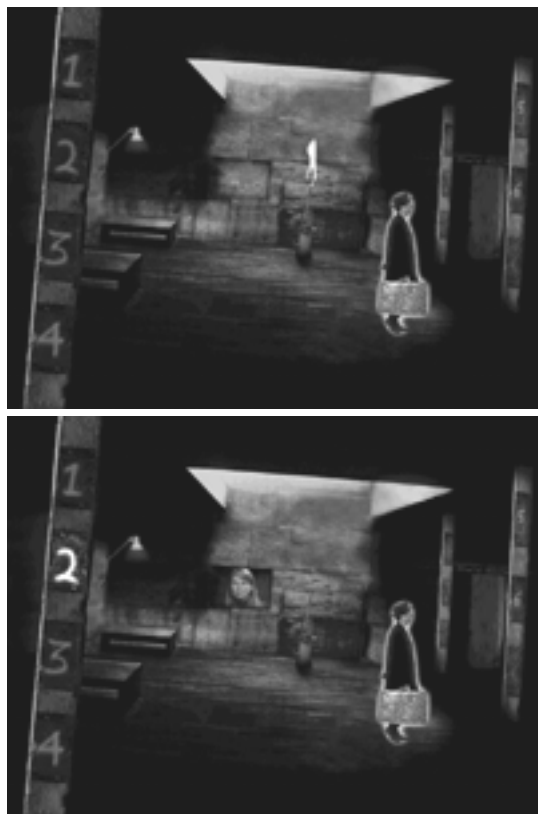


Figura 3. Vídeo, antes e depois de seleccionar um ecrã.

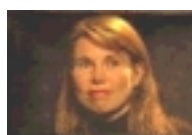


Figura 4. Vídeo seleccionado na Figura 3, 70x50 pixel.

Numa outra parte do jogo vemos uma sala interior onde o jogador pode seleccionar um de vários painéis numerados: com esta selecção desencadeia-se um vídeo clip numa parede ao fundo (Figura 3).

Vemos a pessoa a falar na parede distante e o movimento parece sem falhas. Há várias razões para isto:

- Quando retirada do cenário (Figura 4) a imagem aparece sem detalhe, mas a integração no cenário faz a diferença relativamente à sua percepção;
- O movimento lento da fala, os lábios em movimento, é menos problemático a ritmos baixos: com uma análise cuidada podemos detectar que a taxa de imagens por segundo é de 10 ou 12 ips;
- Todos os vídeo clips consistem numa cara mostrada ao longe numa parede, não vemos meio-corpo ou corpo-inteiro, o enquadramento e captura foram feitos especificamente para este contexto;
- Cada vídeo é curto em duração: foi aplicada contenção de tempo porque, com uma imagem pequena, a atenção do espectador não ficaria presa muito tempo.

Este tipo de estratégias não podem ser aplicadas ao conteúdo de um projecto multimédia onde todo o conteúdo é de carácter vídeo e preenchendo o ecrã disponível: para o *standard de facto* de ecrãs de jogo a resolução de imagem tem que ser 640x480 pixels, ou superior. O conteúdo desta imagem é suposto ser integralmente preenchido com sequências de vídeo onde os movimentos apresentados são, praticamente todos, de pessoas a andar.

Ao desenvolvermos o projecto EcoMedia fomos então confrontados com um paradoxo aparente: se um projecto multimédia se baseia integralmente em vídeo e se, de entre os tipos de informação multimédia utilizáveis (texto, imagem, som, gráficos, animação, vídeo), o vídeo digital tem sido o “parente pobre”, como proceder para gerar, com qualidade e de forma o mais económica possível – em termos de custos directos, custos em tempo de desenvolvimento, custos de processamento e custos de integração – os conteúdos em vídeo digital?

Nas próximas secções apresentam-se as bases para a captura, aquisição, tratamento e integração de vídeo digital nas condições exigidas pelo projecto.

4. VÍDEO E ÁUDIO DIGITAL

O sistema de vídeo digital mais barato e mais expandido neste momento é o sistema DV, *Digital Video*. Inicialmente previsto para o mercado doméstico, o formato DV¹ foi rapidamente adoptado pelo mercado profissional devido à sua qualidade e baixo custo. Para maximizar as margens de lucro no mercado profissional, disponível para pagar mais caro que o mercado doméstico ou institucional, e para oferecer maior qualidade de codificação, nasceram as variantes DVCam (Sony) e DVCPro (Panasonic). Para o mercado doméstico foi criado o formato miniDV, uma variante compacta do formato DV numa cassette mais pequena.

¹ Inicialmente o formato foi introduzido com a designação DVC, *Digital Video Cassette*.

A electrónica da Camcorder², na própria câmara, comprime o vídeo numa relação de 5:1 gerando imagens de grande qualidade, mesmo por comparação com os melhores sistemas analógicos da gama institucional (em formato YC – processamento separado da luminância, Y, e crominância, C, nas variantes de sistema Hi8³ ou S-VHS⁴). Se os elementos da componente óptica da câmara forem de boa qualidade (lentes Zeiss, por exemplo) e forem associados a um sistema de aquisição de imagem de 3 CCDs⁵ para a captura separada dos canais de cor RGB⁶, o resultado final é de muito elevada qualidade.

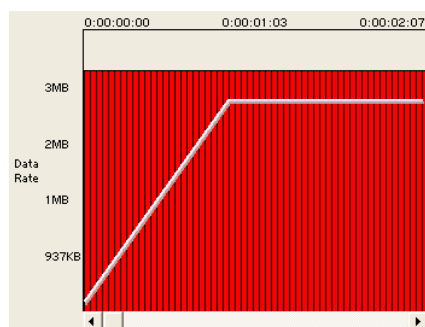


Figura 5. Débito binário de AVI codificado com CODEC DVSD.

Apesar da taxa de compressão ser de 5:1, o débito binário gerado na captura de vídeo PAL, 25 imagens por segundo e resolução de imagem de 720x576 pixels, é muito elevado: a partir do primeiro segundo (frame 25) o débito binário é de quase 3 MB por segundo e tem que ser mantido (regime *sustained*) para garantir que não há degradação da qualidade de imagem e do movimento (Figura 5). Isto implica que a transcrição do vídeo para um formato digital editável e processável em computador necessita de uma ligação de alto débito. Normalmente essa ligação é feita recorrendo à norma IEEE-1394⁷ que permite uma taxa de transferência de 400 Mbs suficientes para a transcrição sem perdas de imagens e sem interromper a fluidez do movimento do vídeo devida à baixa taxa de transfe-

rência ou baixa velocidade de escrita do disco. É comum hoje os computadores portáteis virem já equipados no *motherboard* com uma ligação FireWire e em todas as plataformas, *laptops* ou computadores de mesa, os discos são hoje já de grande capacidade de armazenamento e de alto débito binário tanto em escrita como em leitura.

Para os computadores que ainda não possuam porta FireWire, estão disponíveis placas em PCI ou PCMCIA com preços entre 45 € e 190 € tornando muito barato o acesso à digitalização em tempo real de vídeo de qualidade. Praticamente todas as câmaras de vídeo digital DV possuem portas FireWire.

Há uma variante particular de câmaras, formato Digital8 (ou D8), que são em tudo semelhantes do ponto de vista funcional e na forma como gravam digitalmente a informação excepto no formato da cassette que é o formato Hi8. Numa cassette de 90 mn Hi8 uma câmara Digital8 grava 60 mn dado que passa a fita magnética a maior velocidade. Estas câmaras mantêm a compatibilidade com os sistemas analógicos Video8 e Hi8 permitindo ler as antigas cassetes.

O som gravado no formato DV é sempre digital, codificado em PCM. Há câmaras de vídeo que permitem optar entre 2 ou 4 pistas de som, ie, 1 ou 2 canais estéreo. Esta funcionalidade permite que se façam montagens sonoras, pós-sonorização, na própria câmara depois da captura. No caso do projecto EcoMedia todo o processamento é realizado digitalmente em computador onde, por *software*, se tem a possibilidade de criar até 99 pistas de som estéreo (198 pistas mono). Dado que as 4 pistas de som na câmara são obtidas reduzindo a qualidade global disponível (frequência de amostragem de 32 KHz e 12 bits por amostra), recomenda-se a utilização de apenas 1 canal estéreo (2 pistas de som) para garantir maior qualidade na captura do som original (48 KHz de frequência de amostragem e 16 bits por amostra).

5. AMBIENTE DE TRABALHO E PARAMETRIZAÇÕES

Depois da aquisição do material com a câmara de vídeo transcreve-se o material para computador, num formato digital de ficheiro vídeo digital AVI.

É possível utilizar o formato MOV (*QuickTime*). O formato AVI é originário do ambiente Windows da Microsoft enquanto o formato MOV é originário do ambiente MacIntosh da Apple. Se hoje a interoperacionalidade de formatos de ficheiros e sistemas operativos é grande, por comparação com os tempos iniciais em que os ficheiros MOV só corriam em ambiente Apple e os ficheiros AVI só corriam em ambiente Microsoft, ainda assim a prática demonstra que cada um dos formatos é mais estável, e menos sujeito a erros, quando integrado e utilizado no respectivo ambiente original. Assim sendo, e admitindo que não são necessárias funcionalidades especiais suportadas por um dos formatos, recomenda-se que no ambiente da Microsoft se utilize ficheiros AVI, e MOV no ambiente da Apple.

² Camcorder: designa o equipamento que é simultaneamente câmara e gravador (**Camera recorder**) por oposição aos sistemas (profissionais) de câmara e gravador independentes.

³ Hi8: sistema de gravação analógico em cassette que grava em separado a luminância (Y) e a crominância (C), variante de melhor qualidade do sistema de gravação Video8 da SONY que grava em vídeo composto.

⁴ S-VHS: Super-VHS, sistema de gravação analógico em cassette que grava em separado a luminância (Y) e a crominância (C), variante de melhor qualidade do sistema de gravação VHS (*Video Home System*) proposto pela Philips e JVC.

⁵ CCD: *Charged Coupled Device*, matriz de sensores de estado sólido que convertem a luz incidente em sinais eléctricos; câmaras com 3 CCDs captam e codificam em separado os 3 canais de cor RGB obtendo-se elevada qualidade de imagem.

⁶ RGB: *Red, Green, Blue*, os 3 canais para codificação de cor.

⁷ Tecnologia originalmente criada pela Apple, com o nome de FireWire, normalizada com a designação de IEEE-1394; à sua implementação a Sony deu o nome de i.LINK.

A transcrição do conteúdo da cassette DV (em formato miniDV ou Digital8) captado pela câmara digital é feita recorrendo à ligação IEEE-1394 entre a câmara e o computador, conforme referido na secção 4. Esta transcrição é realizada recorrendo ao programa Adobe Premiere⁸.

Na inicialização do programa, quando é utilizado pela primeira vez, é necessário fazer uma opção de modo de utilização entre dois ambientes de trabalho de edição de vídeo digital: o modo **A/B editing** ou o modo **Single-Track Editing**. Para utilizadores menos experientes recomenda-se a utilização **A/B editing** dado ser a mais intuitiva e fácil para iniciados ou utilização menos intensiva (Figura 6).

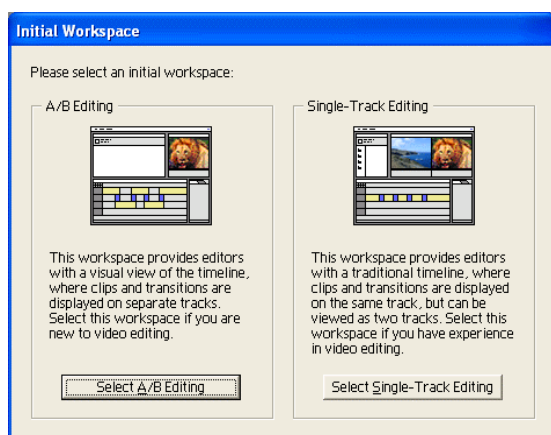


Figura 6. Inicialização do Premiere 6.0.

A etapa seguinte consiste em criar um projecto de trabalho. Para isso é necessário explicitar os parâmetros do projecto. Recomenda-se, pelas razões referidas no início desta secção, a utilização de ficheiros AVI, relativamente ao qual é necessário seleccionar as características.

Estando inseridos na Europa, onde o sistema de vídeo utilizado é o sistema PAL, e utilizando o formato DV para registo, os parâmetros de projecto Premiere a seleccionar são **DV-PAL**. Como se viu na secção 4 o tipo de som utilizado deve ser de 1 canal Estéreo o que implica que a frequência de amostragem é de 48 KHz. Pelo que, dentro da opção **DV-PAL**, se deve optar por **Standard 48 KHz** (Figura 7).

A diferença entre **Standard** e **Widescreen** tem a ver com a possibilidade oferecida por algumas câmaras de programar o formato da janela de enquadramento da imagem entre formato normal (**Standard**) e formato alargado (**Widescreen**). No formato normal tem-se uma imagem com relação de aspecto de 4:3, o formato de imagem da televisão convencional, enquanto no formato alargado se tem uma relação de aspecto de 16:9, o formato de imagem da televisão dita HDTV (*High Definition TV*). No entanto, no contexto de vídeo PAL, esta diferença, ou

variante, é enganadora na medida em que ao formato alargado não corresponde um aumento real de resolução de imagem e, como tal, de real melhoria de qualidade na percepção da imagem: a imagem aparece num enquadramento “ao baixo”, característico do enquadramento 16:9, mas com a mesma resolução de imagem de 720x576 pixel do enquadramento normal. Isto significa que se trata, só e apenas, de uma forma diferente de apresentar a mesma quantidade disponível de elementos de imagem, ie, pixels: procede-se a distorção na captura e no processamento digital em computador, e aplica-se uma distorção inversa quando de novo se transcreve para cassette vídeo. Para ter o benefício completo do formato alargado (**Widescreen**) com 16:9 de resolução de aspecto, a resolução de imagem deveria incluir a resolução HDTV de 1920x1080 pixel, o que só é possível em equipamento profissional de vídeo ou cinema digital topo de gama, com preços na ordem das várias centenas de milhar de Euro. Na Figura 8 mostra-se os dois tipos de enquadramentos referidos.

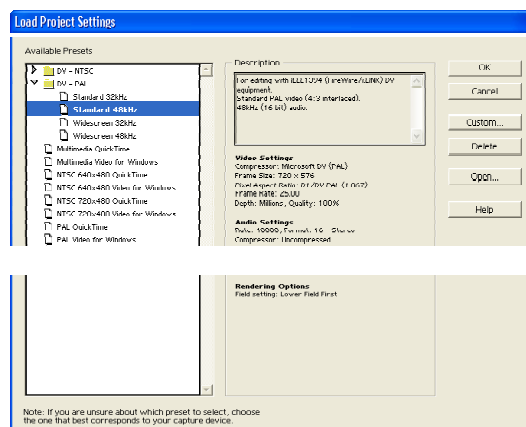


Figura 7. Parâmetros de projecto no Premiere 6.0.

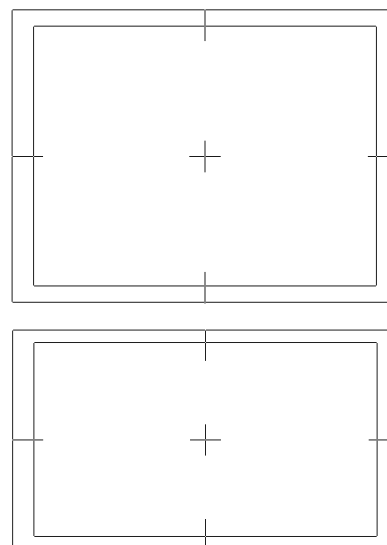


Figura 8. Enquadramentos de janela vídeo, a 4:3 (cima) e a 16:9 (baixo).

Uma vez que nos interessa apenas a captura para ambiente digital de computador e a utilização do vídeo no con-

⁸ A versão presentemente utilizada é a 6.0. Na aquisição de uma placa FireWire este programa é geralmente associado à placa, em regime de venda *bundle*, seja em versão integral ou numa versão “leve” (LE, *Light Edition*).

texto de um jogo, a opção correcta é maximizar a área de visualização para a resolução disponível utilizando o enquadramento normal, a 4/3, o formato *standard de facto* de imagem para jogos de computador.

A transcrição do conteúdo da cassete para computador é feito sob controlo do programa Adobe Premiere que controla a câmara e gera, em tempo real, um ficheiro AVI em formato DV. Aqui a expressão “formato DV” é usada no contexto dos ficheiros digitais em ambiente de computador dado que, em rigor, o formato DV é um formato de gravação digital em cassete.

As características do ficheiro AVI criado são:

- Resolução de imagem: 720x576 pixel
- *Frame rate*: 25 ips (ou fps, *frames per second*);
- CODEC: codec DV, por defeito é o DVSD da Microsoft;
- Som: estéreo, 48 KHz de frequência de amostragem, 16 bit, sem compressão;

Com estas características o espaço ocupado em disco, apesar da taxa de compressão de 5:1 do formato DV, é substancial: 1 minuto de vídeo em formato DV ocupa aproximadamente 220 MB. Para se ter uma melhor percepção refira-se em complemento que um ficheiro com 18 minutos ocupa praticamente 4 GB. Estes valores têm implicações ao nível da gestão do que se regista (filma) e do que se deve transcrever para disco do computador, questões que são tratadas na próxima secção.

6. PROCESSAMENTO E CODIFICAÇÃO

Filmar em cassete de vídeo digital tem um baixo custo, seja por comparação com a captura em filme, seja em valor absoluto: uma cassete de 1 hora custa menos de 15 €. Sendo barato e sendo fácil é comum realizarem-se capturas de vídeo longas. Esta prática tem a ver com a falta de planificação e na falta de critérios de opção face a um tema e/ou projecto.

No contexto de um projecto como o EcoMedia deve planificar-se a captura de modo a conter o material captado numa dimensão razoável, i.e., gerível, para minimizar problemas nas fases seguintes. Para processar o vídeo captado é necessário transferi-lo primeiro para computador para um ficheiro AVI em formato DV. Este formato, mantendo a qualidade visual e de movimento do que foi gravado em cassete, ocupa 220 MB por cada minuto. Se o material captado tiver de duração 5 minutos, o espaço ocupado é praticamente 1 GB. Sem algum critério para minimizar a captura, muito rapidamente se gastam várias dezenas de giga bytes de disco só para guardar os ficheiros AVI a processar. Para lá do espaço ocupado há a questão de gerir, encontrar, ordenar ou re-processar, as sequências ou planos dentro do que se captou: não é indiferente tentar identificar uma sequência de 30 segundos que interessa para o projecto num ficheiro de 3 minutos ou noutra de 18 minutos. Quanto mais material se tiver captado em cassete maior é a probabilidade de depois se tornar complicado, lento e frustrante fazer a escolha final de planos e sequências para integrar num jogo.

A planificação das filmagens deve começar com o conhecimento da área que se vai utilizar (jardim, pátio, sala, ...) como ambiente para a captura. No local deve também ensaiar-se algumas tentativas de enquadramento, mesmo sem utilizar a câmara, de modo a ter-se uma percepção melhor do que poderá funcionar ou não no contexto do projecto. As opções e ensaios devem ser anotados e com isso deve listar-se os planos, conteúdo e acção, que se vai captar. Assim tem-se a certeza que se focou a atenção antes de começar a captura. Mesmo que nem todo o material seja utilizado no fim, minimizou-se a situação de começar a filmar sem uma ideia concreta pensada previamente.

Para o jogo final, e de uma forma geral para outros projectos multimédia, deve optar-se por integrar sequências de vídeo codificadas num formato *standard* que garanta boa qualidade de imagem e movimento, que seja suportada pelas diferentes plataformas (Win95, Win98, ME, 2000, XP) e que garanta uma taxa de compressão muito superior aos 220 MB por minuto do formato DV de modo a minimizar taxas de transferência e espaço ocupado em disco. Esta escolha recaiu sobre a codificação MPEG-1 pelas seguintes razões:

- Garante boa qualidade de imagem (superior a VHS),
- com um ritmo de imagens por segundo de 25 ips,
- é suportado por todas as plataformas,
- tem uma taxa de compressão de 1.4 Mbps (aprox. 10 MB para 1 minuto).

Já existe equipamento barato que permite captar directamente de uma fonte de vídeo para MPEG-1. No entanto a escolha exacta de uma sequência ou plano de vídeo (início e fim precisos função da acção) tem que ser feita digitalmente, em computador. Realizar esta operação directamente em formato MPEG-1 é lento uma vez que o formato não se destina à edição imagem a imagem. O formato DV cumpre esta função de “edição à frame”, facilitando muito esta tarefa. A opção funcionalmente mais correcta é trabalhar em formato DV e, no fim de escolhidos os planos, recodificar os diferentes planos (ficheiros AVI em DV) em formato MPEG-1. Para o efeito pode utilizar-se um codificador de MPEG-1 disponível em *freeware*. Num dos números da revista *Computer Video* (<http://www.computervideo.net>) era feita a análise de vários codificadores de MPEG-1 concluindo a análise que, por exemplo, o codificador TMPGEnc [Hori00], gratuito, tinha maior qualidade de codificação que a maioria dos codificadores comerciais.

7. ENQUADRAMENTO E ACÇÃO

Uma das maiores vantagens do projecto EcoMedia é a possibilidade de reutilização do material para produzir jogos novos. Esta vantagem foi estendida para lá da simples reutilização: com a mesma estrutura de jogo pode gerar-se conteúdos semelhantes onde os actores são diferentes.

Esta possibilidade permite, por exemplo num ambiente escolar, fazer com que os novos alunos que entram no

início do ano lectivo sejam envolvidos na actuação e na criação de conteúdos para o(s) jogo(s). O jogo pode ter uma estrutura idêntica e mudarem-se apenas os “actores” ou pode gerar-se um jogo completamente novo.

Mesmo na situação “mais simples”, ie, manter a estrutura de um jogo já existente, a componente apelativa de os alunos participarem e serem parte integrante do conteúdo de um jogo é uma mais valia atractiva de grande interesse por cativar a atenção dos alunos tanto no conteúdo do jogo como no próprio processo de criar o jogo.

Na Figura 9 está exemplificada a situação da mesma acção captada com um intervalo de 4 meses de diferença e com actores diferentes (Susana, 8 anos, Sofia, 16 anos).

Para a optimização desta abordagem de reutilização de conteúdos de jogo(s) há que ter em linha de conta alguns princípios orientadores que se explanam nesta secção. Estes princípios – apesar de deduzidos no contexto do projecto EcoMedia – são facilmente generalizáveis a todas as situações de planificação e captura de vídeo digital para multimédia.



Figura 9. O mesmo cenário, aquisição com 4 meses de intervalo.

No processo de capturar, de novo, acções idênticas no mesmo ambiente coloca-se a questão do tempo que entre tanto mediou entre as duas capturas. No exemplo mostrado fizeram-se as capturas à mesma hora do dia, com 4 meses de intervalo como se referiu. Se, na segunda captura, se fizer a captura à mesma hora do dia, no mesmo

local como cenário, serão as condições ambiente idênticas às existentes quando se realizou a captura original? O que é que pode ter mudado no ambiente, exceptuando o caso de obras ou outro tipo de intervenção no espaço público, que impeça a aquisição nas mesmas circunstâncias?

Podem dar-se uma série de mudanças, mais ou menos subtis, que podem implicar a necessidade de alguma forma de adaptação.

Em primeiro lugar as condições atmosféricas podem ser diferentes. Pode o céu estar mais sombrio ou mais luminoso. Dependendo do conteúdo, o factor atmosférico pode ter ou não relevância. Para um controlo integral das condições de luz a opção será a de fazer as capturas em ambiente controlado, ie, em ambiente interior em que é possível fazer a gestão das fontes de luz, eliminando o problema da variabilidade atmosférica.

No caso do exemplo apresentado, para uma diferença temporal de 4 meses, para o mesmo cenário, o mesmo enquadramento, a mesma direcção da câmara e a mesma hora do dia para a captura, o resultado foi o que se apresenta na Figura 10.



Figura 10. Efeito do sol no alinhamento óptico da câmara.

O efeito verificado deve-se à mudança da posição relativa entre a terra e o sol que implicou que o sol passasse a ficar praticamente no alinhamento óptico da câmara provocando encadeamento, *glare*, por efeito de luz directa a

entrar na óptica da câmara. Mesmo no caso de se utilizar uma protecção em pala sobre a câmara, é visível na imagem esquerda que há efeito de *glare*. Se a imagem estática, como apresentada, parece pertinente, na sequência vídeo em que a aluna se movimenta entre a câmara e o sol, esta interferência provoca variação do efeito de *glare* provocando um efeito de distração no conteúdo da imagem.

A solução para este tipo de problema é, mantendo a época do ano, escolher um momento do dia em que a direcção do sol não se apresenta de frente, como o obtido e apresentado (2ª imagem) na Figura 9.



Figura 11 Linha de acção do *background* para o *foreground*.



Figura 12 Linha de acção da direita para *foreground* com inflexão da trajectória.

A selecção do enquadramento – posição relativa da câmara face ao cenário e a quantidade de elementos a visualizar – deve ser realizada função do conteúdo da acção a captar. Tal como é costume discutir em disciplinas ditas clássicas (desenho, pintura, fotografia) o factor principal é a forma como os elementos na imagem são percebidos. Com o vídeo acrescenta-se um factor extra a ter em conta: a componente dinâmica e o tempo de duração da acção.

Na Figura 11 e na Figura 12 apresentam-se, para o mesmo enquadramento, duas possibilidades de acção. Na

Figura 11 a acção desenrolar-se-ia do fundo para a frente (de *background* para *foreground*). Devido à extensão do espaço, o tempo desta acção seria muito lento, com o problema adicional de se ter que resolver a forma como o elemento actuante entraria no cenário. Na Figura 12 a solução resolve a questão do tempo, que passa a ser mais curto, e o problema da entrada do elemento no cenário (faz-se pela direita) mas a trajectória sofre uma inflexão junto da papelreira onde a componente principal da acção acontece, a colocação de *lixo* na papelreira. Dependendo do conteúdo que se pretenda, pode manter-se esta opção, ou escolher um enquadramento que facilite uma acção mais linear.



Figura 13 Exemplo de elementos exógenos à acção.

Quando se procede à captura em espaços públicos há que ter em linha de conta a eventualidade de não se poder evitar a inclusão de elementos exógenos à acção. Na Figura 13 mostram-se, realçados em rectângulos, turistas que ficaram incluídos nos enquadramentos da acção do jogo. A forma de resolver esta situação pode assumir duas vertentes: solicitar às pessoas que, no tempo da aquisição, não ocupem o espaço da imagem, solução tanto mais difícil quanto maior for o espaço público; seleccionar a captura de modo a que, no tempo da acção, as pessoas se apresentam paradas e não representam uma fonte dinâmica que distraia o observador da acção principal. Nos exemplos mostrados foi esta segunda hipótese que se seleccionou.

8. CONCLUSÕES

Este artigo abordou a questão de gerar conteúdo em vídeo digital que tenha boas características de qualidade de imagem e reprodução de movimento. Nele se descrevem metodologias para obter conteúdos para as sequências vídeo a integrar no jogos criados com o Projecto EcoMedia, metodologias facilmente extensíveis a outros projectos multimédia.

Descreveu-se ainda os parâmetros e características técnicas de material e equipamento para concretização barata da abordagem proposta.

Com este trabalho pretendemos contribuir para uma melhoria da utilização e integração de vídeo digital de qualidade no contexto mais amplo de realização e produção de multimédia digital de qualidade.

9. AGRADECIMENTOS

À Susana Lopes e à Sofia Pinto pela participação no projecto *Eco-Media*. Ao José Sequeira Paiva pelo apoio e ajuda.

10. 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.
- [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.pcgaming.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.
- [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.
- [Lopes90] P.F. Lopes, *CPI'90*, video, 3D computer animation, 1' 40", Official opening of the 6th Portuguese Congress of Informatics, INESC, July 1990, (Auditório 1, Fund. C. Gulbenkian).
- [Lopes99] P.F. Lopes, M.V. Moreira, M.L. Duarte, "Eco-Vasco, 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, H. Pereira, "Jogos Multimédia Educativos: Estratégias de Desenvolvimento", 3º Simpósio Internacional de Informática Educativa, Viseu, Portugal, 26-28 de Setembro, 2001.
- [Lopes01d] P.F. Lopes, M.V. Moreira, H. Pereira, "Estratégias de Desenvolvimento de Jogos Multimédia", 10ºEPCG - 10º Encontro Português de Computação Gráfica, ISCTE, Lisboa, 1 a 3 de Outubro de 2001.
- [Lopes01e] P.F. Lopes, organizador, Painel Multimédia, "Multimédia e multidisciplinaridade: realidades (virtuais) emergentes", Membros do Painel: Amélia Muge, Rui Zink, Maria Ferrand e Pedro Faria Lopes, Painel moderado por Pedro Faria Lopes, ISCTE, Outubro de 2001; Painel Multimédia realizado no âmbito do 10º Encontro Português de Computação Gráfica e do Projecto EcoMedia, Gestão de Auditório e gravação áudio: Centro de Audiovisuais do ISCTE, Transcrição áudio-texto: Maria de Lourdes Camelo, Digitalização, processamento áudio: Pedro Faria Lopes, Coordenação: Pedro Faria Lopes e Maria Vasconcelos Moreira.
- [Lopes01f] P.F. Lopes, M. Moreira, Relatório Técnico, "Aquisição de sequências para o projecto EcoMedia", Projecto EcoMedia, ADETTI/ISCTE, 6 de Setembro de 2001.
- [Lopes02] P.F. Lopes, M.V. Moreira, et al., *EcoMedia*, jogo multimédia, projecto co-financiado pelo programa Sapiens99 da Fundação para a Ciência e Tecnologia.

logia, projecto nº POSI/1999/CHS/34676 – projecto desenvolvido entre Jan. 2001 e Dez. 2002.

[Lopes02c] P.F. Lopes, M. Moreira, J.S. Paiva, Relatório Técnico, “Boas práticas de aquisição e processamento de vídeo digital para multimédia”, Projecto EcoMedia, ADETTI/ISCTE, Dezembro de 2002.

[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).

Variação Dinâmica do Nível de Detalhe na Simulação Estável de Tecidos

Fernando Pedro Birra

Manuel Próspero dos Santos

Departamento de Informática
Faculdade de Ciências e Tecnologia da
Universidade Nova de Lisboa
Quinta da Torre, 2825-114 Caparica
{ fpb, ps }@di.fct.unl.pt

Sumário

A simulação de tecidos virtuais é um processo que requer um significativo poder computacional. De modo a manter uma velocidade de simulação aceitável sacrifica-se o detalhe. Neste artigo propomos uma técnica de gestão dinâmica do nível de discretização do tecido onde a qualidade das sucessivas malhas obtidas por subdivisão/simplificação é garantida pelo esquema de subdivisão escolhido. A topologia da malha por nós usada exibe algumas características únicas que a tornam atraente no contexto da variação gradual do nível de detalhe. A subdivisão é accionada segundo um critério geométrico, avaliado localmente, tendo como base a presumível curvatura do tecido. A simplificação é despoletada nas regiões do tecido onde a curvatura é suficientemente reduzida. A escolha do método de subdivisão não deverá causar perturbações bruscas no plano do tecido em direcções perpendiculares a este. No entanto, a redução do nível de detalhe produz inevitáveis alterações geométricas instantâneas que provocam descontinuidades ao nível das equações diferenciais usadas no modelo do tecido. Este artigo contribui, de forma original, com um conjunto de técnicas que permitem resolver os problemas de instabilidade numérica que advêm da simplificação da malha.

Palavras-chave

Sistemas de partículas, simulação, modelação de tecidos, nível de detalhe.

1. INTRODUÇÃO

Os sistemas de partículas são a ferramenta de modelação mais frequente na modelação de tecidos virtuais. O tecido é modelado como um conjunto de partículas com massa, sobre as quais actua um conjunto de forças. Essas forças tanto surgem como resultado de interacções internas ao tecido (curvatura, elasticidade e distorção) ou como acção externa: gravidade, interacção com o utilizador ou resposta a colisões. As forças internas do tecido são modeladas ligando conjuntos de partículas entre si. A evolução do sistema é efectuada resolvendo o sistema de equações diferenciais ordinárias cuja solução traduz o movimento de cada partícula do sistema.

A resolução do sistema de equações diferenciais é um processo que requer um significativo poder computacional, podendo ter que se avaliar o modelo do tecido várias vezes por cada passo de simulação.

Os métodos de subdivisão e simplificação adaptativas podem permitir a simulação realista de tecidos, com grande nível de detalhe, usando criteriosamente o poder computacional disponível. Uma vez que o detalhe é introduzido apenas onde é mais necessário, os resultados poderão ser equivalentes aos obtidos quando se utiliza uma malha muito detalhada, visto o detalhe nas zonas menos curvas ser perfeitamente dispensável.

Neste artigo propomos um método de animação baseado num modelo contínuo do tecido, acoplado com um esquema de subdivisão e simplificação adaptativas que preserva a qualidade da malha durante toda a simulação. O esquema de subdivisão utilizado é um dos que permite uma variação mais gradual no número de elementos (triângulos) e vértices da malha. Devido à sua simplicidade permite ainda uma implementação muito eficiente e compacta.

2. TRABALHO RELACIONADO

Alguns trabalhos iniciais na simulação de tecidos, recorrendo a sistemas de partículas [Breen92, Breen94], usavam técnicas de minimização da energia total do sistema para encontrar configurações estáveis para o drapejado dos tecidos. As funções de energia tentavam modelar a mecânica interna do tecido e eram definidas relacionando partículas de um determinado arranjo, entre si. Na primeira fase da simulação deixava-se cair o tecido e determinavam-se os pontos de contacto com os obstáculos. Na segunda fase minimizava-se a energia total do sistema corrigindo as posições obtidas na primeira fase.

Posteriormente, com vista a conseguir-se simular a dinâmica dos tecidos, assistiu-se à proliferação de modelos mecânicos que permitiam simular o movimento das partículas, recorrendo à resolução de

sistemas de equações diferenciais. Aqui podem-se categorizar os modelos mecânicos dos tecidos em contínuos ou discretos. Exemplos de modelos discretos incluem os conhecidos sistemas mola-massa [Provot95] ou sistemas baseados em equações de energia semelhantes às definidas em [Breen92], mas de onde se derivavam as equações diferenciais do movimento [Eberhardt96]. As técnicas de integração usadas eram, geralmente, baseadas em métodos de integração explícitos, tais como o método de Euler ou de Runge-Kutta.

Em relação aos modelos físicos onde o tecido é tratado com um contínuo encontramos exemplos orientados por preocupações distintas. Por um lado, o trabalho de Baraff e Witkin [Baraff98] tem como preocupação a estabilidade da simulação, usando para isso um método de integração implícita onde a precisão é sacrificada pela estabilidade. O comportamento do tecido é definido formulando um vector condição que se pretende que seja nulo. Sobre este vector condição são definidas funções de energia assim como, por derivação destas, as forças que actuam no sistema. Grande parte da originalidade do método advém de os autores terem conseguido resolver o sistema de equações resultante da técnica de integração, que é muito esparsa, usando um método que, simultaneamente, permite garantir que um conjunto de restrições sobre as partículas é satisfeito em cada passo. Por outro lado, o trabalho de Dias et al. [Dias00] tem como preocupação simular conhecidos comportamentos mecânicos dos tecidos, sendo uma das características deste trabalho a incorporação no modelo de dados experimentais obtidos por medições mecânicas directas.

Todos estes trabalhos usam malhas com uma divisão regular, quer seja sob a forma de quadriláteros ou sob a forma de triângulos.

A primeira tentativa de usar o refinamento de forma adaptativa, sobre uma malha rectilínea regular, foi efectuada por Hutchinson et al. [Hutchinson96]. O modelo massa-mola introduzido em [Provot95] foi usado na simulação. Cada vez que era detectado um ângulo, entre duas molas consecutivas que fosse superior a um determinado valor limiar, os 4 quadriláteros da vizinhança eram subdivididos, dando lugar a 16 no total. Os resultados relatados não eram muito satisfatórios e o tempo computacional era enorme. Por outro lado, quando o detalhe já não fosse necessário a malha não era simplificada.

Zhang et al. [Zhang01] inspiraram-se também no modelo de Provot e apresentaram um método de subdivisão regular. A malha inicial, é uma malha regular de triângulos com baixa resolução. O drapejado do tecido é simulado até a malha atingir um ponto de equilíbrio. Nessa altura a malha é refinada de forma regular. Cada triângulo produz quatro novos triângulos, mesmo onde o detalhe não é necessário. O processo repete-se durante diversos níveis de subdivisão. Apesar da ineficiência da subdivisão regular, os autores alegam uma poupança, em termos computacionais, da ordem de

2.5:1 quando em comparação directa com a malha refinada no maior nível de detalhe. O método, infelizmente, apenas pode ser usado para refinar o tecido no final da simulação.

Num trabalho recente, Villard et al. [Villard02] partiram também do modelo de mola-massa de Provot, mas corrigiram o termo responsável pelas forças de curvatura, formulado de forma errada no trabalho original de Provot. A malha, formada por uma grelha regular de partículas, é subdividida, durante a simulação, usando um critério local puramente geométrico. A curvatura é estimada num vértice da malha medindo o maior desvio das normais dos triângulos que partilham esse vértice, em relação à normal do vértice (calculada por soma de todas as normais dos triângulos circundantes). Cada passo de subdivisão divide cada um dos 4 quadriláteros à volta do vértice seleccionado em 4 novos elementos. Desta forma, cada passo multiplica o número de faces por 4 e o número de vértices, se ignorarmos o vértice que causou a subdivisão, é duplicado. Os novos vértices são ainda classificados em nós activos e nós virtuais. Esta divisão em dois conjuntos distintos de nós visa resolver o problema da subdivisão escolhida gerar malhas que não respeitam a topologia inicial. As forças aplicadas aos nós virtuais são distribuídas pelos nós activos da vizinhança. A resolução do sistema de equações que traduz a dinâmica do tecido é efectuada usando um passo de Euler explícito requerendo, por isso, passos de simulação muito pequenos. Infelizmente, a simplificação dinâmica da malha é relegada para trabalho futuro.

Outros autores [Howlett98, DeRose98, Etmuss00] usaram também técnicas onde novos vértices são introduzidos na malha, mas que não contribuem para o realismo da simulação. Esses novos vértices não são usados na simulação e são calculados de forma puramente geométrica.

Volkov e Ling [Volkov02] abordam, também, o problema do refinamento e simplificação adaptativos de malhas de tecidos. No seu trabalho são usadas malhas de triângulos e o sistema é simulado usando a técnica descrita em [Baraff98]. O sistema monitoriza a curvatura do tecido em cada aresta e decide proceder à subdivisão quando o erro da aproximação poligonal da malha é superior a um patamar escolhido. O processo inverso, da simplificação, é efectuada quando se considera que uma malha com menos elementos é suficiente. A técnica de subdivisão escolhida é a descrita em [Kobbelt00] e denomina-se por subdivisão $\sqrt{3}$. Infelizmente, os autores não explicam quais as vantagens do método de subdivisão escolhido, no contexto da sua aplicação específica, à simulação de tecidos. Os resultados apresentados pelos autores não fazem qualquer referência aos problemas de descontinuidade, e consequente instabilidade, que as operações de subdivisão e simplificação daquela técnica causam quando aplicadas a malhas representando tecidos.

Outras estratégias de refinamento são apresentadas em [Bridson02, Hadap99]. A subdivisão é totalmente desacoplada da simulação, sendo efectuada como um passo de pós-processamento depois de a simulação ter sido efectuada com uma malha menos detalhada. O objectivo é o de obter, recorrendo à interpolação, superfícies que, no final, se assemelham a tecidos reais..

3. SUBDIVISÃO HIERÁRQUICA DE MALHAS 4-8 REGULARES

As malhas por nós escolhidas denominam-se malhas 4-8 regulares¹ e são um caso particular das malhas 4- k [Velho00].

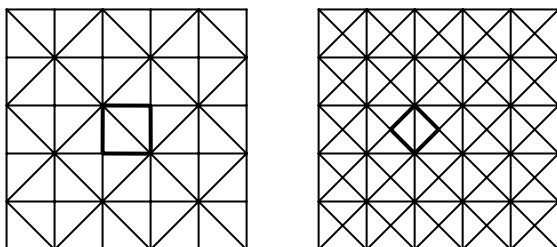


Figura 1 - Duas malhas 4-8 regulares

A topologia, duplamente ilustrada pela Figura 1, pode-se caracterizar da seguinte forma:

- A malha é formada por quadriláteros, sendo cada um destes dividido por uma diagonal dando origem a dois triângulos.
- O grau dos vértices interiores da malha é 4 ou 8.
- A vizinhança-1 de cada vértice de grau 4 é formada apenas por vértices de grau 8
- A vizinhança-1 de cada vértice de grau 8 forma um anel de vértices com graus alternados de 4 e 8.

Apesar de, à primeira vista, a topologia parecer demasiado limitativa, qualquer malha de triângulos pode ser previamente transformada numa malha regular 4-8 através de pré-processamento. Em [Velho01] é descrito um algoritmo para o efeito.

Na Figura 1, a malha do lado direito pode ser obtida subdividindo de forma uniforme a malha do lado esquerdo. É ainda visível, a traço grosso, o bloco fundamental (*prototile*) do esquema de subdivisão usado. Este é composto por um arranjo de dois triângulos partilhando uma aresta.

A operação elementar de refinamento, exemplificada na Figura 2, é a bissecção da aresta interior ao bloco, e é traduzida nos seguintes passos:

1. A aresta interna do bloco é dividida em duas, por inserção de um novo vértice v .
2. Cada uma das faces é dividida em duas, unindo o novo vértice v aos vértices opostos, w e e , das faces iniciais.

¹ O termo regular é aqui usado num sentido lato, uma vez que existe mais do que um tipo de grau ou valência dos vértices.

A vantagem da aplicação desta operação a uma malha conforme, ao contrário de outras operações tais como a divisão de um triângulo em 4 (*face split*), é que a malha resultante continua a ser conforme, ou seja, sem *cracks* (buracos).

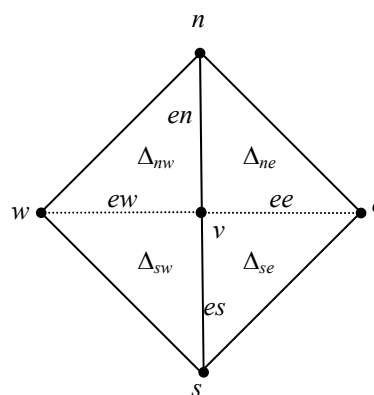


Figura 2 - Bissecção da aresta interna ao bloco

No lado esquerdo da Figura 3 mostra-se o resultado de uma subdivisão aplicada duas vezes, de forma uniforme, à malha representada pelo traço mais grosso. Os vértices a negro resultam do primeiro passo de refinamento e os cinzentos do segundo. O lado direito mostra a hierarquia que o processo de subdivisão vai criando nos triângulos subdivididos.

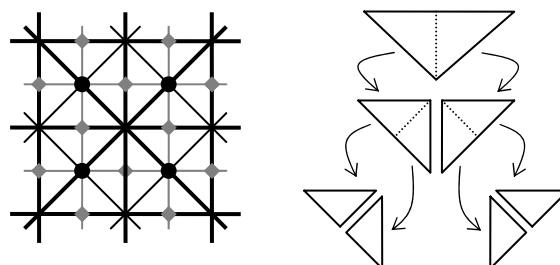


Figura 3 – Dois níveis de subdivisão e correspondente hierarquia para um elemento da malha original

4. VANTAGENS DA SUBDIVISÃO 4-8.

No contexto da aplicação à simulação de tecidos virtuais, o que mais interessa num esquema de subdivisão adaptativa é a capacidade de se refinar a malha e poder voltar-se a simplificá-la, sem introduzir discontinuidades nas equações que regem a simulação. Parece óbvia a inevitabilidade da simplificação produzir essas discontinuidades geométricas, que se transmitirão directamente à avaliação do modelo do tecido. No entanto, mesmo o processo de refinamento pode gerar tais problemas, como se verá de seguida.

Talvez o método de subdivisão mais comum, quando lidamos com malhas de triângulos, seja o método de subdivisão de 1-para-4 de Loop [Loop87]. Cada aresta é

dividida em duas e os novos vértices são ligados entre si, produzindo um triângulo central e três outros junto dos vértices iniciais. Este método, quando aplicado de forma não uniforme, tem o problema de gerar *cracks* na malha, que poderá ser resolvido, como se ilustra na Figura 4(a), dividindo em dois os triângulos circundantes.

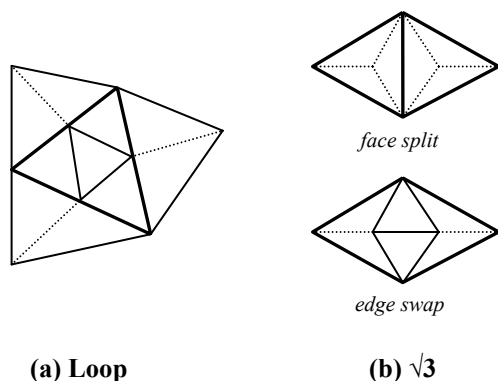


Figura 4 - Outros métodos de subdivisão

Uma desvantagem evidente é a necessidade de distinguir entre os triângulos regulares e os irregulares (separados entre si, na Figura 4(a), pelas linhas a pontilhado). Os triângulos regulares participam da hierarquia, podendo ser subdivididos mais tarde, o mesmo não acontecendo com os triângulos irregulares, que apenas existem para evitar o aparecimento de buracos na malha. Os triângulos irregulares, caso fossem subdivididos, fariam com que a malha degenerasse. Eles são, posteriormente, descartados, sendo substituídos por triângulos regulares quando a subdivisão for efectuada no triângulo (regular) que os contém.

No caso da subdivisão $\sqrt{3}$, para além de existirem igualmente dois tipos de triângulos – regulares e irregulares, a geometria também pode sofrer perturbações modificando, localmente, o plano da malha. Estas alterações ocorrem na etapa designada por *edge swap*, que visa a transformação de triângulos irregulares adjacentes em triângulos regulares, como se mostra na Figura 4(b).

No decorrer do nosso trabalho, descobrimos que a maior causa de instabilidade, introduzida pela utilização de um método de subdivisão adaptativa, resulta das perturbações instantâneas da malha que se traduzam em deslocamentos da geometria inicial em direcções normais ao plano do tecido. O fenómeno pode ser observado por um enrugamento (ou encarquilhar) artificial e instantâneo, como resultado da modificação da topologia da malha num determinado local.

Quando as alterações na malha produzem uma variação instantânea na orientação dos triângulos, o resultado é geralmente catastrófico. As forças estruturais que contrariam o alongamento do tecido sofrem, neste caso, uma mudança brusca, não apenas em módulo mas também, e essencialmente, na sua direcção. Ora estas mudanças não são acompanhadas pela evolução das

suas derivadas, usadas na integração numérica do sistema. Apenas referimos as forças de elasticidade porque são as de maior intensidade e as que causam a maior parte dos comportamentos caóticos nas simulações onde o processo de integração numérica entra em divergência.

Não podemos afirmar, taxativamente, que este fenómeno proíbe a utilização de esquemas de subdivisão onde a orientação dos triângulos pode variar bruscamente, no decorrer de uma operação elementar de refinamento ou subdivisão. Podemos, contudo, afirmar que, na ausência de medidas adicionais que resolvam ou atenuem o problema, as quais não vimos ainda referidas na literatura, tais esquemas de subdivisão se tornam impraticáveis na simulação de tecidos. Mesmo que se consiga evitar a divergência na integração numérica, as oscilações causadas na malha não serão facilmente disfarçadas.

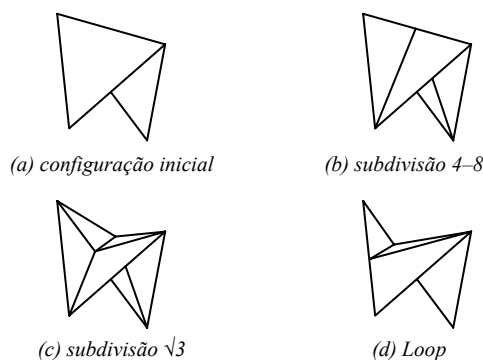


Figure 5 - Subdivisão de dois triângulos por diferentes métodos

Uma das grandes vantagens da subdivisão hierárquica 4-8 é a sua simplicidade, facto que a torna uma das técnicas de subdivisão mais eficientes. Outras características importantes, enumeradas em [Velho00] são: o grande poder de expressividade, a profundidade logarítmica (*logarithmic depth*) e o crescimento linear.

Na Figura 5 pode-se ver as diferenças entre as diversas técnicas de subdivisão. O exemplo refere-se à subdivisão de dois triângulos que partilham uma aresta. No caso da subdivisão de Loop, a situação apenas ocorre quando dois triângulos irregulares necessitam ser convertidos em 4 regulares. Como se pode observar, apenas a subdivisão 4-8 produz triângulos cujas orientações são consistentes com a situação de partida, pois os triângulos gerados estão situados no mesmo plano que os triângulos originais.

A subdivisão 4-8 permite ainda uma transição mais gradual e fina entre as sucessivas malhas, visto o número de partículas e triângulos gerados, em cada caso, ser inferior ao das outras duas técnicas. No caso da subdivisão $\sqrt{3}$, por cada passo de subdivisão uniforme é triplicado o número de faces e por cada triângulo da malha original é gerado um novo vértice. No caso da subdivisão de Loop, o factor de crescimento das faces é ainda maior, sendo criados 4 novos triângulos por cada

triângulo inicial. Estes valores são, em ambos os casos, superiores aos da subdivisão 4–8, onde o número de faces é multiplicado por 2 e o número de vértices adicionais é de 1 por cada dois triângulos.

5. TÉCNICA PROPOSTA

Em primeiro lugar, convém referir alguns detalhes de implementação que se prendem com o modelo do tecido usado, assim como a técnica de integração que faz avançar a simulação.

O nosso trabalho tem como ponto de partida o modelo de tecidos proposto por Baraff e Witkin [Baraff98]. O passo de integração é o passo de Euler implícito, que necessita resolver um sistema de equações esparsas. As estruturas de dados programadas foram optimizadas para operarem com matrizes esparsas e o produto de uma matriz esparsa por um vector tem complexidade linear no número de elementos da matriz.

O objectivo principal da utilização do refinamento e redução adaptativos é o de permitir introduzir detalhe onde ele é estritamente necessário, evitando assim gastar tempo de CPU na simulação de zonas do tecido onde uma aproximação poligonal menos rica seria suficiente. No caso da simulação de tecidos, os fenómenos que conferem maior realismo à simulação e que necessitam, simultaneamente, de maior detalhe são os que estão relacionados com a curvatura. As propriedades de elasticidade e de distorção, uma vez que são fenómenos que ocorrem no plano da malha, conseguem-se simular com recurso a malhas pouco discretizadas. Sendo assim, parece natural que o critério que orienta a subdivisão seja baseado na curvatura do tecido. Uma vez que se pretende um processo de subdivisão adaptativa eficiente, esse mesmo critério terá de ser avaliado localmente.

5.1. Critério de refinamento e redução

O estimador de erro por nós usado mede a distância da superfície poligonal da malha a uma superfície curva, idealizada, que aproxima, localmente, a curvatura do tecido virtual. Para cada par de triângulos, partilhando uma aresta, avalia-se a distância dos vértices opostos à aresta de junção e calcula-se a sua média, L . De seguida, o par de triângulos é aproximado por dois triângulos iguais cujos vértices opostos à aresta comum distam desta o mesmo valor L . A curvatura estimada é a de uma circunferência que passa nesses dois vértices e no ponto médio da aresta. A Figura 6 ilustra o que se acabou de referir. Recorrendo à trigonometria, o valor estimado do desvio da malha, em relação à superfície do tecido, é dado por

$$d = \frac{L(1 - \cos(\theta/2))}{2 \sin(\theta/2)}$$

onde θ é o ângulo formado pelas duas normais aos triângulos considerados.

A cada passo da simulação, as arestas partilhadas são analisadas e, sempre que o estimador ultrapasse um determinado limiar máximo, a subdivisão dessa aresta é accionada. O critério para a simplificação, como

operação inversa, é semelhante, mas seleccionando agora, para possível remoção, apenas as arestas para as quais o valor do estimador é inferior a um valor limiar mínimo. Na realidade, porém, a experiência mostrou que o processo não pode ser assim tão simples e veremos adiante as razões de tal facto, assim como a solução que preconizamos.

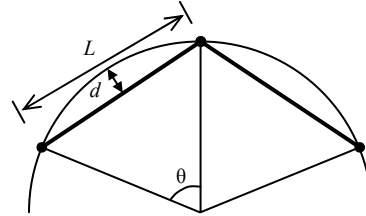


Figura 6 - Estimador da curvatura do tecido na junção de dois triângulos

Em [Volkov02] é usado o mesmo estimador do erro da aproximação poligonal à superfície do tecido, enquanto que em [Villard02] se preferiu estimar o erro nos vértices, analisando o maior dos desvios entre a normal no vértice – obtida ponderando as normais dos triângulos vizinhos – e as normais desses mesmos triângulos.

5.2. Algoritmo de subdivisão adaptativa

De um modo geral, os esquemas de subdivisão de malhas decompõem-se em dois tipos de regras:

- *Face rules* – determinam a localização dos novos vértices.
- *Vertex rules* – determinam as novas localizações dos vértices, já existentes, envolvidos na subdivisão.

Na transposição dos esquemas de subdivisão para a simulação de tecidos, pensamos que apenas o primeiro tipo de regras deverá ser usado. Com efeito, a aplicação das *vertex rules* provocaria mudanças instantâneas nas posições das partículas e nas orientações dos triângulos, facto que pretendemos evitar a todo o custo. Para além do mais, a eficiência é uma preocupação constante na simulação de tecidos, pelo que, ao deixar cair o segundo tipo de regras, estamos a melhorar o desempenho global da técnica.

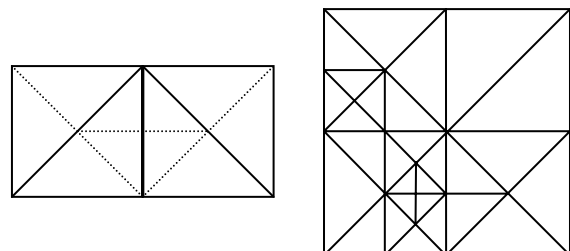


Figura 7 - Exemplos de subdivisão de uma aresta exterior a um bloco e de uma subdivisão não uniforme

Na subdivisão adaptativa de malhas 4–8, sempre que uma aresta desencadeia um processo de subdivisão é

necessário ter em conta se se trata de uma aresta interna ao bloco protótipo (Figura 2) ou se é uma aresta fronteira. No caso de se tratar de uma aresta exterior ao bloco, será necessário refinar, em primeiro lugar, cada um dos blocos adjacentes a essa aresta exterior. A situação encontra-se representada na Figura 7, assim como um exemplo da subdivisão adaptativa.

Existe ainda uma outra situação que é necessário precaver e que se manifesta quando a aresta que determina o refinamento une dois blocos de diferentes níveis de detalhe. Contudo, o nível de detalhe apenas poderá diferir de uma unidade, pelo que bastará subdividir recursivamente o bloco de nível inferior antes de se prosseguir. Eis o nosso algoritmo recursivo de subdivisão, apresentado em pseudo-código:

Subdivide48(Edge h)

begin

if h.splited then return

if h.left.level = max_level return

le = BreakingEdge(h.left)

re = BreakingEdge(h.right)

if h.left.level < h.right.level then

Subdivide48(le)

else if h.right.level < h.level.right then

Subdivide48(re)

// Criar os 4 novos triângulos nw, sw, ne e se
// situados nas direcções cardeais secundárias
// assim como as novas arestas en, es, ew, ee
// nas direcções cardeais principais

nw.level=sw.level=ne.level=se.level=h.left.level+1

if h.type = internal then

edge_type = external

else edge_type = internal

en.type=ee.type=es.type=ew.type=edge_type

return

end

Acrescente-se que a função *BreakingEdge* procura a aresta de um triângulo que está marcada como *internal*. A versão implementada é ligeiramente mais complexa que a apresentada, porque é ainda necessário tratar dos casos em que a subdivisão actua sobre uma aresta situada num bordo da malha. Nesse caso, apenas o campo *left* da aresta estará preenchido.

O processo de simplificação opera de forma inversa. Porém, apenas são consideradas as arestas que tiverem resultado de um processo de subdivisão anterior, sendo as outras dispensadas de testes futuros.

5.3. Estruturas Hierárquicas e gestão da memória

Como em todos os algoritmos hierárquicos de refinamento adaptativo, é necessário guardar a hierarquia de triângulos, exemplificada na Figura 3. No nosso simulador, cada uma das arestas *ew* e *ee* contém um apontador para um triplo formado pela aresta inicial *h* e pelas próprias arestas *ew* e *ee* (ver Figura 2). Note-se que, na nossa implementação, a informação acerca das

arestas *en* e *es* é recuperada seguindo as ligações das arestas para os triângulos adjacentes a elas. Esta opção permite poupar alguma memória e, das experiências efectuadas, a performance não sai demasiado penalizada pelas indirectões adicionais. Refira-se que a estrutura de dados que codifica a hierarquia não é, na realidade, uma árvore, mas sim uma floresta, existindo várias raízes. Outra característica interessante é que a hierarquia de níveis de detalhe não permite efectuar percursos da raiz até às folhas, o que não é necessário, mas sim no sentido inverso, das folhas para a raiz.

Na estratégia de gestão de memória do nosso simulador as operações de subdivisão são penalizadas face às operações de simplificação. Em cada momento, estão presentes, em memória, todos os triângulos (e demais objectos, tais como as arestas) necessários para representar a malha no seu nível de detalhe actual. Para além deles, também os triângulos de níveis superiores (na hierarquia) continuam em memória, apesar de não estarem a ser usados na simulação. Quando ocorre uma simplificação num determinado local da malha, os triângulos a eliminar da malha são destruídos e os do nível acima são reactivados. Este processo de reactivação permite que a simplificação seja uma operação bastante eficiente.

5.4. Simplificação estável da malha

A nossa escolha do método de subdivisão 4-8 elimina a maior parte das causas de instabilidade da simulação, durante os processos de refinamento e de redução da malha. A causa do temido comportamento caótico do tecido modelado é, na maior parte das vezes, originada por alterações instantâneas na geometria da malha que não são suportadas pelas equações que regem e descrevem o comportamento dinâmico do modelo.

Apesar de tudo, subsiste ainda o problema das operações de redução (ou simplificação) da malha. Aqui é impossível evitar a descontinuidade, porque o único cenário que o permitiria fazer seria o de ter os dois triângulos, aí em causa, num arranjo coplanar e a efectuarem um movimento solidário entre si.

A nossa primeira tentativa foi proceder à simplificação assim que o erro estimado da aproximação da malha ao tecido fosse inferior ao valor limiar estipulado. Mas este método simples revelou-se catastrófico. Uma análise mais cuidada à situação mostra que não bastará que os dois triângulos a fundir estejam praticamente coplanares. É também necessário saber se o arranjo é estável ou se é meramente transitório. O vector condição $C(\mathbf{x})$, proposto em [Baraff98] e usado para modelar as forças que contrariam a curvatura do tecido é definido, para cada par de triângulos, como sendo o ângulo θ que as normais dos dois triângulos fazem entre si (ver Figura 8).

Numa situação em que o ângulo permanece estável, então a derivada do vector condição, $\partial C(\mathbf{x})/\partial \mathbf{x}$, deverá ser nula. Neste caso, a condição C depende apenas das posições das 4 partículas envolvidas ($\mathbf{x}_i$, $\mathbf{x}_j$, $\mathbf{x}_k$ e $\mathbf{x}_l$), pelo que a sua derivada será obtida pelo vector de gradientes:

$$\frac{\partial C}{\partial x} = \left(\frac{\partial C}{\partial x_i} \quad \frac{\partial C}{\partial x_j} \quad \frac{\partial C}{\partial x_k} \quad \frac{\partial C}{\partial x_l} \right).$$

Felizmente, estes gradientes já necessitavam ser avaliados no decorrer da simulação [Baraff98], pelo que a penalização introduzida por este critério de estabilidade é quase nula. Assim, em vez de verificarmos se os gradientes são vectores nulos, verificamos simplesmente se os seus módulos se encontram abaixo de um valor limiar muito pequeno. Desta forma, a quase totalidade dos problemas de instabilidade, que verificávamos durante a simplificação da malha, ficaram resolvidos com este teste adicional.

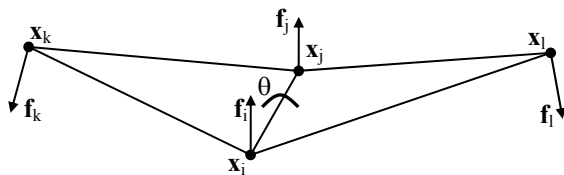


Figura 8 - Condição de curvatura

Outro dos problemas detectados foi a possibilidade de a gestão dinâmica do detalhe gerar ciclos apertados de subdivisão e simplificação. Embora estes ciclos sejam, na sua maior parte, transitórios, causam instabilidade, um efeito visual perturbador e gasta-se tempo de CPU desnecessariamente. A estratégia que decidimos implementar impede que o fenómeno possa ter uma alta frequência e traduz-se numa espécie de amortecimento da evolução da malha. Cada triângulo terá que existir por um período mínimo, parametrizável. Esta técnica simples impede que os triângulos gerados numa determinada altura da simulação possam, muito pouco tempo depois, ser destruídos devido ao processo de redução da malha. Note-se que com esta estratégia não impedimos que esses mesmos triângulos sejam subdivididos ainda mais. Esta característica revelou-se fundamental para que a subdivisão se possa adaptar, com rapidez, às variações de curvatura durante a simulação.

5.5. Manutenção das características físicas da malha

Ao contrário do que se verifica com os modelos de tecidos baseados em arranjos mola-massa [Villard02, Zhang02], uma das grandes vantagens do modelo contínuo, por nós utilizado, é a de não haver qualquer necessidade de ir ajustando, ao longo da simulação, as grandezas físicas que descrevem as características do tecido, tais como as constantes de elasticidade, distorção e curvatura.

Não basta, no entanto, calcular as novas propriedades da partícula gerada a cada passo da subdivisão. Se por um lado a sua posição de repouso é trivialmente calculada por interpolação das coordenadas (u,v), no espaço do tecido, das duas partículas situadas nos extremos da aresta subdividida, o cálculo da massa não se limita à

nova partícula gerada, sendo necessário proceder a uma distribuição de massas conforme se verá de seguida.

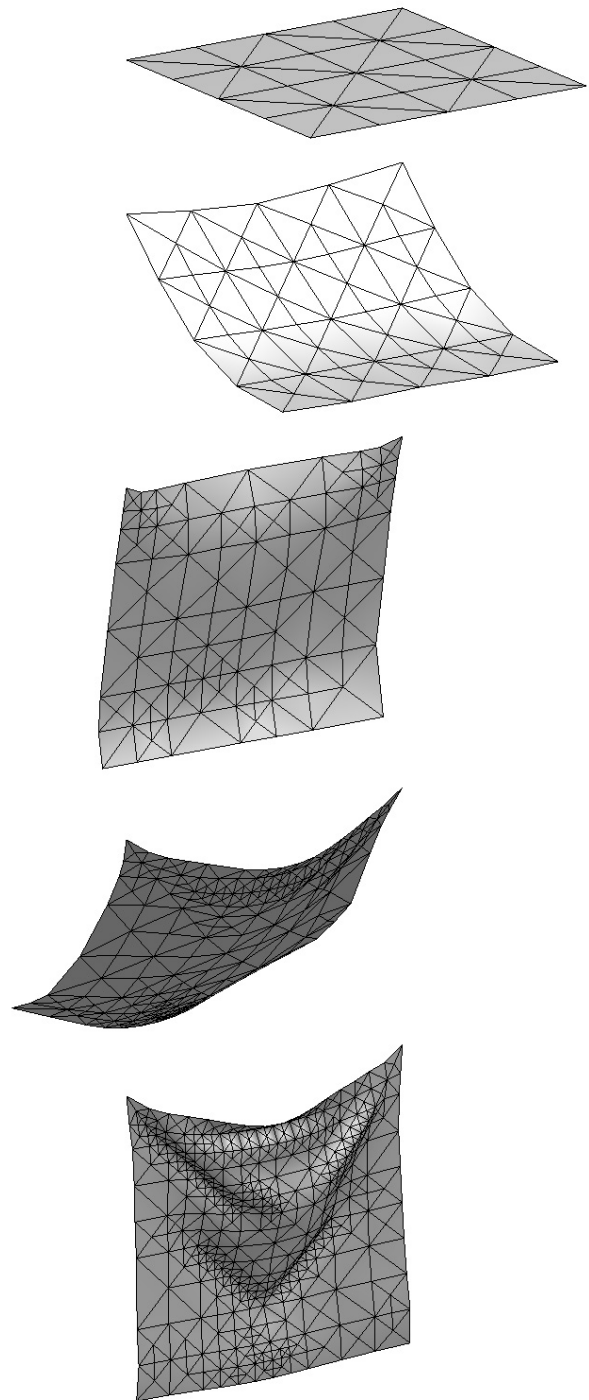


Figura 9 - Tecido em queda livre, suspenso em dois pontos

Durante a subdivisão da malha, a massa total do tecido terá de permanecer constante. No nosso caso, a massa de cada elemento é calculada multiplicando a sua área pela densidade do tecido (uma constante da simulação). A massa é então distribuída pelas três partículas que servem de suporte ao triângulo. Durante o processo de subdivisão e simplificação a massa é calculada de forma incremental. Na primeira fase, cada triângulo a remover

“liberta” o seu peso das partículas que o suportavam. Os novos triângulos efectuam o processo inverso, distribuindo, como indicado anteriormente, o seu peso pelas três partículas situadas em cada um dos seus vértices.

A distribuição final de massas assim obtida não é uniforme e, apesar de em [Volkov02] se apontar este fenómeno como causa de alguns artefactos durante a simulação, nomeadamente a deformação do tecido durante uma queda livre, as nossas experiências não confirmaram a existência de tais problemas.

6. RESULTADOS

A Figura 9 mostra vários passos de uma experiência, realizada partindo de um tecido, uniformemente dividido, formado por 25 partículas e 32 triângulos, e limitando a subdivisão a 6 níveis de profundidade. O tecido foi largado em queda livre estando, no entanto, suspenso por dois cantos. As imagens mostram como a subdivisão pode ser usada para, partindo de uma malha simplificada, e sem qualquer conhecimento prévio das condições da simulação, simular e mostrar detalhes do tecido que, de outra forma, apenas poderiam ser visíveis usando uma malha uniforme muito refinada.



Figure 10 - Tecido suspenso, subdividido dinamicamente

Na Figura 11 podem ver-se os tempos médios de CPU despendidos pelo algoritmo de subdivisão e simplificação adaptativas, para os primeiros 2 segundos de tempo simulado. O tempo médio de CPU gasto por frame, foi de 284.5ms, variando entre 3.27ms e 835.4ms. Sem recurso ao algoritmo, para se obterem triângulos com a mesma área dos de nível de detalhe maior, seria necessário recorrer a uma malha de 33×33 partículas. Nesse caso, cada passo da simulação demoraria, em média, cerca de 3012.7ms, ou seja, superior em mais de 10 vezes.

Note-se que a subdivisão e simplificação adaptativas apenas consomem, em média, cerca de 3% do tempo total gasto na simulação. Embora, por vezes, essa proporção possa atingir valores consideráveis, conforme se pode visualizar no gráfico, à medida que o número de partículas se torna mais significativo, o tempo de CPU

gasto é dominado pela resolução do sistema de equações lineares. Assim, a percentagem de tempo despendido pelas fases de subdivisão e simplificação converge, de forma rápida, para valores muito baixos.

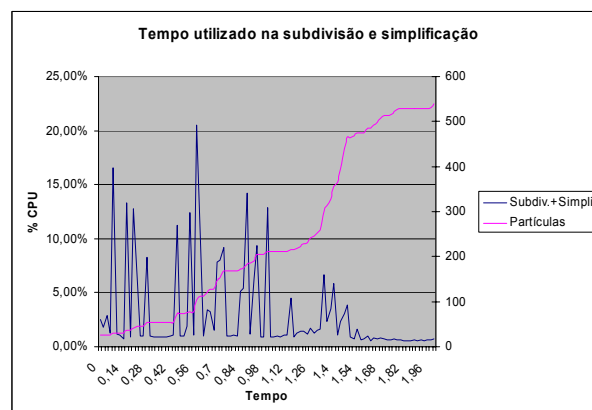


Figure 11 – Percentagem de tempo de CPU usado

Uma experiência diferente pode ser visualizada na Figura 12. Neste caso, uma malha, previamente subdividida de forma regular, é deixada cair sobre uma superfície circular. Na zona de contacto pode ver-se, facilmente, o resultado do processo de simplificação.

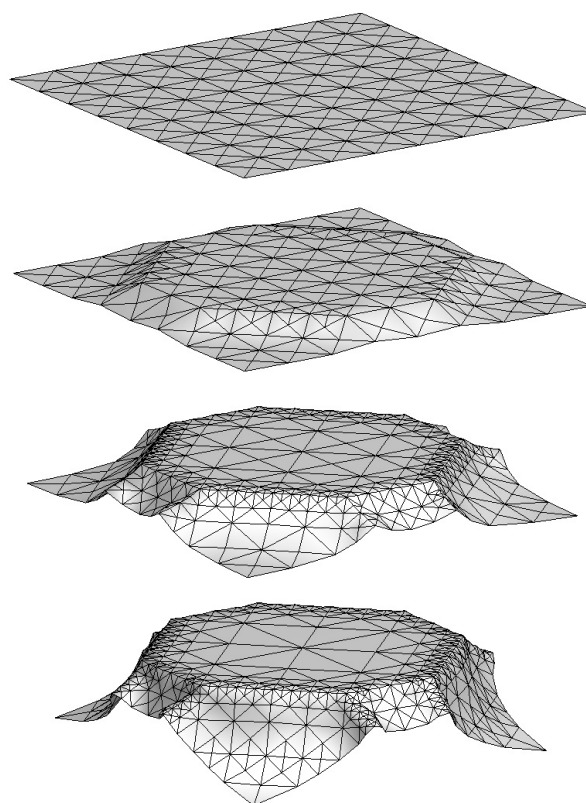


Figure 12 - Tecido a cair sobre uma superfície circular

7. CONCLUSÕES E TRABALHO FUTURO

Neste artigo mostrámos como a subdivisão de malhas 4–8 pode ser usada no contexto da variação dinâmica do nível de detalhe, aplicada à simulação de tecidos virtuais. O esquema de subdivisão usado exhibe boas características no que diz respeito à qualidade das

malhas que vão sendo sucessivamente obtidas, sem que a geometria dos triângulos se torne excêntrica ou degenera. A sua escolha teve como critério fundamental evitar problemas de descontinuidade geométrica que provocariam certamente um comportamento caótico do tecido. Outra vantagem do esquema 4-8 é o número reduzido de novos elementos (vértices e faces) introduzidos na malha a cada passo elementar de subdivisão. Este fenómeno permite um controlo mais gradual da evolução da malha. Apresentámos, igualmente, algumas técnicas que combatem os problemas de instabilidade e descontinuidade introduzidos pelas inevitáveis modificações instantâneas da geometria durante o processo de simplificação da malha. Os resultados experimentais mostram que o ganho obtido em termos computacionais é significativo e que o peso total das operações de refinamento e simplificação é negligenciável, tornando atractiva a utilização do método.

Um dos defeitos anteriormente apontados à técnica de Baraff, aqui utilizada, seria o de não haver garantia que a versão modificada do método do gradiente conjugado convergisse para a solução do sistema. Recentemente, em [Ascher02], para além de se provar a convergência desse método, os autores sugerem ainda uma pequena alteração na inicialização do algoritmo que permite uma aceleração do mesmo, fazendo-o convergir mais depressa. No entanto, não nos foi possível ainda implementar e verificar os ganhos que se poderão obter com tal alteração.

Por outro lado, estamos cientes que outras estratégias de gestão de memória, especialmente as que disponham de mecanismos de *cache*, poderiam, eventualmente, ser mais bem sucedidas no estabelecimento do compromisso entre ocupação de memória central e rapidez de execução. Mas, não sendo prioritário, também deixaremos esse estudo como tópico de investigação para trabalho futuro.

8. REFERÊNCIAS

- [Ascher02] Uri M. Ascher and Eddy Boxerman. On the modified conjugate gradient method in cloth simulation. Electronically available at: <http://www.cs.ubc.ca/~ascher>
- [Baraff98] David Baraff and Andrew Witkin. Large Steps in Cloth Simulation. In *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 43-54. ACM SIGGRAPH, 1998.
- [Breen92] D.E. Breen, D.H. House and P.H. Getto. A physically Based Particle Model of Woven Cloth. *The Visual Computer*, 8(5-6):264-277, June 1992.
- [Breen94] D.E. Breen, D.H. House and M.J.Wozny. Predicting the drape of woven cloth using interacting particles. *Computer Graphics (Proc. SIGGRAPH)*, pages 365-372, 1994.
- [Bridson02] R. Bridson, R. Fedkiw and J. Anderson. Robust Treatment of Collisions, Contact and Friction for Cloth Animation. *ACM Transactions on Graphics*, 21(3): 594-603, 2002.
- [DeRose98] Tony DeRose, Michael Kass and Tien Truong. Subdivision surfaces in character animation. In *Proceedings of the 25th annual conference on computer graphics and interactive techniques*, pages 85-94. ACM Press, 1998.
- [Dias00] J.M.S. Dias, M.N. Gamito and J.M. Rebordão. A Discretized Linear Elastic Model for Cloth Buckling and Drape. *Textile Research Journal*, 70(4):285-297, April 2000.
- [Eberhardt96] B. Eberhardt, A. Weber and W. Strasser. A Fast, Flexible, Particle-system for Cloth Draping. *IEEE Computer Graphics and Applications*, 16(5):52-59, September 1996.
- [Etmuss00] Olaf Etmuss, Bernhard Eberhardt, Michael Hauth, and Wolfgang Strasser. Collision adaptive particle systems. *Proceedings Pacific Graphics 2000*, pages 338-347, 2000.
- [Hadap99] S. Hadap, E. Bangarter, P. Volino and N. Magnent-Thalmann. Animating Wrinkles on Clothes. *IEEE Visualization '99*, pages 175-182. IEEE Computer Society Press, San Francisco, USA, October 1999.
- [Howlett98] P. Howlett and W.T. Hewitt. Mass-spring simulation using adaptive non-active points. *Computer Graphics Forum*, 17(3), 1998.
- [Hutchinson96] Dave Hutchinson, Martin Preston, and Terry Hewitt. Adaptive refinement for mass/spring simulations. In *Computer Animation and Simulation '96*, Eurographics, pages 31-45. Springer-Verlag Wien New York, 1996.
- [Kobbelt00] Leif Kobbelt. $\sqrt{3}$ -Subdivision. In *Proceedings of the 27th annual conference on Computer Graphics and interactive techniques*, pages 103-112. ACM Press/Addison-Wesley Publishing Co., 2000.
- [Loop87] C.T. Loop. *Smooth Subdivision Surfaces Based on Triangles*. Master's Thesis, Department of Mathematics, University of Utah, August 1987.
- [Provot95] Xavier Provot. Deformation Constraints in a Mass-Spring Model to Describe Rigid Cloth Behaviour. *Computer Interface Proceedings*, pages 147-154. Quebec City, Canada, May 1995.
- [Velho00] Luis Velho and Jonas Gomes. Variable Resolution 4-k Meshes: Concepts and Applications. *Computer Graphics Forum*, 19(4):195-214, 2000.
- [Velho01] Luis Velho and Denis Zorin. 4-8 Subdivision. *Computer Aided Geometric Design*, 18(5): 397-427, 2001. Special Issue on Subdivision Techniques.
- [Villard02] Julien Villard and Houman Borouchaki. Adaptive Meshing for Cloth Animation. *11th International Meshing Roundtable*, pages 243-252, Ithaca, New York, USA. September, 2002.
- [Volkov02] Vasily Volkov and Li Ling. Adaptive Local Refinement and Simplification of Cloth Meshes. In *First International Conference on Information*

Technology & Applications-ICITA 2002, Bathurst, Australia, 25-28 November 2002. Electronically available at:

<<http://charybdis.mit.csu.edu.au/~mantolov/CD/ICITA2002/papers/266-5.pdf>>

[Zhang01] Dongliang Zhang and Matthew M.F. Yuen. Cloth Simulation using Multilevel Meshes. *Computers & Graphics*, 25(3):383-389, 2001.

[Zhang02] Jian Zhang, Shahram Payandeh and John Dill. Haptic Subdivision: an Approach to Defining Level-of-detail in Haptic Rendering. *Symposium on Haptic Interfaces for Virtual Environments and Teleoperator Systems 2002*, pages 201–208.

Sessão 3

AMBIENTES VIRTUAIS

Computação Gráfica e Arqueologia Urbana

O caso de *Bracara Augusta*

Paulo Bernardes Manuela Martins
Unidade de Arqueologia, UM
Av. Central 39, 4710-228 Braga
{pbernardes, mmmartins}@uaum.uminho.pt

Resumo

A Computação Gráfica é, seguramente, uma ferramenta importante, e crescentemente utilizada, para representar e manipular grandes quantidades de informação de elevada complexidade.

Por sua vez a arqueologia, de uma forma geral, produz um manancial de informação de grande complexidade, pelo que a sua representação utilizando as tecnologias da Computação Gráfica constitui um elevado e aliciante desafio.

As reconstruções virtuais que representam arqueosítios, como, por exemplo, a cidade romana de Bracara Augusta, devem ser consideradas uma ferramenta importante e fundamental para o desenvolvimento da investigação levada a cabo por peritos (arqueólogos, arquitectos e especialistas em urbanismo) e para o despertar das consciências relativamente à importância da herança cultural.

Abstract

Computer Graphics is undoubtedly an important tool, widely used for representing and manipulating enormous amounts of high complex information.

Usually, the archaeological information is highly complex, so its representation using computer graphics technology is a true and engaging challenge.

Virtual reconstructions representing archaeological sites, as for example the roman town Bracara Augusta, should be considered a fundamental tool for the research improvement carried out by experts and extremely important for the uprising of cultural heritage consciousness.

Keywords

Reconstrução Virtual; Multimédia; Visualização; Arqueologia Urbana; Arqueologia Virtual;

1. INTRODUÇÃO

Nos últimos anos tem-se vindo a assistir a uma progressiva utilização da Computação Gráfica na representação de dados arqueológicos [Barcelo98]. A Arqueologia, de uma forma geral, produz e manipula um manancial de informação de grande complexidade, pelo que a sua representação, utilizando técnicas de aquisição de dados 3D e métodos de reconstrução 3D, constitui um elevado e aliciante desafio. Apesar de esta representação virtual ser de indiscutível importância para a Arqueologia, nem sempre ela é desenvolvida de modo a servir a Arqueologia. De um modo geral, a Computação Gráfica utiliza estes modelos virtuais para testar e validar os diferentes algoritmos desenvolvidos no âmbito da modelação, visualização e navegação em mundos virtuais.

Na verdade, muitas aplicações relacionados com a reconstrução de arqueosítios ou de espólio arqueológico conferem aos objectos reconstruídos uma aparência extremamente realista. Em muitos casos será até

exageradamente realista, tendo em conta que a sua reconstituição se baseia, muitas vezes, em dados bastante incompletos e mal documentados. Este facto mostra que o discurso arqueológico nem sempre foi (e em alguns casos ainda não é) entendido pela Computação Gráfica. No entanto, este cenário tem vindo a sofrer algumas alterações, principalmente a partir do momento em que os especialistas em Computação Gráfica começaram a ter um diálogo mais construtivo com os peritos ligados ao estudo do património arqueológico, ou até, em alguns casos, começaram a procurar uma formação complementar na área da Arqueologia.

Tal como é referido em [Martins00b], a adopção de novas tecnologias na Arqueologia, e particularmente na Arqueologia Urbana, para ajudar a investigação, a gestão e a apresentação do passado, é uma consequência natural que resulta da necessidade de gerir enormes quantidades de dados arqueológicos e, também, da evolução rápida das novas tecnologias, cada vez mais adaptáveis ao processamento de informação arqueológica.

O desenvolvimento de reconstruções virtuais que representam arqueossítios, como, por exemplo, a cidade romana de *Bracara Augusta*, deve ser considerado uma ferramenta importante e poderosa para o desenvolvimento da investigação levada a cabo por especialistas (arqueólogos, arquitectos, urbanistas) e para a ampliação da consciência cívica relativamente à importância da Herança Cultural.

1.1 Motivação

Em Arqueologia, os vestígios existentes acerca de uma estrutura podem ser apenas alguns fragmentos de muros, alicerces, pilares ou colunas extremamente danificados pela acção do Homem ao longo dos anos. Um edifício ou infra-estrutura pode ter sido alterado ou reconstruído diversas vezes, o que determina modificações significativas tanto na sua aparência como na sua funcionalidade. Daqui se pode inferir que uma necessidade dos arqueólogos consiste em utilizar uma ferramenta capaz de representar, para um determinado arqueossítio, interpretações alternativas, bem como mudanças ao longo do tempo.

As escavações realizadas em Braga nos últimos 27 anos produziram uma imensa quantidade de informação, apenas entendida por especialistas, que produzem um discurso muito próprio. Este discurso, devido à sua especificidade, nem sempre é correcta ou facilmente apreendido pelo público. Mas, para sensibilizar a população para os problemas e dificuldades que a Arqueologia Urbana enfrenta, particularmente em Braga, e para contribuir, no que diz respeito a *Bracara Augusta*, para a democratização do saber, é necessário simplificar o discurso arqueológico.

Assim, este trabalho não procurará desenvolver ou otimizar as tecnologias existentes da Computação Gráfica, mas antes entender o discurso dos arqueólogos e traduzir esse discurso para a linguagem própria da Computação Gráfica. Só desta forma se pode assegurar rigor e qualidade na divulgação da Arqueologia, ou da Herança Cultural, usando a Computação Gráfica: as imagens produzidas não serão apenas imagens bonitas, mas sim imagens ou reconstituições virtuais que servirão, também, de ferramenta ao arqueólogo na sua própria investigação.

1.2 Objectivos

O objectivo principal deste trabalho consiste em representar virtualmente a cidade de *Bracara Augusta*, bem como a sua evolução ao longo de cinco séculos. Este modelo virtual deverá permitir ao seu utilizador, independentemente da sua especialização, percorrer o espaço reconstruído de *Bracara Augusta*. Para além disso é também possível mostrar a evolução no tempo, de forma a fazer uma leitura da cidade em termos temporais.

Por outro lado, o mesmo modelo deverá servir, também, no processo de validação das reconstituições arquitectónicas que resultam da interpretação dos dados arqueológicos. As vantagens destas reconstruções virtuais não residem apenas na partilha do conhecimento. O próprio processo

de criação dos modelos tridimensionais conduz, com alguma frequência, a descobertas adicionais.

Em suma, a reconstrução virtual deverá ser capaz de servir de ferramenta de investigação aos peritos que têm de estudar os dados arqueológicos e, também, terá de cumprir a função de divulgar esses mesmos dados, já devidamente analisados e tratados, a um público muito mais vasto e leigo.

2. O PROJECTO BRACARA AUGUSTA

Como é referido em [Martins00a], a realidade portuguesa até aos anos 60, no que respeita a relação entre as cidades históricas portuguesas e a Arqueologia, era muito semelhante aos restantes países europeus. Nas décadas de 50 e 60 aparecem, com alguma frequência, vestígios romanos um pouco por todo o país. Isto porque foi nesta altura que se iniciou uma fase de renovação dos equipamentos urbanos, nomeadamente redes de saneamento, água e telefones. Com a abertura de valas de implantação para estes equipamentos urbanos, foram-se descobrindo vestígios arqueológicos por todo o lado.

Também foi nesta altura que se verificou, graças ao capital da emigração e um pouco por todo o país, uma expansão urbana que avançou de uma forma descontrolada para terrenos onde, em muitos casos, estavam sepultadas as antigas cidades romanas. Este avanço provocou, como infelizmente foi o caso de Braga, uma destruição irreparável de estruturas e de espólio, fundamentais para ampliar o conhecimento acerca do passado da cidade [Martins92].

Em Braga, precisamente para travar a destruição em massa que aqui decorria, foi criado em 1976 o Projecto de *Bracara Augusta* [Martins98], que se destinava a salvar a zona Arqueológica de Braga. Este projecto, que possui contornos muito particulares, uma vez que foi criado por uma decisão governamental, permitiu a constituição de uma equipa, dependente da Universidade do Minho, para fazer face às intervenções arqueológicas na área urbana de Braga.

Ao longo de mais de 1 quarto de século de existência, este projecto teve de enfrentar diversas dificuldades e passou por diferentes fases que ditaram diferentes estratégias e ritmos de intervenção arqueológica na cidade de Braga.

Inicialmente, o objectivo deste projecto era o salvamento da área arqueológica da cidade romana e envolveu um conjunto de iniciativas legislativas e um vasto trabalho arqueológico de salvamento e emergências. Este trabalho foi acompanhado por prospecções geofísicas para delimitar, com algum rigor, a área de *Bracara Augusta*. Esta definição rigorosa da área da antiga cidade romana tinha em vista a protecção legal dos terrenos que faziam parte dessa área. Já na década de 80 o projecto de salvamento evoluiu para um projecto de investigação, que procurava definir o urbanismo e a arquitectura de *Bracara Augusta*.

No entanto, um conjunto de circunstâncias menos favoráveis remeteram para segundo plano a componente

científica, dando de novo, prioridade aos salvamentos e emergências. Não obstante, o projecto de *Bracara Augusta* mantém, ainda hoje, tanto a componente de salvamento, como a de investigação. Este facto faz dele um projecto pioneiro e único no nosso país. A direcção científica deste projecto depende da Unidade de Arqueologia da Universidade do Minho e esta situação permitiu que as intervenções arqueológicas em Braga não fossem consideradas escavações de circunstância, mas sim integradas numa intervenção coerente em termos de investigação da antiga cidade romana.

Também na área da informática aplicada à Arqueologia, este projecto se tornou pioneiro. De facto, desde 1996 [Martins98] a equipa responsável pelo projecto *Bracara Augusta* apostou na informatização dos dados provenientes de dezenas de intervenções realizadas em Braga. Estes dados foram então integrados no Sistema de Informação Arqueológica de *Bracara Augusta* (SIABRA) [Giestal98], que é um SIG dedicado ao projecto *Bracara Augusta*. O SIABRA é, como todos os SIG, um sistema geo-referenciado que trata a informação arqueológica de Braga com vista a garantir e auxiliar a preservação, investigação, gestão e divulgação do património arqueológico da cidade.

A etapa seguinte deste projecto consistiu em recorrer às potencialidades da reconstrução virtual para proceder à representação tridimensional de *Bracara Augusta*. Esta representação, que foi realizada a partir dos dados arqueológicos que se encontram no SIABRA, possibilita um melhor entendimento de *Bracara Augusta*, tendo em conta que se podem visualizar as reconstituições de alguns espaços desta cidade romana.

3. COMPUTAÇÃO GRÁFICA EM ARQUEOLOGIA

Ainda hoje a aplicação da Computação Gráfica, particularmente dos ambientes virtuais e multimédia, é mais conhecida junto de áreas tais como a arquitectura e planeamento urbano, a medicina, a visualização científica e a formação e treino. No entanto, a área da Herança Cultural Virtual e tem vindo a tornar-se cada vez mais importante, no que diz respeito à conservação, preservação e interpretação do imenso legado histórico da Humanidade [Refsland00]. De facto, a vasta área de aplicação dos ambientes virtuais permite-nos concluir que estes constituem uma forma privilegiada de divulgação de aspectos culturais e científicos. A Cultura e a Ciência são assimiladas e compreendidas mais facilmente, uma vez que os utilizadores podem interagir com o mundo virtual e manipular os dados, produzindo diferentes modelos de conhecimento.

As primeiras aplicações da Computação Gráfica à arqueologia datam dos finais dos anos 70 [Badler78], começaram a intensificar-se em finais dos anos 80 e inícios dos anos 90 e foram desenvolvidas em países como Inglaterra [Reilly87], Estados Unidos [Cornforth91] e Alemanha [Jakobs94].

No entanto, é de notar que quando se começou a aplicar estas novas tecnologias à Arqueologia os resultados nem sempre satisfaziam as necessidades dos especialistas nem a curiosidade do público [Addison00]. Esse fracasso deveu-se, fundamentalmente, ao facto da tecnologia de então ainda estar numa fase larvar e, também, à circunstância de haver pouca sensibilidade, por parte dos peritos das novas tecnologias, para o discurso arqueológico. Hoje, estas dificuldades estão mais suavizadas tendo em conta que as tecnologias ligadas à reconstrução virtual e multimédia evoluíram muito e, também, que as equipas que trabalham em projectos de reconstituição de um sítio arqueológico são cada vez mais multidisciplinares.

Um dos primeiros trabalhos apresentados nesta área, em Portugal, foi a reconstituição virtual do Mosteiro de Santa Clara-a-Velha [Teixeira97], em Coimbra. É de notar que o interesse de investigadores ligados à Computação Gráfica pela arqueologia já se verifica há algum tempo, como se pode verificar em [Brodie87] e em [Spicer87], e que, com a evolução da Computação Gráfica, este interesse se justifica cada vez mais [Berndt00].

3.1 A importância da CG em Arqueologia

A reconstrução arqueológica, usando um grande número de técnicas de Modelação Geométrica e de Computação Gráfica, é uma ferramenta de suporte à investigação muito útil se os modelos forem rigorosos. Como o rigor é um grande objectivo de todos os arqueólogos, é necessário investir em recursos que permitam a obtenção do mesmo, por forma a preservar e transmitir subtilezas que, de outro modo, se poderão perder, ser ignoradas em processos futuros de estudo e investigação ou deixar de ser conhecidas pelas gerações vindouras.

A divulgação da informação arqueológica, usando os ambientes virtuais e o multimédia, é também de enorme importância para a democratização do saber, prestando um importante auxílio, no que diz respeito à descodificação da informação arqueológica para o grande público e, consequentemente, favorecendo a consciencialização da importância do património arqueológico.

A partir de ruínas e de fragmentos de diferentes artefactos, os arqueólogos tem de recriar modelos que sejam capazes de transmitir a vivência de comunidades desaparecidas há vários séculos, ou até mesmo milénios. A sua tarefa não se resume apenas a escavar, a recolher os dados e a organizá-los, mas também e fundamentalmente a interpretar e criar modelos cognitivos do passado.

Precisamente para criar uma visualização do passado, os arqueólogos dispõem já de um conjunto de ferramentas na área da Computação Gráfica, que concretiza as suas representações mentais através de modelos tridimensionais. Estas reconstituições 3D virtuais servirão para a investigação e poderão ser partilhados

com outras pessoas. O trabalho apresentado em [Abouaf99] reflecte bem esta situação.

Efectivamente, os ambientes virtuais aplicados à Arqueologia são uma ferramenta particularmente interessante para os arqueólogos divulgarem as suas interpretações. Em vez de descrever uma escavação recorrendo a um texto e a desenhos, pode-se reconstruir um arqueosítio em três dimensões e explorá-lo interactivamente.

Além disso, à medida que novos dados vão sendo disponibilizados, o modelo virtual pode ser actualizado com essa informação. Outra forma de melhorar os modelos do passado passa por introduzir-lhes maior detalhe e mais rigor, recorrendo ao trabalho cooperativo, em ambiente virtual, entre arqueólogos. Esta partilha de conhecimento verifica-se também com o público em geral. Geralmente, a informação sobre um sítio arqueológico é divulgada em publicações e, também, através de exposições, em museus. No entanto, neste último caso esta forma de divulgação apresenta algumas limitações:

- Grande parte dos museus reservam apenas algum espaço para as suas exposições, pelo que cada uma pode estar montada apenas durante algum tempo e mostrando apenas parte do espólio;
- Durante a exposição o espólio de um sítio arqueológico corre sempre o risco de ser danificado por acção humana ou mesmo furtado;
- A exposição de algumas peças à acção do ar e da luz directa pode causar danos irreparáveis às mesmas.

Uma forma de minimizar estas limitações passa precisamente pela utilização de ambientes virtuais para divulgar não apenas o espólio de um arqueosítio, mas também os contextos onde esse espólio foi encontrado.

Assim, e para que o rigor da informação científica seja preservado, é necessária uma elevada atenção durante a produção dos conteúdos que pretendem ser divulgados. Estes dependem, naturalmente, dos diferentes tipos de público-alvo a que se destinam. Para os especialistas os conteúdos terão muito mais importância do que o meio de divulgação, uma vez que aquilo que lhes interessa é o rigor científico da informação.

De facto, para o arqueólogo as simulações em ambientes virtuais podem e devem ser entendidas como informação complementar aos mapas, plantas e cortes com os quais se encontra familiarizado. Por sua vez, o estudante é especialmente sensível à forma como a informação lhe é transmitida. Para o grande público o interesse poderá residir essencialmente nos aspectos de inovação tecnológica, que complementam o conteúdo. Neste último caso, o conteúdo deverá conter informações de carácter geral que facilmente possam ser absorvidas por um público menos exigente.

3.2 Alguns Projectos

No que diz respeito à aplicação de ambientes virtuais na reconstituição de espólio existem alguns projectos muito

interessantes, como é o caso de [Steckner98] e de [Brogni98].

Um trabalho também particularmente válido nesta área está descrito em [Zheng00]. O projecto em questão, que conta na sua equipa com arqueólogos, conservadores de museu e investigadores na área da informática, usa a Computação Gráfica, ambientes virtuais e multimédia para investigar a possibilidade de recuperar algumas relíquias desenterradas na China, no "*Museum of the Terra Cotta Warriors and Horses*". O sítio arqueológico, que foi descoberto à cerca de 20 anos, conta com uma equipa de oitenta pessoas a qual já pôs a descoberto aproximadamente 3000 esculturas das 8000 estimadas. Este projecto de recuperação procura concretizar os seguintes objectivos:

- Digitalização do espólio escavado;
- Teste e utilização de técnicas de recuperação em Ambiente Virtual;
- Apresentar os resultados numa exposição virtual.

Muito embora a importância da reconstituição virtual do espólio de um arqueosítio seja indiscutível e mereça, segundo [Papaioannou01], particular atenção por parte da informática, esta é maioritariamente utilizada na reconstituição de estruturas e edifícios desaparecidos ou em avançado estado de degradação.

Um exemplo claro desta prática é o que se está a fazer no Sítio Romano de *Sagalassos* (Sudoeste da Turquia) [Pollefeys98]. Há, também, projectos que procuram restituir as estruturas de uma forma mais completa, como é o caso da restituição da *Domus Aurea* de Nero e do Coliseu em Roma [Forte98a], das termas de Badalona (*Baetulo*) [Costa98], da reconstrução da cidade medieval Turku na Finlândia [Uotila98] ou da reconstrução do castelo de Ferrara em Itália [Forte98b].

Actualmente estão a decorrer vários projectos que utilizam os ambientes virtuais como uma interface para a investigação e análise arqueológica. Um destes projectos é o ARCHAVE, que é um sistema que representa todos os componentes de uma escavação arqueológica no contexto de uma reconstrução arquitectónica do Templo de Petra na Jordânia [Acedo00].

O ARCHEOGUIDE (*Augmented Reality based Cultural Heritage On-site GUIDE*) é um outro projecto que utiliza já a realidade aumentada para o estudo e divulgação de um sítio arqueológico ou de interesse cultural [Hildebrand00]. Estes exemplos atestam, sem dúvida, alguma importância dos ambientes virtuais na arqueologia e mostram, também, que o recurso a esta tecnologia está em fase crescente.

4. O CASO DE BRACARA AUGUSTA

Como já foi referido anteriormente, a evolução urbana da cidade de Braga ditou, em alguns casos, a destruição de importantes vestígios arqueológicos referentes à ocupação romana desta cidade. Este facto transformou a interpretação arqueológica, já por si bastante difícil e subjectiva, numa tarefa extremamente delicada.

Por este motivo, a reconstituição virtual da antiga cidade romana de *Bracara Augusta* foi, e continuará a ser, uma tarefa complexa, que sofrerá uma evolução constante à medida que forem descobertos novos vestígios, os quais produzirão mais informação para enriquecer e completar a reconstituição actual.

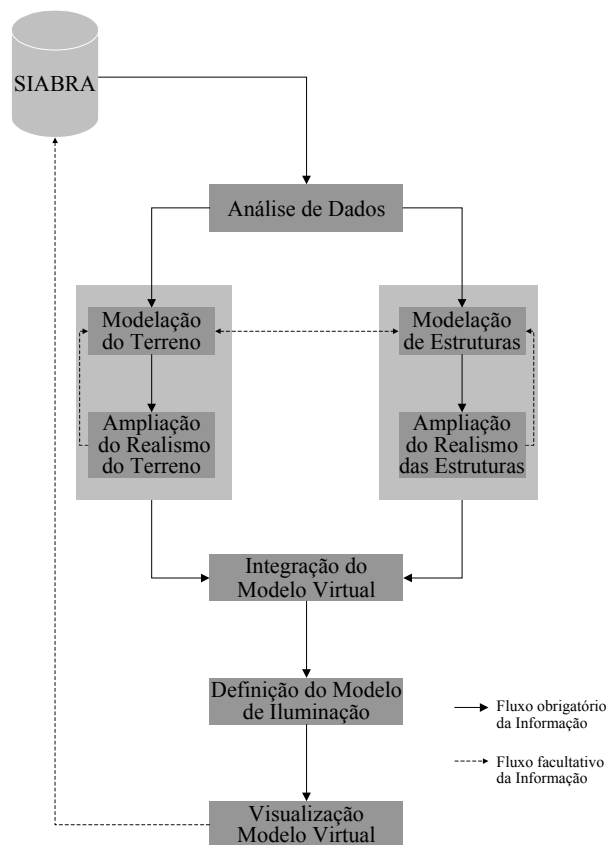


Figura 1: Metodologia de restituição de *Bracara Augusta*

O modelo tridimensional de *Bracara Augusta* surgiu a partir de um processo construtivo que compreendeu um conjunto de tarefas, as quais podem ser agrupadas nas seguintes fases (ver figura 1):

- Análise dos dados existentes;
- Modelação das estruturas e do terreno envolvente;
- Texturação dos modelos;
- Integração dos modelos das estruturas no modelo do terreno;
- Definição de um modelo de iluminação adequado à cena virtual;
- Visualização da reconstrução virtual de *Bracara Augusta*.

4.1 Análise de Dados

A análise dos dados provenientes das escavações e já devidamente interpretados é, de facto, o ponto de partida para todo o processo de reconstrução virtual de *Bracara Augusta*. Estes dados estão disponíveis no SIABRA e materializam-se da seguinte forma:

- Plantas das escavações;
- Alçados, planos e cortes das estruturas escavadas;
- Valores de cotas de terreno;
- Informação fotográfica;
- Informação escrita;
- Plantas, cortes e alçados resultantes da interpretação dos dados das escavações.

É, principalmente, este último tipo de informação que vai ser preponderante na modelação tridimensional das estruturas que foram encontradas em *Bracara Augusta*. São estes dados que vão dar forma à cidade.

Todos os outros elementos são necessários para, com rigor, aumentar o grau de realismo e de pormenor dos modelos tridimensionais das construções que compõem o modelo virtual de *Bracara Augusta*.

A análise de todos estes dados foi de capital importância para:

- Compreender a dimensão e complexidade de *Bracara Augusta*;
- Hierarquizar correctamente os modelos das estruturas;
- Definir o grau de detalhe na construção de cada modelo tridimensional;
- Estabelecer os níveis de detalhe mais adequados para a visualização.

4.2 Desenvolvimento dos Modelos Virtuais

Como já foi mencionado anteriormente, os dados mais importantes para restituir volumetricamente *Bracara Augusta* são as plantas, cortes e alçados existentes sobre as estruturas encontradas. A partir destes dados, procurou-se modelar geometricamente todos os edifícios e infra-estruturas conhecidas da cidade romana fazendo uso da técnica de modelação de sólidos conhecida por Geometria de Sólidos Construtiva (*Constructive Solid Geometry - CSG*) [Mäntylä88].

O terreno que envolve a cidade romana foi recriado usando as curvas de nível de um mapa do século XIX. Tendo em conta a natureza destes dados, o tipo de modelação que será aplicada neste caso não será a modelação de sólidos, mas sim, a modelação de superfícies.

No caso do terreno de *Bracara Augusta*, a modelação de superfícies consistiu em recriar uma superfície a partir de um conjunto finito de pontos existentes no espaço.

4.2.1 Estruturas Arquitectónicas e Infra-estruturas

A reconstituição tridimensional das estruturas arquitectónicas de *Bracara Augusta* representam um exercício bastante complexo que obedeceu a uma metodologia comum, apresentada nos parágrafos seguintes, e desenvolvida em duas fases:

- Modelação;
- Ampliação do Realismo.

Os modelos tridimensionais das estruturas arquitectónicas foram desenvolvidos utilizando o *MicroStation 95* da *Bentley* [Olson96] e o *Caligari trueSpace™4 v4.3* [CALIGARI98].



Figura 2: Pavimentos de circulação à cota exacta

Nas estruturas arquitectónicas de *Bracara Augusta*, o primeiro passo consistiu em modelar as plataformas de circulação dos edifícios, com base na informação da planta dos mesmos e no valor das cotas de terreno. Assim, como se pode observar na figura 2, o chão de cada edifício é representado por um conjunto de cubos cuja forma inicial foi modificada por transformações geométricas apropriadas. No caso particular das infra-estruturas de *Bracara Augusta* este primeiro passo consistiu em adaptar o traçado, tanto da muralha como das ruas, à morfologia do terreno, que na circunstância é bastante irregular.

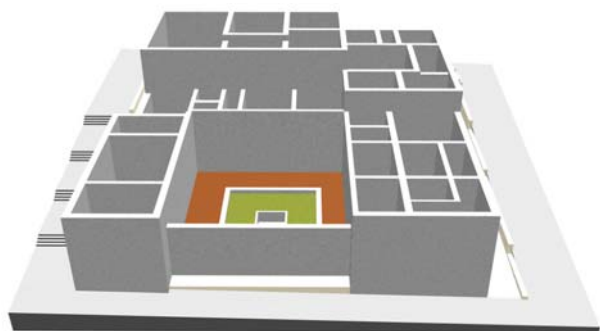


Figura 3: Estruturas de um edifício

O passo seguinte nesta fase de criação dos modelos virtuais consistiu em recriar as paredes interiores e exteriores do edifício. Estas paredes também foram modeladas a partir de cubos, aos quais são aplicadas transformações geométricas, que os posicionam correctamente no modelo e lhe dão a forma aproximada (ver figura 3). Depois, e tendo em conta a informação dos alçados e dos cortes projectados pelos, são criados sólidos que têm os contornos das janelas, das portas ou de outras aberturas e com espessura superior à das paredes do modelo. Estes objectos vão depois cruzar as paredes nos sítios onde estão localizadas as diferentes aberturas e, com o auxílio de operações de subtracção, são criadas nas paredes dos edifícios as respectivas portas e janelas (ver figura 4).

O passo seguinte consistiu em modelar, ainda tendo em conta a informação das plantas e os modelos

arquitectónicos, a estrutura que suportava o telhado. Esta estrutura é modelada, mais uma vez, a partir de uma forma simples, na circunstância um cubo, que é modificado a partir de operações booleanas regulares e transformações geométricas até atingir a forma desejada. Depois é posicionada sobre o modelo já existente, que posteriormente é coberto com objectos modelados que representam as telhas romanas (as *tegulae*). O efeito final da modelação pode ser observado na figura 5. Para as infra-estruturas, este último passo é dispensável.



Figura 4: Aberturas nas paredes

Frequentemente, após a modelação da estrutura do telhado e da cobertura, verifica-se a necessidade de proceder a alguns ajustes, no que diz respeito à modelação das paredes e de interiores.

Durante toda a fase de modelação existiu sempre a preocupação de estabelecer uma hierarquia correcta entre os vários objectos para facilitar, mais tarde, todo o processo de *rendering* e de animação de objectos que compõem os modelos.



Figura 5: Modelo completo de um edifício Romano

As propriedades materiais são um aspecto fundamental, no que concerne ao grau de realismo com que se pretende visualizar um cenário virtual, tendo em conta que o material é a informação atribuída a um objecto para que este tenha uma determinada aparência.

A hierarquização da cena é, também aqui, importante para a tarefa de definição das propriedades materiais dos diferentes objectos. Na verdade, o modelo virtual vai ficar estruturado em árvore, onde as folhas representam todos os objectos da cena, e os nós correspondem a agrupamentos de objectos que possuem o mesmo tipo de material. Desta forma, o processo de atribuição das características materiais aos diferentes objectos está facilitado. Mais ainda, fica minimizada a hipótese de se

esquecer de atribuir, ou de atribuir erradamente, características materiais a algum objecto.

Assim, para todos os objectos, que compõem a reconstituição virtual de *Bracara Augusta*, são definidas as respectivas propriedades materiais, cujos parâmetros são:

- A cor;
- A reflexão difusa e especular;
- O índice de refração;
- O brilho e transparência.

A tarefa da texturação de um modelo tridimensional começa com a criação de texturas que confirmem ao modelo um aspecto mais realista. Este processo é naturalmente facilitado quando as estruturas encontradas, durante a intervenção arqueológica, se encontram em bom estado de conservação. Neste caso, as texturas são obtidas a partir de fotografias de estruturas, tratadas num sistema de processamento de imagem.

No caso de *Bracara Augusta*, no entanto, não é muito frequente encontrar estruturas que possam servir de base para texturas realistas. Aqui, e porque a arquitectura e o urbanismo romano obedeciam a regras muito claras, a criação de texturas passa por um processo de pesquisa de sítios que possuam estruturas semelhantes e contemporâneas às de *Bracara Augusta*. O levantamento fotográfico dessas estruturas e o posterior tratamento digital permite, então, a obtenção de texturas adequadas ao modelo 3D.

A aplicação das texturas ao modelo 3D finaliza a tarefa da texturação e não envolve qualquer dificuldade.

4.2.2 Terreno Envolvente

A modelação do terreno envolvente foi desenvolvida utilizando, também, o *MicroStation 95* da *Bentley* para o tratamento bidimensional da informação. O modelo 3D do terreno foi obtido com o *AutoCAD® Land Development Desktop R2* da *Autodesk* [Autodesk99].

Utilizando as ferramentas do *AutoCAD® Land Development* para a criação de terrenos, foi modelada uma superfície que corresponde ao terreno da área envolvente de *Bracara Augusta*, representada através de uma rede irregular de triângulos (*Triangulated Irregular Network – TIN*).

Para diminuir o número de polígonos da superfície do terreno, transformou-se a superfície, constituída por uma rede de triângulos, numa superfície de malha de quadriláteros. Apesar de se verificar uma diminuição de precisão no modelo do terreno, este passo é extremamente vantajoso uma vez que se reduz para cerca de um quarto o número de faces. Desta forma contribui-se para um melhor desempenho computacional de todo o modelo de *Bracara Augusta* virtual.

Todo o processo para incrementar o realismo visual do terreno é semelhante ao descrito para o caso da modelação de estruturas arquitectónicas. Também para o terreno é necessário definir uma cor, atribuir um valor

adequado à reflexão difusa e especular, ao brilho e à transparência e, ainda, estabelecer o valor ideal para o índice de refração.

4.3 Integração da Informação

A integração dos modelos tridimensionais consistiu em posicionar as diferentes estruturas e infra-estruturas correctamente sobre o modelo do terreno de *Bracara Augusta*.

Naturalmente que existem sempre alguns ajustes e adaptações que se têm de fazer quando se integram as estruturas e infra-estruturas num terreno, mas, tendo em conta que todo o trabalho feito anteriormente, desde o levantamento dos dados no sítio arqueológico à integração da informação no SIABRA, foi sempre de elevado rigor e qualidade, esta tarefa não se revelou de execução difícil. Na figura 6 podemos observar o efeito final da integração da informação.

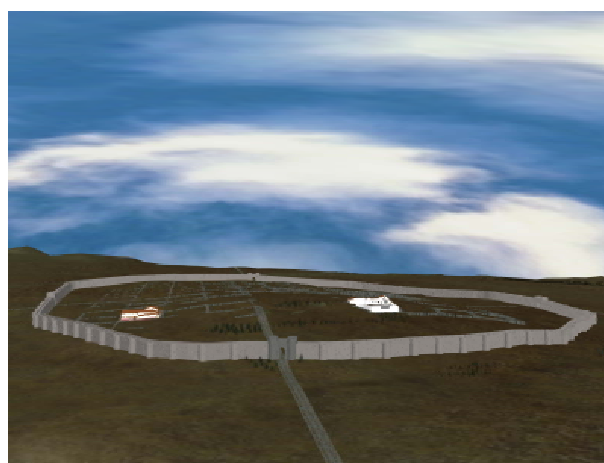


Figura 6: Integração da informação

Para além dos modelos rigorosos que foram criados, houve também necessidade de desenvolver modelos de edifícios simplificados, com os quais se preencheram todos os quarteirões de *Bracara Augusta* e sobre os quais ainda não existe informação, ou os seus dados ainda não foram convenientemente analisados e interpretados (ver figura 7).

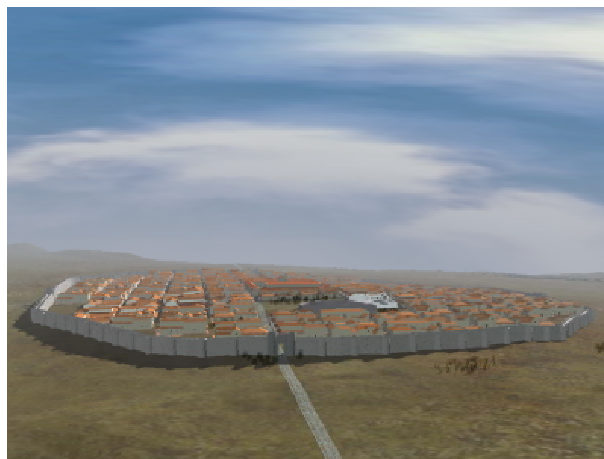


Figura 7: Reconstrução global de *Bracara Augusta*

4.4 Modelo de Iluminação

Um modelo de iluminação pretende ser um meio para aumentar o realismo visual de um cenário virtual. Para isso, é feita uma combinação de diferentes tipos de luz e fontes de luz, tais como:

- Luz Ambiente;
- Fonte de Luz Direcional;
- Fonte de Luz Pontual.

No caso da reconstituição virtual de *Bracara Augusta*, o modelo de iluminação encontrado despreza a componente da luz ambiente e combina apenas a fonte de luz direcional com a fonte de luz pontual. Desta forma, o modelo de iluminação resulta numa maior aproximação à realidade.



Figura 8: Iluminação global

O modelo de iluminação utilizado para a reconstituição virtual de *Bracara Augusta* compreende uma fonte de luz direcional, posicionada estrategicamente na cena e com a qual se consegue simular o Sol e, consequentemente, a iluminação global do espaço (ver figura 8).



Figura 9: Iluminação interior

Para além das fontes de luz direccionais, foram utilizadas fontes de luz pontuais de modo a simular a iluminação associada aos edifícios do modelo virtual. A diferente combinação deste tipo de luz conseguiu produzir um

conjunto de sombras, tornando mais realistas os interiores dos compartimentos e dos espaços dos conjuntos arquitectónicos que foram reconstituídos (ver figura 9).

5. CONCLUSÕES E TRABALHO FUTURO

O passado afigura-se como um elemento cada vez mais importante na sociedade contemporânea. Daí que a sua correcta representação se torna crescentemente importante. No entanto, é importante ter sempre em mente que esta representação do passado não é o passado em si, mas apenas um modelo do mesmo que resulta dos dados disponíveis e da sua interpretação.

Sem qualquer tipo de dúvida, pode-se afirmar que a Computação Gráfica, o Multimédia e os Ambientes Virtuais possuem técnicas e metodologias fundamentais para a representação desse passado, contribuindo, efectivamente, para uma melhor conservação, preservação e interpretação do património cultural e, muito particularmente, do património arqueológico.

A representação virtual de *Bracara Augusta* constituiu um importante passo no projecto de *Bracara Augusta*, na medida em que, a partir de agora, os dados sobre as escavações são representados, sempre que possível, de uma forma tridimensional. Para além da tridimensionalidade do espaço, também a representação cronológica da evolução da ocupação espacial é feita a três dimensões.

A importância desta forma de representação da informação arqueológica reflecte-se, principalmente, a nível da divulgação dos resultados de escavação junto do público leigo. De facto, um público não-especialista entende muito melhor um determinado arqueosítio se tiver a possibilidade de o ver reconstituído virtualmente. Por outro lado, desta forma, a informação arqueológica não será pertença exclusiva dos arqueólogos, contribuindo para uma maior e importante democratização do saber. Assim, torna-se mais fácil alertar a população em geral para a problemática da preservação e conservação do património arqueológico, inculcando nelas uma maior consciência para a herança cultural.

Mas, as vantagens são, também, imensas para o especialista que está a estudar um sítio arqueológico. A representação virtual de um determinado sítio arqueológico baseia-se, sempre, numa interpretação elaborada por uma equipa de especialistas (arqueólogos, arquitectos e historiadores). Essa representação é apenas um dos modelos interpretativos. Os dados existentes poderão contribuir para o aparecimento de outros modelos interpretativos, que nem por isso serão menos válidos, partindo do princípio que se baseiam em pressupostos verdadeiros. O facto de estas representações serem modelos geométricos tridimensionais, facilita a criação de outros modelos geométricos que representam diferentes interpretações.

Se é verdade que os modelos virtuais carecem de ser validados pelos especialistas, não é menos verdade que

algumas interpretações de estruturas são corrigidas durante o processo de criação dos modelos virtuais. De facto, tradicionalmente as interpretações são representadas em suporte papel, ou seja, em desenhos bidimensionais. Ao transformar esta informação em modelos tridimensionais, pode acontecer que a equipa de investigação seja alertada para algumas inconsistências, que rapidamente são ultrapassadas. Apesar de ser globalmente positivo representar o património arqueológico em ambiente virtual, isto não significa que este procedimento apresente apenas vantagens para a Arqueologia.

As representações gráficas em suporte computacional são, por vezes, de tal forma realistas que rapidamente podem ser entendidas como se de realidade se tratasse. Este facto representa um equívoco que pode ser perigoso, na medida em que um público menos informado pode confundir a virtualidade com a realidade. Não é isso que é pretendido com a aplicação dos ambientes virtuais à Arqueologia. Esta deve ser entendida como forma de recriar o passado, facilitando a sua assimilação e simultaneamente uma ampliação do próprio conhecimento, pois a sua representação melhora significativamente as grelhas mentais com que o arqueólogo lê o registo arqueológico.

6. BIBLIOGRAFIA

- [Abouaf99] J. Abouaf. The Florentine Pietà: Can visualization solve the 450-year-old Mystery? *IEEE Computer Graphics and Applications*, vol.19, num.1, Janeiro-Fevereiro, págs.6-10. 1999.
- [Acevedo00] D. Acevedo, E. L. Vote, D. Laidlaw e M. S. Joukowsky, *ARCHAVE: A Virtual Environment for Archaeological Research*, in *IEEE Visualization 2000 Proceedings, Virtual Reality in Archaeology*, Salt Lake City - Utah - USA 2000.
- [Addison00] A. C. Addison. Emerging Trends in Virtual Heritage. *IEEE Multimedia*, vol.7, num.2, Abril-Junho, págs.22-25. 2000.
- [Autodesk99] Autodesk, *AutoCAD Land Development Desktop Release*, 1999. User's Guide.
- [Badler78] N. I. Badler e V. R. Badler, *Interaction with a Color Computer Graphics System for Archaeological Sites*, in *Computer Graphics SIGGRAPH'78 Proceedings*, 12(3), 1978.
- [Barcelo98] J. A. Barcelo, M. Forte e D. H. Sanders, *The Diversity of Archaeological Virtual Worlds*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, et al., Eds., BAR International Series 843, Barcelona 1998.
- [Berndt00] E. Berndt e J. C. Teixeira. Cultural Heritage in the Mature Era of Computer Graphics. *IEEE Computer Graphics and Applications*, vol.20, num.1, Janeiro-Fevereiro, págs.36-37. 2000.
- [Brodlie87] K. W. Brodlie, *Standardisation in Computer Graphics: an introduction to GKS*, in *Computer Applications and Quantitative Methods in Archaeology - CAA'87 Proceedings, British Archaeological Reports*, C. L. Ruggles e S. P. Q. Rahtz, Eds., BAR International Series 393, Leicester 1987.
- [Brogni98] A. Brogni, E. Bresciani, M. Bergamasco e F. Silvano, *An Interactive System for the Presentation of a Virtual Egyptian Flute in a Real Museum*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, et al., Eds., BAR International Series 843, Barcelona 1998.
- [CALIGARI98] CALIGARI, *CALIGARI trueSpace4*, 1998. User's Guide.
- [Cornforth91] J. Cornforth, C. Davidson, C. J. Dallas e G. R. Lock, *Visualising Ancient Greece: Computer Graphics in the Sacred Way Project*, in *Computer Applications and Quantitative Methods in Archaeology - CAA'91 Proceedings, British Archaeological Reports*, G. Lock e J. Moffet, Eds., BAR International Series S577, Oxford 1991.
- [Costa98] J. G. Costa e E. G. Costa, *The Baths of Baetulo: From Archaeological to Virtual Reality*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, et al., Eds., BAR International Series 843, Barcelona 1998.
- [Forte98a] M. Forte, *About Virtual Archaeology: Disorders, Cognitive Interactions and Virtuality*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, et al., Eds., BAR International Series 843, Barcelona 1998.
- [Forte98b] M. Forte e D. Borra, *The Estense Castle of Ferrara (Italy): Multimedia Project and Virtual Reconstruction*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, et al., Eds., BAR International Series 843, Oxford 1998.
- [Giestal98] C. D. Giestal. *Sistema de Informação Geográfica para a Arqueologia Urbana: o caso de Bracara Augusta*. Departamento de História, Universidade do Minho, Braga, 1998. (Tese de Mestrado).
- [Hildebrand00] A. Hildebrand, P. Dähne, F. Seibert, I. T. Christou, A. Demiris, M. Diorinos, N. Ioannidis, L. Almeida, A. Diogo e J. Weidenhausen, *An Augmented Reality Based System for Personalized Tours in Cultural Heritage Sites*, in *International Conference on Augmented, Reality Virtual Environments and 3D Imaging Proceedings*, Grécia 2000.
- [Jakobs94] K. Jakobs e K. Kleefeld, *Multimedia Communication in Archaeology - Why and How*, in *Computer Applications and Quantitative Methods in Archaeology - CAA'94 Proceedings, British Archaeological Reports*, J. Hugget e N. Ryan, Eds., BAR International Series S600, Glasgow 1994.

- [Mäntyla88] M. Mäntyla, *An Introduction to Solid Modeling*, USA, Computer Science Press, 1988.
- [Martins92] M. Martins. *Bracara Augusta: A memória de uma cidade. Cadernos de Arqueologia*, vol.8-9, págs.177-197. 1992.
- [Martins98] M. Martins e F. S. Lemos. Duas Décadas de vida de um Projecto: o Salvamento de *Bracara Augusta. Cadernos de Arqueologia*, vol.14-15, págs.9-21. 1998.
- [Martins00a] M. Martins. *A Arqueologia em Busca da Cidade*. Braga. 2000.
- [Martins00b] M. Martins e P. Bernardes. A Multi-Disciplinary Approach for Research and Presentation of *Bracara Augusta's* Archaeological Heritage. *Archeologia e Calcolatori*, vol.11, págs.347-357. 2000.
- [Olson96] N. A. Olson, *MicroStation 95 Fundamentals*, Indianapolis - Indiana - USA, New Riders Publishing, 1996.
- [Papaioannou01] G. Papaioannou, E.-A. Karabassi e T. Theoharis. Virtual Archaeologist: Assembling the Past. *IEEE Computer Graphics and Applications*, vol.21, num.2, Março-Abril, págs.53-59. 2001.
- [Pollefeys98] M. Pollefeys, M. Proesmans, R. Koch, M. Vergauwen e L. v. Gool, *Acquisition of Detailed Models for Virtual Reality*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, *et al.*, Eds., BAR International Series 843, Barcelona 1998.
- [Refsland00] S. T. Refsland, T. Ojika, A. C. Addison e R. Stone. Virtual Heritage: Breathing new life into our Ancient Past. *IEEE Multimedia*, vol.7, num.2, Abril-Junho, págs.20-21. 2000.
- [Reilly87] P. Reilly e J. Richards, *New Perspectives on Sutton Hoo: The potential of 3-D Graphics*, in *Computer Applications and Quantitative Methods in Archaeology - CAA'87 Proceedings, British Archaeological Reports*, C. L. Ruggles e S. P. Q. Rahtz, Eds., BAR International Series 393, Leicester 1987.
- [Spicer87] D. Spicer, *Computer Graphics and the Perception of Archaeological Information: Lies, Dammed Statistics and ... Graphics!*, in *Computer Applications and Quantitative Methods in Archaeology - CAA'87 Proceedings, British Archaeological Reports*, C. L. Ruggles e S. P. Q. Rahtz, Eds., BAR International Series 393, Leicester 1987.
- [Steckner98] C. Steckner, *Form and fabric, the Real and the Virtual - Roman Economy-related Geometrical Mass Constrains in Dressel's Table of Amphora Forms*, in *Virtual Reality in Archaeology Proceedings, British Archaeological Reports*, J. A. Barcelo, *et al.*, Eds., BAR International Series 843, Barcelona 1998.
- [Teixeira97] J. C. Teixeira, A. Côte-Real, P. Bernardes e F. P. d. Macedo, *Virtual Environments and Cultural Heritage*, in *The Fifth International Conference on CAD and Graphics Proceedings*, China 1997.
- [Uotila98] K. Uotila e M. Sartes, *Medieval Turku - The Lost City. A Project trying to reconstruct a Medieval Town in Finland*, in *Virtual Reality in Archaeology - CAA'98 Proceedings, British Archaeological Reports*, J. A. Barcelo, *et al.*, Eds., BAR International Series 843, Barcelona 1998.
- [Zheng00] J. Y. Zheng. Virtual Recovery and Exhibition of Heritage. *IEEE Multimedia*, vol.7, num.2, Abril-Junho, págs.31-34. 2000.

Modelação Expedita de Cenas Urbanas

António Fernando Coelho
UTAD
Vila Real
acoelho@utad.pt

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

Fernando Nunes Ferreira
FEUP
Porto
fnf@fe.up.pt

Sumário

A natureza complexa e extensa dos ambientes urbanos torna morosa e dispendiosa a tarefa de construção de modelos virtuais destes. No entanto, a grande quantidade de informação disponível em formato digital sobre estas áreas motiva a automatização dos processos de modelação de forma a reduzir a intervenção humana.

Neste artigo apresenta-se um sistema para a modelação de ambientes urbanos baseado em informação disponível em formatos e plataformas diversas. A integração dos dados é efectuada utilizando XML, um standard para interoperabilidade. Constitui-se os processos de modelação, baseados na utilização de sistemas L e especificados num formato baseado em XML, denominado XL3D. O sistema permite a geração de forma automática de um modelo inicial, possibilitando o seu melhoramento de forma incremental através da adição de informação e/ou do refinamento dos processos de modelação.

Palavras-chave

Modelação 3D, Sistemas L, Realidade Virtual, XML, X3D, Computação Gráfica, Sistemas de Informação Geográfica

1. INTRODUÇÃO

A geração de modelos tridimensionais de áreas urbanas coloca um grande número de problemas no âmbito da Computação Gráfica, dado que a quantidade de informação necessária para a criação de modelos realistas é dependente da dimensão e complexidade da área a modelar. Embora existam alguns casos em que os modelos tenham sido criados por modelação manual e individual de cada um dos seus componentes, os processos semi-automáticos provaram que podem ser muito mais eficientes, reduzindo a duração, o custo e a alocação de recursos humanos [Pimentel01] [Dodge98].

1.1 Motivação

A natureza complexa dos ambientes urbanos, e a sua grande relevância socio-económica, levam inúmeros profissionais a realizar estudos sobre eles, recolhendo, analisando, processando e produzindo grande quantidade de informação. Actualmente, a grande maioria desta informação é armazenada em formato digital, na forma de ficheiros diversos ou em bases de dados.

Muita dessa informação disponível sobre áreas urbanas constitui um enorme potencial para servir de base à automatização do processo de criação de modelos tridimensionais de áreas urbanas através de processos automáticos. No entanto, a natureza abstracta, contextualizada e segmentada desta informação, bem como a diversidade de formatos e de plataformas em que se encontra disponível, dificultam o processo de desenvolvimento deste tipo de aplicações.

Verifica-se assim que existe um problema que é o de possibilitar a utilização de informação em formatos e plataformas diferentes. Uma solução para colmatar este problema passa pela utilização de standards de interoperabilidade como o XML (*Extended Modelling Language*) [XML]. Por outro lado, dado o carácter contextualizado e segmentado da informação relacionada com áreas urbanas, pode tornar-se necessária a integração de diversos documentos, bem como a dedução de alguma da informação que não está presente na informação de base.

Como exemplo, referira-se o caso da modelação de um edifício com base na informação disponibilizada num SIG (Sistema de Informação Geográfica). A informação geográfica relativa ao edifício é bidimensional e apenas representa a projecção desse edifício no elipsóide terrestre. Para se poder construir o poliedro correspondente, é necessária, no mínimo, alguma informação relacionada com a altura. Esta informação pode ser obtida de dados diversos, tais como o ponto de cércea (altitude do ponto mais elevado do edifício) ou o número de andares do edifício. Dado que esta informação se pode encontrar como atributo em temas geográficos distintos, ou numa base de dados relacional, torna-se necessária a sua integração. Adicionalmente, os edifícios podem ter estilos diferentes consoante a data de construção, sendo que esta pode ser encontrada na escritura e a sua arquitectura pode estar ligada à região da cidade em que se localiza.

A inadequação, ou mesmo falta de informação constitui assim um outro problema, cuja solução pode passar pela

capacidade de ampliação de dados, bem como pela introdução de alguma aleatoriedade, em partes do modelo em que a informação é escassa e não é necessário um elevado nível de realismo. Estas características são típicas de ferramentas matemáticas como os sistemas L [Prusinkiewicz90], que têm sido bastante utilizadas na Computação Gráfica, nomeadamente para a modelação de organismos e fenómenos naturais [Deussen98] [Prusinkiewicz01] [Lane02], embora existam poucos exemplos da sua utilização para a modelação de ambientes urbanos [Parish01].

A motivação principal deste trabalho é o desenvolvimento de um sistema de modelação para a criação de modelos virtuais de cidades reais. O sistema deve operar de forma semi-automática, utilizando informação existente previamente ou a recolher, para o processo de modelação. Dado o número elevado de elementos a modelar para cada cidade virtual e a variabilidade da quantidade de informação disponível em cada caso, o sistema deve ser capaz de deduzir informação, de forma a preencher lacunas na informação de base. A escassez de informação de modelação não deve, no entanto, ser impeditiva da obtenção de um elevado grau de detalhe do modelo virtual. Todo o processo de modelação deve ser controlado de forma concisa e genérica pelo utilizador, de forma a obter-se uma solução inicial, cujo realismo possa ser incrementalmente aumentado, através do refinamento da especificação do processo de modelação.

1.2 A Questão da Interoperabilidade

Ao nível da Computação Gráfica 3D, a questão da interoperabilidade tem sido abordada pelo consórcio Web3D [Web3D] através do desenvolvimento do X3D (*Extended 3D*) [X3D]. Sendo um desenvolvimento do VRML (*Virtual Reality Modelling Language*) [VRML97], o X3D possibilita a codificação num formato baseado em XML, sendo este um standard de interoperabilidade *de facto* para a representação e transferência de informação.

A informação geográfica é possivelmente o tipo de informação com maior relevância para a modelação dos espaços urbanos, uma vez que representa, em forma de mapas (representação bidimensional), aspectos geométricos e de localização geográfica, que são extremamente úteis para a geração dos modelos tridimensionais. Ao nível da interoperabilidade, também o consórcio *OpenGIS* [OpenGIS] tem desenvolvido alguns standards baseados em XML. A título de exemplo, refira-se o GML (*Geography Markup Language*) [GML] para representação da informação geográfica e o WFS (*Web Feature Service*) [WFS], uma especificação de serviços Web para disponibilização de informação geográfica. A integração que se propõem deste tipo de informação no processo de modelação é assim fundamental, dado que se pretendem desenvolver soluções automáticas.

As questões da reutilização dos processos de modelação, bem como da modularidade, são relevantes no aspecto

da eficiência e celeridade de criação de uma solução inicial por parte do utilizador. O projecto CONTIGRA [Dachsel02] é um claro exemplo de como a codificação XML do X3D pode contribuir claramente para a construção de modeladores que possibilitem a integração e reutilização de componentes no processo de construção de aplicações tridimensionais.

2. OS SISTEMAS L

Os sistemas L (acrónimo de Sistemas Lyndenmayer) foram introduzidos em 1968 por Astrid Lyndenmayer [Lyndenmayer68], e têm sido utilizados para a modelação da morfologia de uma grande variedade de organismos e entidades. A teoria matemática que os sustenta tem pontos de contacto com as teorias dos autómatos abstractos e das gramáticas formais.

Alguns dos pontos fortes dos sistemas L são a ampliação de dados [Smith84], que permite a geração de estruturas complexas a partir de um conjunto pequeno de dados, e a emersão. O conceito de emersão reflecte processos em que um conjunto de unidades interactuantes adquire, de forma qualitativa, novas propriedades que não podem ser deduzidas através de uma simples sobreposição das contribuições individuais [Taylor92]. Assim, estas novas propriedades “emergem” a partir da representação inicial.

Um dos conceitos fundamentais no contexto dos sistemas L, é o de módulo, designando uma unidade construtiva discreta que é repetida à medida que o sistema se desenvolve. Exemplo concreto deste conceito é o dos ramos e das folhas no caso da modelação de uma árvore.

A essência do desenvolvimento ao nível modular é capturada através de um sistema de reescrita paralela, que substitui individualmente os módulos ascendentes, por configurações de módulos descendentes.

Todos os módulos pertencem a um alfabeto finito de tipos de módulos e todo o comportamento de desenvolvimento modular é especificado através de um conjunto, também finito, de regras de reescrita ou de produção.

No seu caso mais simples, os sistemas L de contexto livre, cada regra de produção substitui um módulo predecessor, por zero, um, ou mais módulos sucessores. No caso dos sistemas sensíveis ao contexto, a aplicabilidade de uma determinada regra de produção não depende apenas do módulo predecessor, mas também dos seus vizinhos.

As regras de produção são aplicadas em paralelo, sendo reescritos todos os módulos simultaneamente em cada passo de derivação. A sequência de estruturas obtida em passos de derivação consecutivos a partir de uma estrutura inicial predefinida (axioma) denomina-se sequência de desenvolvimento.

Os sistemas L paramétricos [Prusinkiewicz95] estendem o conceito base dos sistemas L, com atributos numéricos em cada símbolo. Os sistemas L paramétricos operam com palavras paramétricas, ou seja, cadeias de módulos com parâmetros associados. Neste tipo de sistema, a

aplicação de uma regra de produção pode ser dependente da avaliação de uma condição.

Os sistemas L estocásticos [Prusinkiewicz90] adicionam o carácter da aleatoriedade, através da atribuição de um valor de probabilidade às regras de produção.

A nossa abordagem aos processos de modelação passa pela utilização de sistemas L sensíveis ao contexto, paramétricos e estocásticos.

3. O SISTEMA DE MODELAÇÃO XL3D

No presente documento apresenta-se um sistema de modelação para cenas urbanas que responde a uma especificação de modelação, gerando de forma automática um modelo tridimensional em formato X3D.

O sistema baseia-se numa abordagem procedimental, onde toda a modelação é especificada de forma declarativa, baseada em documentos segundo um XML-Schema [XML-Schema] denominado XL3D.

Os dados para modelação, em formatos baseados em XML, são convertidos, através de transformações XSLT [XSLT], em cadeias de módulos. Estas cadeias são concatenadas para cada um dos processos de modelação.

O processo de modelação é definido e controlado por sistemas L. Através de um conjunto de regras de produção, definidas pelo utilizador, os dados iniciais são transformados, através de um processo iterativo, até se obter uma solução satisfatória.

A solução final obtida é posteriormente interpretada, formando um *scenegraph* em formato X3D.

3.1 Processo de Modelação

Os sistemas L paramétricos conferem ao processo de modelação um elevado poder de ampliação de dados, assim como de emersão, gerando novos dados a partir do conhecimento associado às regras de produção definidas pelo utilizador. Por outro lado, a utilização de sistemas L estocásticos permite a incorporação de aleatoriedade no processo de modelação, de forma a simular a natureza diversificada dos cenários urbanos.

A primeira fase do processo de modelação consiste na formação de um axioma inicial, seguindo-se a sua transformação através da aplicação de regras de produção e, finalmente, a interpretação da cadeia resultante, para geração de um segmento de *scenegraph* em formato X3D.

3.1.1 Axioma

Em primeiro lugar torna-se necessário criar uma cadeia inicial de módulos que sirva como axioma.

A cadeia inicial é construída a partir de diversos ficheiros XML que são transformados em cadeias de módulos através de transformações XSLT. O resultado final da transformação deve obedecer ao XML-schema apresentado na figura 1.

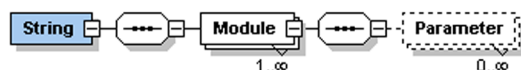


Figura 1 – Esquema da cadeia de módulos

```
<wfs:Edificado fid="Edificado.1">
  <wfs:NPisos>5</wfs:NPisos>
  <wfs:Altitude>435</wfs:Altitude>
  <wfs:Geometry2>
    <gml:Polygon srsName="Port:ugal">
      <gml:outerBoundaryIs>
        <gml:LinearRing>
          <gml:coordinates>
            524000.084,      211684.916
            524004.656,      211684.519
            524005.727,      211695.958
            524007.280,      211696.026
            524007.608,      211703.813
            524007.605,      211703.813
            524000.084,      211704.467
            524000.084,      211684.916
          </gml:coordinates>
        </gml:LinearRing>
      </gml:outerBoundaryIs>
    </gml:Polygon>
  </wfs:Geometry2>
</wfs:Edificado>
```

Figura 2 - Segmento de ficheiro GML

Por exemplo, partindo de um elemento de edificado representado num ficheiro GML (figura 2) e após aplicação de uma transformação XSLT, obtém-se uma cadeia de módulos como se mostra na figura 3.

```
<XL3D:String>
  <XL3D:Module name="Origem">
    <XL3D:parameter>
      524000.08
    </XL3D:parameter>
    <XL3D:parameter>435</XL3D:parameter>
    <XL3D:parameter>
      2116849.16
    </XL3D:parameter>
  </XL3D:Module>
  <XL3D:Module name="Edificio">
    <XL3D:parameter>5</XL3D:parameter>
    <XL3D:parameter>
      0.000 0.000, 4.572 -0.397,
      5.643 11.042, 7.196 11.110,
      7.524 18.897, 7.521 18.897,
      0.000 19.551, 0.000 0.000
    </XL3D:parameter>
  </XL3D:Module>
</XL3D:String>
```

Figura 3 - Cadeia de módulos

Como se pode ver, a cadeia resultante é constituída por módulos paramétricos, compostos por um nome e por uma sequência de parâmetros. Na criação de um axioma pode utilizar-se uma destas cadeias, ou integrar várias, segundo um processo de concatenação ou de composição através de uma transformação XSLT.

3.1.2 Transformação

A transformação do axioma (figura 4) é realizada através de um conjunto de regras de produção definidas pelo utilizador. Nesta fase, o conhecimento do utilizador sobre

o problema é utilizado para ampliar o conjunto de dados associados ao axioma.

Origem(524000.084, 435, 211684.916)
Edificio(5,"0.000 0.000, 4.572 -0.397,
 5.643 11.042, 7.196 11.110, 7.524
 18.897, 7.521 18.897, 0.000 19.551
 0.000 0.000")

Figura 4 - Axioma

Cada regra de produção é aplicada a uma determinada cadeia de módulos parametrizados.

A notação utilizada para a especificação das regras de produção é a dada por:

$$n : E < P > D : cond \xrightarrow{prob} S_1 \dots S_n \quad (1)$$

Onde, P representa o módulo a identificar e a substituir por um ou vários módulos sucessores S ; E representa os módulos vizinhos situados à esquerda; D os módulos vizinhos situados à direita, $cond$ é a condição da qual depende a aplicabilidade da regra, $prob$ a probabilidade da regra de produção e n o número da regra de produção.

Uma determinada regra de produção só é aplicada se existir um módulo P na cadeia a transformar e se a condição associada for verdadeira. No caso de ser definido um contexto, ou seja um conjunto de módulos situados na vizinhança (à sua esquerda ou à sua direita) do módulo predecessor, também é necessário que estes coincidam para que a regra seja aplicável.

Sempre que existam várias regras de produção aplicáveis para cada módulo predecessor a substituir, selecciona-se aleatoriamente uma delas, com base na probabilidade associada.

Só após efectuar a avaliação e selecção das regras de produção, é que estas se aplicam sincronamente a toda a cadeia, gerando a cadeia seguinte.

2:Edificio(npisos,contorno) $\xrightarrow{0.25}$
EdificioF1(npisos, contorno, 1)
3:Edificio(npisos,contorno) $\xrightarrow{0.25}$
EdificioF1(npisos, contorno, 2)
4:Edificio(npisos,contorno) $\xrightarrow{0.25}$
EdificioF1(npisos, contorno, 3)
5:Edificio(npisos,contorno) $\xrightarrow{0.25}$
EdificioF1(npisos, contorno, 4)
6:EdificioF1(npisos,cont,mat) $\xrightarrow{0.5}$
EdificioF2(npisos,cont,mat,1)
7:EdificioF1(npisos,cont,mat) $\xrightarrow{0.5}$
EdificioF2(npisos,cont,mat,2)

Figura 5 - Regras de produção iniciais

Este processo repete-se sucessivamente, até que se chegue a uma situação em que mais nenhuma regra de

produção seja aplicável, ou que se tenha atingido um número máximo de iterações.

A título de exemplo, o axioma apresentado na figura 4 pode ser transformado, de acordo com um conjunto de regras de produção, numa cadeia que descreve a estrutura de um edifício para a geração do *scenegraph* do seu modelo tridimensional em X3D.

Na figura 5 apresenta-se um subconjunto de regras de produção, do tipo estocástico, que permitem atribuir aleatoriamente um determinado material e tipo de telhado ao edifício a ser gerado (figura 6).

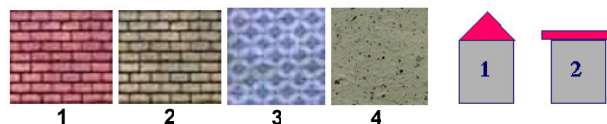


Figura 6 - Materiais de parede e tipo de telhado

Um outro conjunto de regras de produção expandem o contorno 2D inicial numa sequência de arestas, calculando o valor do seu perímetro, bem como criam um contorno tridimensional.

Na figura 7 apresenta-se a regra que constrói a estrutura base do *scenegraph* final. O edifício é transformado num LOD (*Level Of Detail*) com um modelo pormenorizado visível até 250 metros de distância, e um modelo mais simples, só com o bloco do edifício, para distâncias superiores.

```
22:Translacao(x,y,z)<EdificioF4
(npisos,cont,mat,telhado,perim,cont3
D,npts)→LOD(0,0,0,100) [ Grupo
[ Predio(mat,cont,npisos*3,perim) ]
[ Translacao(0,npisos*3,0)
[ Telhado(telhado,cont,perim) ]
[ Caleira(mat,cont3D,perim) ] ]
[ Rebordo(mat,cont3D,perim) ]
[ ConstFachada(mat,cont3D,npts,
npisos*3) ]
[ ConstRC(mat,cont3D,perim,npts) ]
] PredioSimples(mat,cont,npisos*3) ]
```

Figura 7 - Regra de produção que estrutura o edifício

O modelo pormenorizado é composto por um bloco (volumetria), telhado, caleira, rebordo da base, rés do chão e restante fachada.

```
31:
InstElemFachadaF4(npisos,mat,x1,y1,d
x,dy,nbandas,banda,ang): banda>0
→ InstJanelaF1(npisos,mat,x1+dx
*(2*banda-1)/(2*nbandas),y1+dy*
(2*banda-1)/(2*nbandas),ang)
InstElemFachadaF4(npisos,mat,x1,y1,
dx,dy,nbandas,banda-1,ang)
```

Figura 8 - Regra para a construção de bandas

O contorno do edifício é decomposto nas diferentes faces da fachada do edifício. Para cada face é calculada a sua

largura, e a partir deste valor, a regra 31 (figura 8) determina o número de bandas de janelas ou varandas.

Na figura 9 apresentam-se as regras de produção que instanciam os elementos da fachada que integram as bandas. As regras 33 e 34 determinam aleatoriamente se essa banda é uma janela ou varanda, sendo a probabilidade da primeira duas vezes superior à segunda. Na regra 35 decompõe-se cada banda nas posições relativas aos diferentes pisos do edifício. Para dar um aspecto mais realista ao edifício, nas regras 36 a 39, selecciona-se aleatoriamente se a janela deve estar com a persiana aberta ou entreaberta.

33: InstJanelaF1 (npisos,mat,x,y,ang) $\xrightarrow{0.67}$ InstJanelaF2 (npisos,mat,x,y,ang,0)
34: InstJanelaF1 (npisos,mat,x,y,ang) $\xrightarrow{0.33}$ InstJanelaF2 (npisos,mat,x,y,ang,1)
35: InstJanelaF2 (nps,mat,x,y,a,v) : npisos > 1 $\xrightarrow{\quad}$ [InstJan (x,3*(nps-1)+1.5,y,a,mat,v)] InstJanelaF2 (nps-1,mat,x,y,a,v)
36: InstJan (x,y,z,a,m,v) :v=0 $\xrightarrow{0.67}$ Jan (x,y,z,a,m)
37: InstJan (x,y,z,a,m,v) :v=1 $\xrightarrow{0.67}$ Varan (x,y,z,a,m)
38: InstJan (x,y,z,a,m,v) :v=0 $\xrightarrow{0.33}$ Jan2 (x,y,z,a,m)
39: InstJan (x,y,z,a,m,v) :v=1 $\xrightarrow{0.33}$ Varan2 (x,y,z,a,m)

Figura 9 - Regras de produção para a fachada

Após a execução de todas as iterações, o sistema L termina a sequência de desenvolvimento e, para o caso exemplificado, obtém-se a cadeia de módulos representada na figura 10.

3.1.3 Interpretação

Depois de obtida uma representação de um determinado objecto para modelação, é necessário interpretar essa descrição de forma a gerar o segmento de *scenograph* correspondente, em formato X3D.

Nesta fase o utilizador tem a liberdade de definir a forma como cada um dos módulos paramétricos é convertido para X3D.

Cada módulo paramétrico é substituído por um segmento do *scenograph* em que determinados parâmetros deste gráfo são instanciados pelos argumentos do módulo. Assim, a cadeia de módulos final é interpretada como uma estrutura hierárquica em forma de árvore, em que o primeiro módulo (mais à esquerda) é a raiz, e o último módulo uma folha. De forma a permitir a criação de

“ramos” paralelos, existe um conjunto de módulos adicionais (representados pelos símbolos [e]) que os delimitam.

```
[Translação(x,y,z)
LOD(0,0,0,range)
Grupo
  [Predio(mat,cont,alt,perim)]
  [Translacao(x,y,z)
    [Telhado(tipo,cont,perim)]
    [Caleira(mat,cont3D,perim)]
  ]
  [Rebordo(mat,cont3D,perimetro)]
  [Jan(x,y,z,a,m)] [Jan2(x,y,z,a,m)]
  [Jan(x,y,z,a,m)] [Jan(x,y,z,a,m)]
  [Varan(x,y,z,a,m)] [Varan(x,y,z,a,m)]
  [Varan(x,y,z,a,m)] [Varan2(x,y,z,a,m)]
  [Jan2(x,y,z,a,m)] [Jan2(x,y,z,a,m)]
  [Jan2(x,y,z,a,m)] [Jan(x,y,z,a,m)]
  [Jan2(x,y,z,a,m)] [Jan(x,y,z,a,m)]
  [Jan2(x,y,z,a,m)] [Jan2(x,y,z,a,m)]
  [Varan2(x,y,z,a,m)] [Varan(x,y,z,a,m)]
  [Varan(x,y,z,a,m)] [Varan(x,y,z,a,m)]
  [Varan2(x,y,z,a,m)] [Varan(x,y,z,a,m)]
  [Varan2(x,y,z,a,m)] [Varan2(x,y,z,a,m)]
  [Jan(x,y,z,a,m)] [Jan2(x,y,z,a,m)]
  [Jan(x,y,z,a,m)] [Jan2(x,y,z,a,m)]
  [Jan(x,y,z,a,m)] [Jan(x,y,z,a,m)]
  [Jan2(x,y,z,a,m)] [Jan(x,y,z,a,m)]
  [Jan(x,y,z,a,m)] [Jan2(x,y,z,a,m)]
  [Jan(x,y,z,a,m)] [Jan(x,y,z,a,m)]
  [Porta(x,y,z,a,m)]
PrédioSimples(mat,cont,altura)]
```

Figura 10 - Cadeia final representando um edifício

Prusinkiewicz foi um dos pioneiros ao nível da utilização dos sistemas L na Computação Gráfica e, num dos seus trabalhos [Prusinkiewicz86], introduziu um método de interpretação baseado na metáfora da “tartaruga” utilizada na linguagem LOGO [Abelson82]. Nessa abordagem, um cursor 3D desloca-se num espaço tridimensional seguindo uma interpretação da cadeia final, através de um vocabulário restrito de símbolos que alteram o seu posicionamento e orientação. Ao longo do trajecto, vão sendo instanciadas primitivas tridimensionais.

A abordagem utilizada neste trabalho é diferente e baseia-se na interpretação de uma estrutura hierárquica em árvore representando o *scenograph*, em que cada módulo é convertido, por meio de protótipos definidos pelo utilizador, para segmentos de *scenograph* em X3D. Os protótipos baseiam-se no nó PROTO do X3D, sendo permitidas expressões para os campos.

Partindo da cadeia final que representa um edifício e após a sua interpretação por um conjunto de protótipos, obtém-se como representação final o *scenograph* em X3D representado na figura 11.

3.1.4 Refinamento incremental do modelo

A solução obtida após a aplicação inicial de um determinado sistema L nem sempre produz os melhores

resultados ou um nível de realismo suficiente para a aplicação em causa.

Nestes casos, o utilizador pode ir refinando o conjunto de regras de produção no sentido de atingir a solução desejada, ou pode ainda aumentar a informação descrita no axioma, colectando maior quantidade de dados.

O sistema é assim de refinamento incremental: permite obter rapidamente um modelo grosseiro mas coerente de uma área urbana, sendo que o utilizador pode, à medida das suas necessidades, melhorá-lo globalmente ou em detalhes particulares.



Figura 11 – Modelo final do edifício em X3D

3.2 Especificação de Modelação (XL3D)

Todo o processo de modelação é especificado de forma declarativa através de um documento baseado num XML-Schema definido especificamente para este sistema, denominado XL3D [Coelho03].

A estrutura do documento é hierárquica e a raiz do documento é o projecto XL3D (*XL3Dproject*, figura 12). Cada documento contém um único projecto, sendo este constituído por um cabeçalho, modelos, procedimentos de modelação, protótipos e fontes de dados.

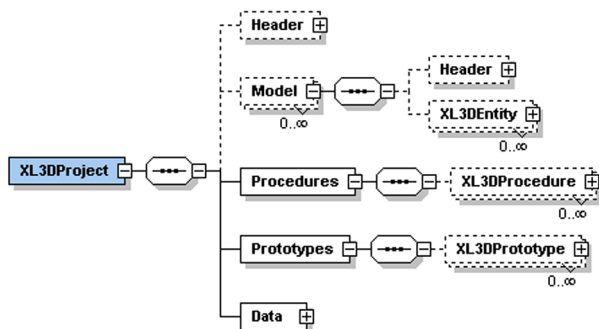


Figura 12 – Projecto XL3D

O cabeçalho (*Header*, figura 13) foi definido de forma a proporcionar informação de autoria, nomeadamente o seu título, o nome dos autores, a versão e alguma documentação. Devido à sua natureza estruturante, este elemento faz parte integrante da maioria dos restantes elementos.

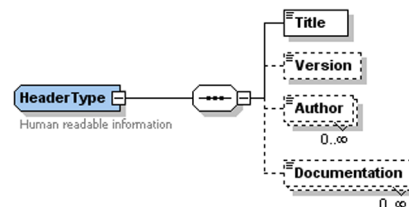


Figura 13 – Cabeçalho

Cada modelo (*Model*, figura 12) é composto por entidades, reflectindo a natureza complexa e estruturada dos ambientes urbanos, segundo uma visão subjectiva do utilizador. As entidades (*Entity*, figura 14) encapsulam um conjunto de modelos geométricos correspondentes às diversas entidades que compõem o projecto. Sendo definida numa estrutura hierárquica, uma entidade pode conter outras entidades cujo modelo pode ser instanciado em qualquer posição do *scenegraph*.

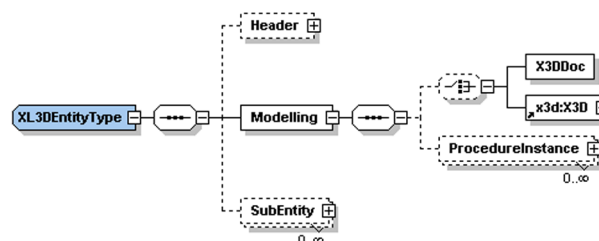


Figura 14 – Entidade

Como exemplo, um modelo de uma cidade pode ser composto por várias entidades como a rede viária, a vegetação e o edificado. A rede viária, por sua vez também poderá ser composta pelas entidades pavimento, passeios, sinalização, etc. Cada entidade instancia um conjunto de procedimentos de modelação que geram os modelos tridimensionais a partir de cadeias de módulos, obtidas de fontes de dados diversas.

O axioma é composto a partir de uma ou várias fontes de informação, sendo composto por concatenação ou aplicação de uma transformação XSLT.

Os procedimentos de modelação (*XL3DProcedure*, figura 15) constituem o componente nuclear do processo de modelação e baseiam-se em regras de produção que controlam o Sistema L que modela partes específicas de cada entidade.

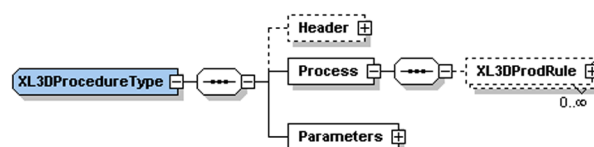


Figura 15 – Procedimento de modelação

Os procedimentos de modelação são definidos como componentes reutilizáveis que podem ser instanciados de forma a modelar distintas entidades e que podem ser estendidos para constituir novos procedimentos de modelação.

Cada procedimento de modelação possui assim um conjunto de regras de produção que transformam uma cadeia inicial de módulos, denominada axioma. As regras de produção (*XL3Dproduction*, figura 16) seguem a fórmula representada em (1) relativas a sistemas L sensíveis ao contexto, paramétricos e estocásticos.

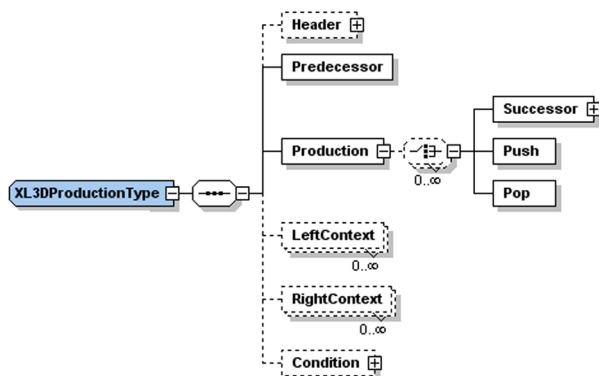


Figura 16 – Regra de produção

A cadeia de módulos resultante é interpretada de forma a criar um *scenegrph*, através da instanciação de um conjunto de protótipos.

Os protótipos (*Prototype*, figura 17) são segmentos de um grafo de cena X3D que podem ser instanciados através da interpretação da cadeia final de módulos obtida dos procedimentos de modelação.

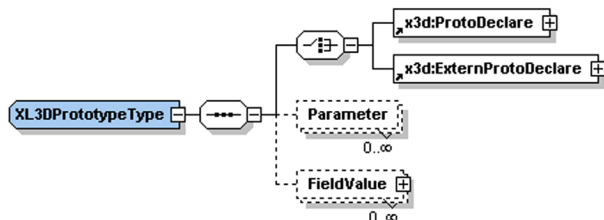


Figura 17 – Protótipo

As fontes de dados (*DataSources*, figura 18) contêm referências a dados provenientes de ficheiros XML. Os dados são convertidos do seu formato original para uma cadeia de módulos de informação através das transformações XSLT.

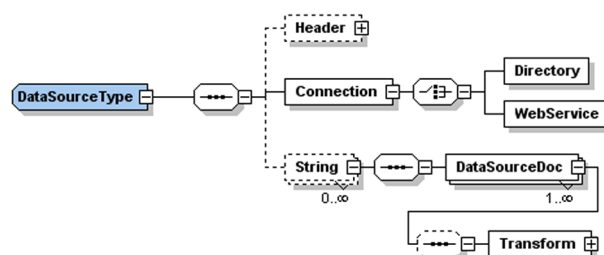


Figura 18 – Fonte de dados

4. RESULTADOS OBTIDOS

Ao longo da secção 3.1 exemplificou-se o processo de modelação de um elemento de edificado partindo de dois módulos. Após o processo de modelação, obteve-se uma cadeia final representando um edifício, composta por 46 módulos. Depois da sua interpretação foi gerado um ficheiro X3D com 277 nós. Demonstra-se assim o poder de ampliação de dados que este sistema possui, bem como a capacidade de emersão, ao gerar um objecto tão complexo como um edifício, partindo de algo tão simples como um contorno geométrico e o número de pisos. Existe efectivamente uma mais valia, quando comparamos com o processo de modelação manual, em que seria necessário posicionar, orientar e caracterizar cerca de 45 elementos distintos para a construção deste edifício.

O processo, que foi aplicado a um só elemento para exemplificação, pode ser aplicado, da mesma forma, a axiomas contendo grande quantidade de elementos, o que permite gerar cenas urbanas extensas sem qualquer esforço acrescido por parte do utilizador. No entanto, o processo de modelação manual obrigaria o utilizador a modelar individualmente cada um dos edifícios, aumentando o esforço envolvido de forma proporcional ao número de edifícios. Utilizando grupos de procedimentos de modelação e associando-os a vários axiomas representando informação diversificada, conseguem-se modelar os diferentes elementos que estruturam e compõem um ambiente urbano.

Os processos de modelação especificados para um determinado caso podem facilmente ser reutilizados para serem aplicados a outros com características semelhantes, estendendo o conjunto das suas regras de produção.

As figuras 19 e 20 exemplificam a potencialidade do sistema desenvolvido para a modelação de ambientes urbanos, para a geração de forma automática de soluções iniciais com grau de realismo aceitável.

O modelo foi gerado a partir da informação contida num sistema de informação geográfica, sendo a informação geográfica (figura 21), disponível à escala 1:2000, estruturada segundo 3 temas: espaços verdes, rede viária e edificado.

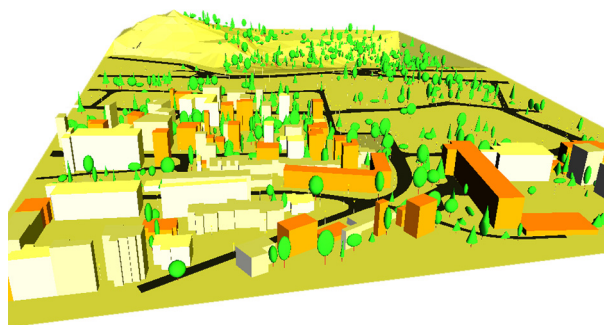


Figura 19 – Modelo virtual de ambiente urbano



Figura 20 – Vista de pormenor do modelo virtual



Figura 21 - Informação geográfica disponível

5. CONCLUSÕES E TRABALHO FUTURO

O sistema de modelação desenvolvido pode ser utilizado com grandes vantagens na geração de modelos tridimensionais de zonas urbanas reduzindo, quer os custos, quer a duração do processo.

O modelo obtido pode ser incrementalmente melhorado, através do refinamento das regras que controlam o processo de modelação, bem como através da adição de mais informação.

No caso específico de alguns elementos complexos do modelo, como é o caso de monumentos ou edifícios emblemáticos, o sistema não garante facilmente o nível de realismo desejado, pelo que estes devem ser modelados de forma cuidada em outro tipo de modeladores e posteriormente integrados no processo de modelação do sistema.

O sistema pode produzir resultados de elevado nível de detalhe e realismo, sempre que a informação disponível seja vasta e de qualidade. No entanto, mesmo nos casos em que tal informação é escassa, o sistema é capaz de chegar a uma solução inicial que, embora possua menor nível de realismo, mantém um nível de detalhe aceitável por via da incorporação de alguma aleatoriedade.

A definição do conjunto de regras de produção para a modelação de uma determinada entidade não é trivial, sendo um processo minucioso e iterativo. No entanto, a utilização de procedimentos de modelação paramétricos,

bem como a facilidade com que se consegue estender um determinado processo para modelar outra entidade, obvia essa dificuldade, possibilitando a reutilização de muitos dos processos de modelação.

Em termos de trabalho futuro, é essencial criar uma ferramenta de apoio à especificação do projecto de modelação, com especial ênfase na criação dos conjuntos de regras de produção que formam os diversos processos de modelação.

Pretende-se ainda implementar este sistema como um serviço Web, de forma a estar disponível para integração com outras aplicações, numa arquitectura distribuída.

Como exemplo da relevância desta arquitectura, está a ser desenvolvido um sistema móvel de navegação em tempo real [Pinto03] onde se pretendem integrar modelos tridimensionais, do trajecto e da envolvente. Dado que estes trajectos são determinados em tempo real em resposta a um pedido do utilizador, os modelos têm que ser gerados dinamicamente a partir daquele serviço Web.

Outro desenvolvimento possível é a integração de outros serviços Web, como fonte de dados para o processo de modelação. Dados os avanços na especificação de serviços Web para disponibilização de informação geográfica, como o WFS [WFS], o acesso ao resultado de interrogações avançadas à base de dados e de operações espaciais ficaria simplificado.

Também ao nível da interacção com o utilizador se pretende desenvolver um conjunto de ferramentas que permitam a geração dos ficheiros XL3D de forma intuitiva, bem como a integração no processo de informação raster proveniente de imagens captadas sobre o modelo real.

6. REFERÊNCIAS

- [Abelson82] Abelson H., diSessa A.: *Turtle geometry*, M.I.T. Press, Cambridge, 1982.
- [Coelho03] Coelho A., Sousa A., Ferreira F.: 3D Modelling of Large Urban Scenes from Diverse Sources of Information. *Proceedings of the 7th ICCG/IFIP International Conference on Electronic Publishing*, pp. 278-287, 2003.
- [Dachselt02] Dachselt R., Hinz M., Meißner K.: CONTIGRA: An XML-Based Architecture for Component-Oriented 3D Applications, In *Proceedings of Web3D '02*, pp. 155-163, 2002.
- [Deussen98] Deussen O., Hanrahan P., Lintermann B., Mech R.: Realistic modelling and rendering of plant ecosystems. *Proceedings of SIGGRAPH 98*, 1998.
- [Dodge98] Dodge M., Doyle S., Smith A., Fleetwood S.: Towards the Virtual City: VR & Internet GIS for Urban Planning. Paper presented at the *Virtual Reality and Geographical Information Systems Workshop*, London, 1998.
- [GML] GML - Geography Markup Language. <<http://www.opengis.net/gml/>>

- [Lane02] Lane B., Prusinkiewicz P.: Generating spatial distributions for multilevel models of plant communities. *Proceedings of Graphics Interface 2002*, pp. 69-80, 2002.
- [Lindenmayer68] Lindenmayer, A.: Mathematical models for cellular interaction in development, Parts I and II. *Journal of Theoretical Biology*, 18, pp. 280-315, 1968
- [OpenGIS] The OpenGIS Consortium, Inc.
<<http://www.opengis.org/>>
- [Parish01] Parish Y., Muller P.: Procedural Modeling of Cities. *Proceedings of SIGGRAPH 2001*, pp. 301-308, 2001.
- [Pimentel01] Pimentel J., Baptista N., Goes L., Dionísio J.: Construção e gestão da complexidade de cenários urbanos 3D em ambientes virtuais imersivos. *Actas do 10º Encontro Português de Computação Gráfica*, pp. 165-174, 2001.
- [Pinto03] Pinto P., Serôdio C., Coelho A., Mestre P., Soares S., Amaro C., Ferro P.: Audio Interface for Real Time Navigation. *Proceedings of the 4th Conference in Telecommunications*, pp. 337-341, 2003.
- [Prusinkiewicz01] Prusinkiewicz P., Muendermann L., Karwowski R., Lane B.: The use of positional information in the modeling of plants. *Proceedings of SIGGRAPH 2001*, pp. 289-300, 2001.
- [Prusinkiewicz86] Prusinkiewicz P.: Graphical applications of L-systems. In *Proceedings of Graphics Interface '86 – Vision Interface '86*, pp. 247-253, 1996.
- [Prusinkiewicz90] Prusinkiewicz P., Lindenmayer A.: *The Algorithmic Beauty of Plants*, Springer, 1990.
- [Prusinkiewicz95] Prusinkiewicz P., Hammel, M., Mech R., Hanan J.: *The Artificial Life of Plants. From Artificial life for graphics, animation and virtual reality*, volume 7 of SIGGRAPH '95 Course Notes, pages 1-1-1-38. ACM Press, 1995.
- [Smith84] Smith A.: Plants, fractals and formal languages. *Proceedings of SIGGRAPH '84 in Computer Graphics*, 18, 3, pp. 1-10, ACM SIGGRAPH, New York, 1984.
- [Taylor92] Taylor, C.: “Fleshing out” Artificial Life II. In C. G. Langton, C. Taylor, J. D. Farmer and S. Rasmussen, editors, *Artificial Life II*, pp. 25-38, Addison Wesley, Redwood City, 1992.
- [VRML97] The VRML Consortium Inc.: “The Virtual Reality Modelling Language – International Standard ISO/IEC 14772-1:1997”.
http://www.web3d.org/technicalinfo/specifications/ISO_IEC_14772-All/index.html
- [Web3D] The Web3D Consortium.
<http://www.web3d.org/>
- [WFS] Web Feature Service
<http://www.opengis.org/techno/RFC13.pdf>
- [X3D] Web3D Consortium: “X3D: The Virtual Reality Modelling Language – International Standard ISO/IEC 14772:200x”.
<http://www.web3d.org/TaskGroups/x3d/specification/>
- [XML] XML - Extensible markup Language.
<<http://www.w3.org/XML/>>
- [XML-Schema] XML-Schema.
<<http://www.w3.org/XML/Schema>>
- [XSLT] XSLT- Extended Stylesheet Language Transformations (XSLT).
<<http://www.w3.org/TR/xslt11/>>

Identificação e reconstrução 3D de objectos numa malha urbana através de nuvens de pontos georeferenciados

Rodrigo Garcia¹, Mauro Farracha¹, José Dionísio², Alexandre Bernardino³, João Paulo Costeira³

Instituto de Sistemas e Robótica^{1,3}, IDMEC²

Instituto Superior Técnico

Av. Rovisco Pais, 1049-001 Lisboa

{rjsdg,mdfa}@mega.ist.utl.pt¹, jdionisio@ist.utl.pt², {alex,jpc}@isr.ist.utl.pt³

Resumo

Este artigo descreve algumas técnicas utilizadas no âmbito da detecção e reconstrução tridimensional de objectos numa malha urbana através de uma nuvem de pontos georeferenciados. Os dados de entrada foram obtidos através de varrimento laser sobre a baixa da cidade de Lisboa e são apresentados numa malha rectangular de 1m². Dada a especificidade dos dados, o trabalho desenvolvido focou primordialmente a detecção de objectos a partir da construção do DSM (Digital Surface Model). Uma vez determinados os limites dos objectos (maioritariamente edifícios), através da estimação de parâmetros característicos, como por exemplo o gradiente de altura, foi gerado o pipeline de reconstrução dos mesmos e consequente processamento tendo em vista a sua visualização em ambientes virtuais interactivos.

Palavras Chave

Varrimento Laser, DSM, Crescimento de Regiões, “Best Fit Ellipse”, Reconstrução 3D, Ambientes Virtuais Interactivos, Visualização em tempo real

1. INTRODUÇÃO

Durante os últimos anos, o aumento da necessidade de projecto e planeamento de áreas urbanas e a sua consequente actualização, levou a esforços de investigação com o objectivo de construir ferramentas automáticas ou semi-automáticas para a aquisição e validação de dados cadastrais. Uma das áreas com interesse nestas ferramentas é por exemplo a do Urbanismo, que pretende saber o que na realidade existe em determinado espaço urbano e qual o impacte que poderia ter uma modificação neste [9].

A automação da criação de modelos de cidades, requeridos por muitos utilizadores de sistemas de informação geográfica, juntamente com os levantamentos da componente fotométrica, permitiu nos últimos anos, a validação de algoritmos para aproximações 3D, como por exemplo, o agrupamento de características retiradas pela análise de múltiplas imagens aéreas de pontos com informação sobre o solo. Devido à sua vantagem como uma técnica activa para a determinação de pontos 3D, o varrimento laser aéreo tornou-se assim uma fonte de informação importante para a criação de modelos altimétricos complexos.

É neste contexto que se descreve o trabalho realizado, no sentido de desenvolver uma metodologia de análise deste tipo de dados e que proporcione a reconstrução de uma cena virtual 3D com capacidade de edição interactiva, visualização fotorealista e navegação em tempo real.

2. DESCRIÇÃO DO PROBLEMA

O problema a resolver consiste em diversas fases sequenciais fundamentais: em primeiro lugar a construção de um *pipeline* que permita avaliar, identificar, segmentar e visualizar um cenário 3D que melhor represente a área urbana correspondente aos dados recolhidos a partir de dados altimétricos do solo, obtidos por varrimento laser. Finalmente, uma vez obtido o novo modelo, pretende-se desenvolver uma aplicação que permita a sua comparação com modelos previamente existentes, quer sejam 2D (áreas de implantação) ou 3D (*bouding boxes* com os valores de cêrcea).

3. ESTADO DE ARTE

Exposto o problema, realizou-se uma pesquisa para analisar com maior profundidade o trabalho já desenvolvido nesta área. De entre os vários métodos que

abordaram o problema da detecção e reconstrução de objectos a partir de pontos altimétricos, destacam-se um par de contribuições pela sua complementar abordagem.

Antes de mais, uma ressalva para explicitar as siglas DSM (*Digital Surface Model*) e DTM (*Digital Terrain Model*). O DSM, ou modelo digital da superfície, é a representação para os pontos originais obtidos a partir de varrimento laser, é indexado por x e y contendo os valores em z para cada par (x,y) (Fig. 1). Pode ver-se que as áreas com um tom mais claro são os pontos com altura maior. O DTM tem o mesmo esquema de representação do DSM mas corresponde ao modelo digital da elevação do terreno do DSM.



Fig. 1: Exemplo de um DSM representado em tons de cinzento

Em relação ao levantamento bibliográfico efectuado, foram escolhidas como referência inicial os trabalhos de Ansgar Brunn e Uwe Weidner 1998 [8], por serem frequentemente citados, terem desenvolvido trabalho relevante e recente, e baseado em princípios básicos de processamento de DSMs (Digital Surface Models) e DTMs (Digital Terrain Models). A forma como descreveram a sua investigação pode subdividir-se nas três fases convencionais, como de seguida se apresenta:

- detecção de características padrão (DSM)
- segmentação do domínio (DSM)
- reconstrução 2D e 3D

mas os métodos utilizados em cada fase demonstram pressupostos diferentes.

Em primeiro lugar, geram o DSM normalizado consistindo na diferença entre o DTM e DSM da mesma área urbana. Se imaginarmos uma área urbana contendo edifícios no vale e no pico de uma montanha, e se gerarmos o DSM normalizado, então o valor da elevação da montanha será retirada e os edifícios encontrar-se-ão, consequentemente, ao mesmo nível. Este processo facilita o processamento da fase seguinte, a segmentação.

A mesma equipa realizou uma segmentação binária relativamente à altura, obtendo assim os objectos com altura significativa utilizando um “threshold” (Fig. 2). Desta segmentação vários segmentos são escolhidos como possíveis representações de edifícios (segmentos baseados em altura local). O problema desta escolha é o

facto de existir vegetação e que influencia negativamente a segmentação dos edifícios. A solução adoptada passou por calcular a rugosidade da superfície através da variância das normais à superfície, grau de curvatura local a um ponto e gradientes (Fig. 3). As áreas detectadas são isoladas e são subtraídas ao DSM normalizado (Fig. 4).



Fig. 2: Resultado da segmentação do DSM da Fig. 1



Fig. 3: Segmentação após o cálculo das rugosidades do DSM



Fig. 4: Aspecto visual do mapa de diferenças

Agora a segmentação binária descrita pode ser realizada novamente e todos os segmentos são fidedignos aquando da sua selecção para descreverem edifícios. O problema desta segmentação é efectivamente a utilização de um “threshold” fixo já que obriga a um processo iterativo. A possível solução apresentada para este caso pelos mesmos autores é a utilização de redes Bayesianas.

As redes Bayesianas não efectuam a segmentação através de um “threshold” fixo mas sim com a utilização de características. Para cada ponto é calculada a probabilidade de ele pertencer a um edifício através do valor da característica em análise. Pode entender-se como uma característica a altura do ponto, a curvatura local, o gradiente nesse ponto, ou outros conceitos que permitam caracterizar pontos. A rede é construída e são calculados os nós com maior probabilidade e que através de métodos precisos de cálculo podem ser agrupados por semelhança. A grande desvantagem desta solução é o enorme tempo de cálculo para o processamento da rede.

Finalmente, aproximamo-nos da terceira fase convencional: após a detecção dos segmentos que descrevem os edifícios, há que desenvolver os algoritmos de reconstrução 3D. Para tal, a mesma equipa utiliza modelos prismáticos e paramétricos de edifícios e o trabalho consiste em analisar os segmentos detectados na primeira fase e com eles estimar os parâmetros dos modelos. Estes modelos não contêm parâmetros para os telhados dos edifícios e basicamente o método utilizado baseia-se na estimação dos superfícies planares através dos pontos do DSM normalizado. Os planos são intersectados e é obtido o modelo aproximado do telhado.

È neste ponto que começam realmente a ser interessantes as diversas abordagens em cada uma das fases.

A primeira variação fundamental surge numa aplicação apresentada por *Steinle e Vögtte* [7] que falam sobre a detecção de mudanças estruturais em edifícios após um desastre natural, no caso, um terramoto. Neste cenário, são obtidos dois modelos da área urbana, antes do desastre e pós desastre. É realizada então uma comparação e são identificados os edifícios com maior risco de abaterem e provocarem danos adicionais. No seu trabalho identificam a possibilidade de reconhecimento de objectos a partir de cinco análises distintas: sinal espectral, altura acima do chão, altura da textura, vectores normais e pela forma e contorno dos objectos:

Sinal espectral - apenas o método que utiliza o sinal espectral não recorre ao DSM e é reconhecido pela equipa de trabalho como o melhor método para o reconhecimento de objectos;

Altura acima do chão - semelhante ao apresentado anteriormente por *Brunn e Weidner* [8];

Altura da textura - utilizando os métodos de definição da altura da textura de *Maas* [1][2][3]. São suficientemente

precisos para conseguir separar a vegetação de edifícios e conseguir detectar estes últimos;

Vectores normais - consiste em detectar no DSM áreas homogéneas, ou seja, numa área plana os vectores normais são semelhantes enquanto que numa área de vegetação os vectores normais variam bastante, formando uma área bastante heterogénea relativamente à direcção dos vectores normais.

Forma e contorno de objectos [4][5] - os pontos são ligados através de uma rede triangular e pequenos triângulos são formados e juntos. Posteriormente é realizada uma suavização que permite obter o modelo estimado do edifício. Para a sua reconstrução 3D, é estimada a altura do edifício, são estimadas as superfícies planares 3D dos telhados a partir dos pontos do objecto detectado e são intersectados estes planos para obtenção do modelo final.

4. IMPLEMENTAÇÃO

Como já foi referido, é comum utilizar a seguinte sequência de processamento de nuvens de pontos altimétricos: construção do DSM a partir dos pontos obtidos pelo laser, geração de uma primeira segmentação binária para separar os objectos de interesse, remoção selectiva de vegetação e, finalmente, a reconstrução geométrica, prismática ou 3D. Neste trabalho adopta-se esta orientação, todavia, apresentam-se variações relevantes ao objecto de estudo e implementação neste trabalho. Com especial relevo, destacam-se:

- a) a segmentação binária através de um algoritmo de crescimento de regiões ou “Region Growing” que admite elevações do terreno, logo dispensando a utilização do DTM para a segmentação inicial;
- b) após a obtenção das regiões de interesse pelo método exposto em a), estimam-se as paredes dos edifícios através da implementação do algoritmo “Best Fit Ellipse” que reduz a computação feita em relação aos algoritmos da secção 2;
- c) para a detecção da vegetação, introduz-se nova componente inovadora: alguns conceitos matemáticos como a Entropia e a Variância dos vectores normais.

As várias abordagens serão objecto de estudo e exemplificação nas secções seguintes.

4.1 Caso de estudo: baixa de Lisboa

O DSM (Digital Surface Model) é a base de partida para o desenvolvimento deste trabalho, com o objectivo de processar o modelo DSM, detectar os objectos nele contido e a reconstruir um cenário com superfícies planares minimizando o erro por superfície (Fig. 5).

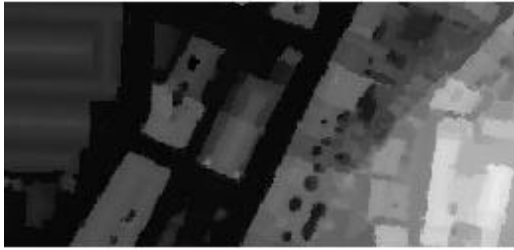


Fig. 5: Exemplo do DSM parcial da baixa de Lisboa (os tons brancos são os que contêm as alturas mais altas e os tons mais escuros as alturas mais baixas)

O ruído inerente á aquisição de dados, não perceptível pela visualização a olho nú da nuvem de pontos, aparece explícito aquando da sua representação em mapa de alturas (Fig. 6).

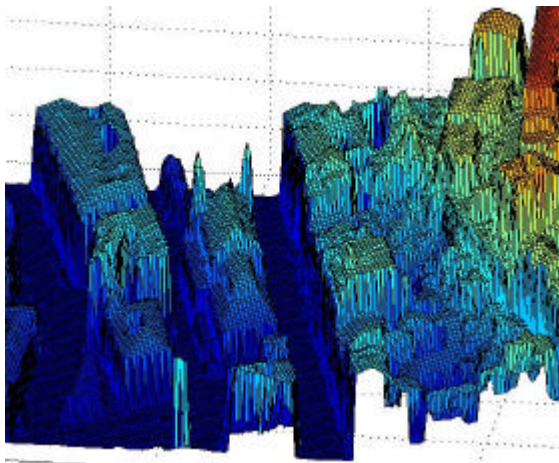


Fig. 6: Representação parcial do mapa de alturas da baixa de Lisboa

O trabalho desenvolvido neste contexto pode ser representado da seguinte forma (Fig. 7):

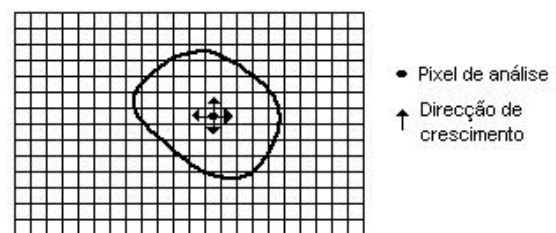


Fig. 7: Etapas do processamento do DSM

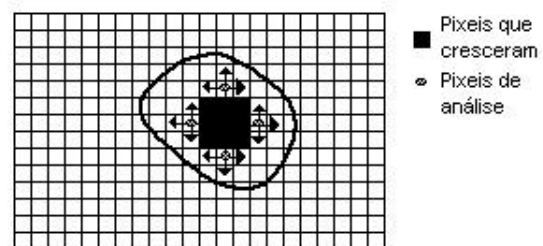
A primeira fase, consiste em utilizar o DSM para detectar regiões nele contidas que sejam homogêneas em alguma característica. Neste caso foram usados métodos com características particulares para detectar estas regiões, como por exemplo, áreas de altura semelhante, com gradientes semelhantes, com ângulos de gradientes semelhantes, entre outros.

O algoritmo base utilizado é o “Region Growing” que analisa o DSM e baseado ora nos gradientes, ora no ângulo destes, ou na altura, compara ponto a ponto e reúne os pontos em grupos que tenham estas características em comum. Esta é uma aproximação “bottom up” que reúne pontos até que todos pertençam a alguma região. Imagine-se que o cenário da Fig. 8 representa uma região ainda não detectada por este algoritmo. A linha limite preta representa a região que se quer obter no final e os pixels no seu interior têm em comum uma mesma característica, por exemplo, a altura. O algoritmo consiste em partir de um ponto de análise e comparar em ciclo os seus pontos vizinhos. Caso os seus vizinhos tenham a característica de análise em comum são adicionados à região em crescimento. As setas indicam quais os vizinhos a analisar e quais as direcções de crescimento:

1) Início do crescimento da região



2) Passo do crescimento



3) Resultado Final

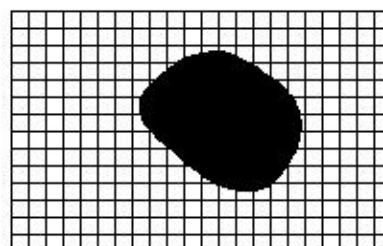


Fig. 8: Resultados gráfico de “Region Growing”

No final obtêm-se regiões que se podem identificar como os objectos principais validados. A Fig. 9 contém todos os objectos a branco que foram reconhecidos.

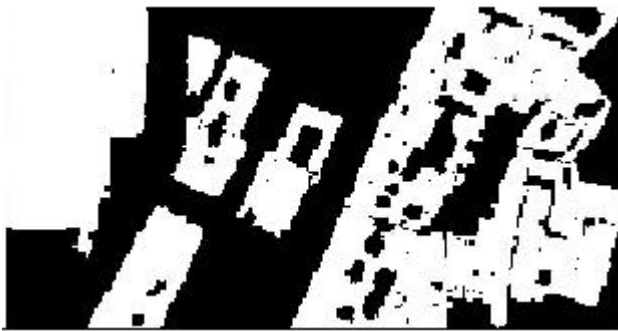


Fig. 9: Algoritmo “region growing” aplicada à baixa de Lisboa

Um caso particular, o “Region Growing” detecta o chão como um objecto, mas é fácil a sua remoção dos objectos eleitos. O facto das regiões brancas parecerem contínuas, pode causar a sensação de formarem uma única região mas, na verdade, o algoritmo garante que os objectos estão bem segmentados e individualizados.

O passo seguinte à detecção de regiões é a estimação de parâmetros das regiões. Este passo vai permitir estimar todos os parâmetros que no futuro serão estruturados para a fase de reconstrução, como por exemplo, quantas paredes tem o edifício, qual a sua altura, a forma do telhado, etc... Nesta fase de estimação, o primeiro passo efectuado foi a obtenção dos limites das regiões detectadas anteriormente com o intuito de poder para cada uma delas estimar as paredes laterais (Fig. 10).



Fig. 10: Exemplificativo gráfico para um dos cenários de estimação de parâmetros

Para obter estes limites fez-se uma aproximação à derivada do DSM para obter o máximo local do gradiente em cada ponto. Obtidos estes pontos limites o objectivo primário passa a ser a análise desses pontos e a sua segmentação tendo em vista a obtenção das linhas das paredes. Estas linhas serão estimadas não só em direcção e comprimento mas também em altura (relativa aos pontos limite que fazem parte da linha estimada).

A solução criada para conseguir, a partir de um conjunto de pontos, a sua subdivisão em grupos que estejam aproximadamente na mesma linha, é o algoritmo “Best Fit

Ellipse”. Este algoritmo consiste na estimação de uma ellipse envolvente aos pontos e que tenha um dos seus eixos o mais pequeno possível. Assim, quando um ponto faz com que o eixo salte de um valor muito pequeno para um valor maior, a linha é quebrada e continua-se a estimar uma nova ellipse com os pontos seguintes.

Imagine-se um conjunto de pontos x, y do plano cujos extremos mais distantes são $(x_1, y_1), (x_2, y_2)$ (Fig. 11).

O “Best Fit Ellipse” calcula a maior distância d que tem de haver para a ellipse conseguir envolver todos os pontos (x, y) entre (x_1, y_1) e (x_2, y_2) . Para cada ponto deste tipo e através das seguintes formulas, deduzidas a partir da Fig. 11, pode calcular-se o valor da distância d :

$$b^2 = (a^2 d^2 \sin^2 \tau) / (a^2 - d^2 \cos^2 \tau)$$

$$\cos \tau = (a^2 + d^2 - f^2) / (2ad)$$

$$x_0 = (x_1 + x_2) / 2$$

$$y_0 = (y_1 + y_2) / 2$$

$$a = [(x_2 - x_1)^2 + (y_2 - y_1)^2]^{1/2} / 2$$

$$a = \text{atan}[(y_2 - y_1) / (x_2 - x_1)]$$

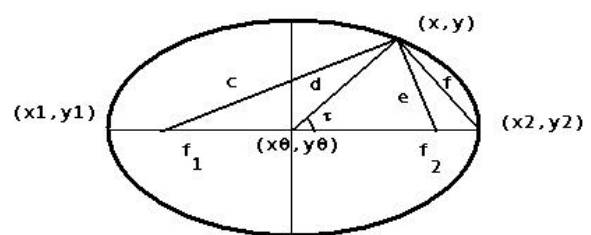


Fig. 11: Representação gráfica do “best fit ellipse”

Legenda: (x_0, y_0) é o centro da ellipse, f_1 e f_2 os focos da mesma.

Caso exista algum ponto entre (x_1, y_1) e (x_2, y_2) que obrigue d a ser maior que um valor T ($T \sim 2$), então estaremos perante o caso de quebra de linha ou seja, provavelmente será um canto de edifício e, desta forma, é possível começar a definir uma parede. Quando uma parede é detectada através deste algoritmo, estima-se também a sua altura média, comprimento e direcção, tendo em vista a sua reconstrução posterior em 3D. O resultado de uma estimação com este algoritmo a um grupo de pontos limite de uma região está representado na Fig. 12.

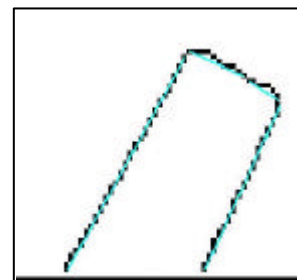


Fig. 12: Representação gráfica dos resultados do método da ellipse para formação do perímetro do edifício

Como se trata essencialmente de objectos em áreas urbanas, o objectivo primordial foca agora os edifícios que ocupam as maiores áreas em volume. Assim, no futuro, haverá vantagens em aperfeiçoar esta estimação de parâmetros só para regiões deste tipo.

Em relação à componente vegetação, foram implementados algoritmos que calculam os vectores normais dos pontos do DSM e através destes a variância (Fig. 13) e entropia (Fig. 14) para posterior refinamento da segmentação das regiões detectadas (p.ex., vegetação onde a variância e entropia têm valores muito elevados).



Fig. 13: Representação gráfica de resultados para o espectro de variâncias

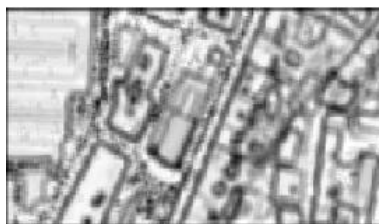


Fig. 14: Representação gráfica de resultados para o espectro de entropia

A entropia é aplicada sobre as componentes Z dos vectores normais à superfície, obtendo-se as matrizes das componentes x, y e z. Aplicando a entropia para esta última componente é calculado, segundo uma janela de NxN pixels sobre a matriz da componente Z, um histograma de diferenças de valores presentes. Normalizando este histograma obtêm-se as probabilidades de cada valor. A entropia de uma região é calculada segundo a seguinte fórmula:

$$E = \sum_i -P(h_i) \times \log(P(h_i))$$

onde h_i são os valores do histograma da janela. Estes pontos poderão ser removidos do DSM para melhorar a qualidade de estimação dos parâmetros.

Finalmente, no seguimento do organograma, falta apenas a fase de estruturação dos parâmetros. Pretende-se construir uma hierarquia de objectos e atributos estruturada e compatível com os requisitos para futura visualização em tempo real. A hierarquia é construída a partir da fase de estimação de parâmetros e pode representar-se da seguinte forma (Fig. 15).

No início deste trabalho pensou-se em estruturar as regiões homogêneas em bairros, ruas, etc..., mas após análise e obtenção dos primeiros resultados constatou-se que não seria fácil arranjar padrões de regiões. Às diferenças devido às reais diferenças altimétricas há a somar o ruído que foi introduzido aquando da obtenção por varrimento laser dos pontos originais. Este ruído introduz erros que influenciam os algoritmos e não facilitam a padronização por grupos de regiões (Fig. 16).

A hierarquia apresentada é alterada sempre que o grupo necessite de estimar mais algum parâmetro de um edifício em particular. Como podemos ver o resultado de toda a fase de detecção serão planos 3D bem definidos que a fase de reconstrução se limitará a representar no ecrã (Fig. 17).

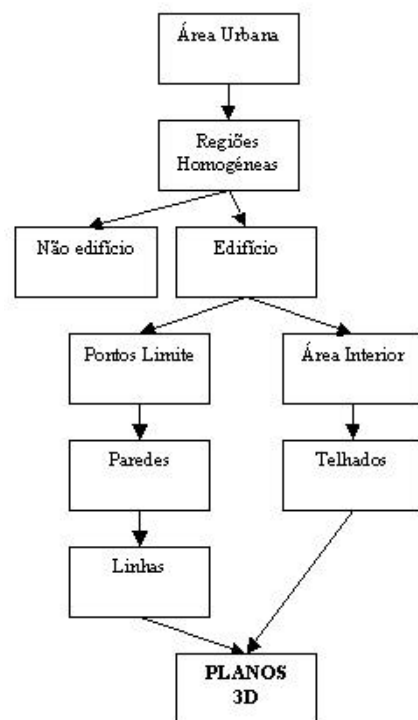


Fig. 15: Organograma dos procedimentos para reconstrução tridimensional



Fig. 16: Representação pré-prismática

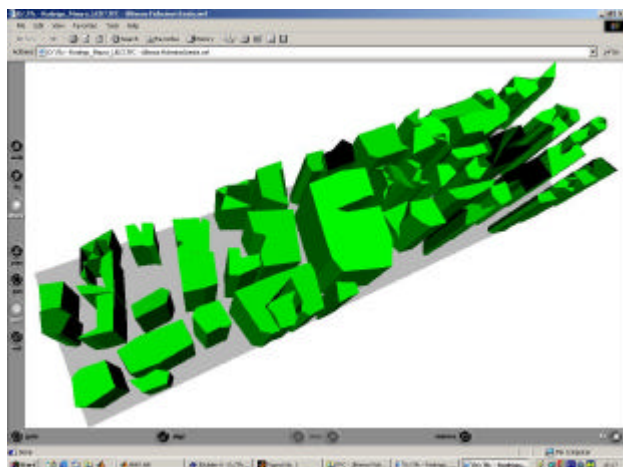


Fig. 17: Representação 3D constituída somente por superfícies planares

5. CONCLUSÕES

Os resumos dos artigos apresentados são a referência em relação ao *pipeline* convencional e contribuíram para a compreensão do problema e introdução dos conceitos gerais desta área.

Além do aperfeiçoamento dos algoritmos apresentados para a estimação dos parâmetros apresentados, o grupo ainda tem de implementar a parte de estimação de telhados e definição dos parâmetros em planos 3D para a reconstrução. Este processo está ainda em fase de processamento e não será incluído neste artigo. Todavia, a estimação dos telhados será feita através da selecção dos pontos interiores das regiões detectadas. Esses pontos forneceram a possibilidade de estimar os planos que se ajustam mais ao perfil dos telhados. Após intersectar esses planos obtemos a forma final do telhado.

Conclui-se que grande parte do trabalho desenvolvido é centrado na vertente de detecção de objectos, ficando a reconstrução 3D muito dependente da qualidade dos resultados obtidos. É, pois, fundamental que uma das vertentes fortes deste artigo seja o aprofundamento de alguns destes algoritmos com as necessárias variações.

Continua ainda o processo de procura da melhor solução para detectar e representar realisticamente as entidades que correspondem à vegetação.

Um processo eficaz de reconhecimento de objectos permitirá uma mais exacta hierarquização do cenário 3D

(casas isoladas, quarteirões, bairros, entre outros) com a consequente vantagem na vertente edição ou navegação interactivas.

6. REFERÊNCIAS

- [1] Vosselman G., Suveg I. *Map based building reconstruction from laser data and images*, Proceedings of Automatic Extraction of Man- Made Objects from Aerial and Space Images (III), Ascona 11 - 15 Junho 2001, Balkema Publishers, ISBN 9 589 252 6, pp. 231- 242
- [2] Vosselman G. *3D Building model reconstruction from point clouds and ground plans*, *Int. Archives of Photogrammetry and Remote Sensing*, XXXIV-3/W4:37-43, 2001
- [3] Vosselman G., Maas H. *Two algorithms for extracting building models from raw laser altimetry data*, *ISPRS Journal of Photogrammetry & Remote Sensing*, 54(2-3):153-163, 1999
- [4] Haala N., Brenner C. *Extraction of buildings and trees in urban environments*, *ISPRS Journal of Photogrammetry & Remote Sensing*, 54(2-3):130-137, 1999
- [5] Haala N. *Combining Multiple Data Sources for Urban Data Acquisition*, *Institut für Photogrammetrie, Proc. Photogrammetrische Woche 1999*, Fritsch/Spiller (Eds.)
- [6] Rieger K, Reiter R. *Roads and buildings from laser scanner data within a forest enterprise*, *Int. Archives of Photogrammetry and Remote Sensing*, Vol. XXXII
- [7] Steinle E., Vögtte T. *Automated extraction and reconstruction of buildings for disaster management*, E. Baltasvias et.al. (eds.): *Automatic Extraction of Man-Made Objects from Aerial and Space Images (III)*, Swets & Zeitlinger, Lisse, The Netherlands, 2001, ISBN 90 5809 252 6, pp. 309 - 318
- [8] Brunn A., Weidner U. *Extracting Buildings from Digital Surface Models*, *IAPRS*, Vol. 32, Part 3-4W2, *3D Reconstruction and Modelling of Topographic Objects*, Stuttgart, September 17-19, 1997
- [9] Pimentel J., Batista N., Dionísio J. *Construção e gestão da complexidade de cenários urbanos 3D em ambientes virtuais imersivos*, *Actas 10º Encontro Português de Computação Gráfica, ISCTE*, 1-3 de Outubro, Lisboa, 2001.

VRML... porque não?

Alexandrino Gonçalves
Escola Superior de Tecnologia e Gestão
do Instituto Politécnico de Leiria
alex@estg.ipleiria.pt

António José Mendes
Centro de Informática e Sistemas da
Universidade de Coimbra
toze@dei.uc.pt

Resumo

O VRML surgiu da ambição em projectar a *World Wide Web* para um outro nível, o nível tridimensional. No entanto, e mesmo sem grandes alternativas, o VRML não é muito utilizado. De facto, o número de trabalhos/projectos disponíveis na Web que utilizam esta linguagem é mais baixo do que o que seria de esperar. Será que isso deriva das características e funcionalidades do próprio VRML?

O Fórum Flaviano de Conimbriga, um dos monumentos mais emblemáticos da antiga cidade romana de Conimbriga, serviu de base ao trabalho apresentado neste artigo. O nosso objectivo foi disponibilizar uma reconstrução virtual desse monumento, mas fazê-lo de forma a torná-la acessível a qualquer pessoa, nomeadamente através da Web, contribuindo para uma verdadeira democratização do saber.

Normalmente quando se pretendem criar ambientes tridimensionais na Web há três conceitos aparentemente incompatíveis entre si: realismo, desempenho e dimensão do(s) ficheiro(s). No entanto, através da linguagem utilizada no desenvolvimento deste trabalho, o VRML, e tirando partido das suas potencialidades, principalmente ao nível da optimização, conseguiu-se aliar esses três conceitos num só trabalho:

- Realismo: comprovado por especialistas, particularmente pelo Director do Museu Monográfico de Conimbriga;
- Desempenho: funciona perfeitamente num qualquer equipamento informático doméstico adquirido nos dias de hoje;
- Dimensão dos ficheiros: o principal e único ficheiro VRML ocupa apenas 22 KB, e o trabalho na sua totalidade (com as texturas), não ultrapassa os 150 KB.

Por estes resultados, este trabalho está disponível na Web em: <http://lsm.dei.uc.pt/forum>.

Expressões-chave

VRML, Optimização, Realismo, Desempenho, Dimensão dos ficheiros

1. INTRODUÇÃO

Paralelamente aos desenvolvimentos iniciais da Internet, surgiu também o conceito de ambiente virtual. A ideia é permitir às pessoas interagir com determinado ambiente, seja ele um museu ou um monumento já desaparecido, sem terem de sair de suas casas. Este conceito, aliado à expansão da *World Wide Web* e em consonância com as crescentes solicitações de projectar a Web para outro patamar (tridimensional), levou à criação do VRML (*Virtual Reality Modeling Language*).

O entusiasmo inicial com o aparecimento desta tecnologia, foi perdendo fulgor até aos dias de hoje, sendo actualmente o VRML uma tecnologia pouco utilizada.

Não existindo para já uma alternativa sólida (na Web), quais os motivos de tal facto? Provavelmente devido a estes três conceitos aparentemente incompatíveis: realismo, dimensão dos ficheiros e desempenho.

Realismo: sem ele, o ambiente virtual não é “competitivo” e o utilizador, agora visitante, não só não terá uma correcta percepção do espaço recriado, como não terá grande vontade de “lá voltar”.

Desempenho: se não proporcionar um movimento fluido num qualquer computador doméstico, limita fortemente o seu acesso pelos utilizadores finais.

Dimensão dos ficheiros: se a mesma não for diminuta, o consequente tempo de espera para a sua transferência tornará inviável a sua disponibilização via Web.

Em nossa opinião, é a não conciliação destes três conceitos o grande responsável pela inexistência de mais trabalhos/projectos em VRML, redundando num desinteresse crescente por esta tecnologia. Mas, será que esta responsabilidade poderá ser imputada às características e funcionalidades do próprio VRML?

2. O CASO DE ESTUDO

O Fórum Flaviano de Conimbriga foi construído entre 75 e 80 D.C. numa das maiores e mais prósperas cidades do antigo Império Romano em território “Lusitano”, a cidade de Conimbriga. O Fórum Flaviano era, como a maioria dos fóruns romanos, um monumento imponente e majestoso, quer pelas suas dimensões (96m x 48m), mas também pelo seu requinte arquitectónico, resultando num dos espaços mais emblemáticos e representativos da antiga cidade de Conimbriga. Actualmente, tal como a cidade em si, o Fórum está em ruínas e praticamente destruído.

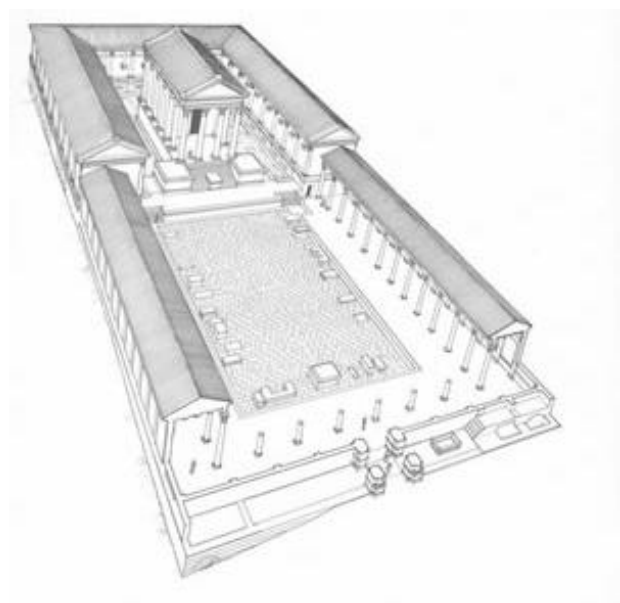


Fig. 1 O Fórum Flaviano de Conimbriga

O trabalho desenvolvido e apresentado neste artigo, teve como objectivo principal possibilitar a qualquer pessoa o desfrutar de uma visita interactiva, ainda que virtual, a este local representativo do património histórico e cultural de Portugal [Gonçalves, 2002].

Para tal, foi feita uma recriação tridimensional, o mais fidedigna possível (desde o número de degraus das escadarias, dimensões das lajes, padrões da época, cores, número de colunas e o seu correcto posicionamento, etc.) e à escala real do Fórum Flaviano de Conimbriga.

Uma nota para referir que este trabalho não teria sido possível, sem a colaboração directa do Museu Monográfico de Conimbriga, principalmente através do seu Director o Dr. Virgílio Correia.

3. A TECNOLOGIA

Como por vezes acontece, o acesso a estes paradigmas de visualização fica limitado a um grupo restrito de pessoas. Um dos objectivos deste trabalho visava a sua disponibilização a um maior número de pessoas possível. Pelas suas inúmeras potencialidades, que não é necessário aqui descrever, a *World Wide Web* surge como

um meio natural de difusão. É aqui que surge o VRML, dado que é a tecnologia standard para a publicação de conteúdos tridimensionais na Web.

Do processo de investigação e análise que se seguiu, cedo se percebeu a importância da conciliação dos três conceitos mencionados anteriormente, para a obtenção dos objectivos propostos. Foi assim, que um outro tipo de objectivos emergiram, estes de índole mais técnica e relacionados com a tecnologia que iria permitir “dar corpo” ao trabalho, o VRML. No caso foi dado especial enfoque às questões de optimização, para uma melhor e mais fluída navegabilidade e naturalmente à problemática da dimensão final do(s) ficheiro(s), de modo a minimizar o tempo de *download*.

4. ETAPAS DE DESENVOLVIMENTO

De seguida serão apresentadas as diversas etapas de desenvolvimento deste trabalho, cronologicamente encadeadas.

4.1 Investigação

Nesta fase, fundamental para os resultados finais obtidos, foi feita uma aprendizagem das características e reais potencialidades do VRML. Para numa etapa posterior se proceder à recolha de informação sobre as possibilidades e capacidades de optimização em VRML.

4.2 Ferramenta de Desenvolvimento

Apesar do VRML ser uma linguagem de sintaxe própria, que gera simples ficheiros de texto interpretados em tempo real pelo *browser*, existem aplicações que geram automaticamente o código VRML, libertando o autor dessa tarefa, podendo concentrar-se unicamente na concepção do espaço virtual. Pela possibilidade de implementação das questões de optimização já mencionadas, optou-se pela utilização de uma ferramenta de desenvolvimento que opera nativamente com o VRML, usufruindo desse modo das suas características. Por diversos motivos, a escolha recaiu na aplicação da Silicon Graphics, Cosmo Worlds 2.0.

4.3 Modelação

Partiu-se de seguida para a modelação das diversas superfícies geométricas pertencentes ao Fórum. Nessa modelação utilizou-se uma estratégia de “construção” modular e gradual, onde cada peça ou objecto era criado individualmente e em separado do ficheiro principal, e só posteriormente colocado na sua posição na cena, tal como peças de um puzzle. Nesta fase, foi dada particular atenção à existência de padrões geométricos repetitivos, pois tal facto permitia simplificar (e optimizar como se verá mais à frente) o processo de desenvolvimento, ao permitir modelar apenas um desses padrões geométricos e usar as propriedades do VRML para promover a sua repetição.

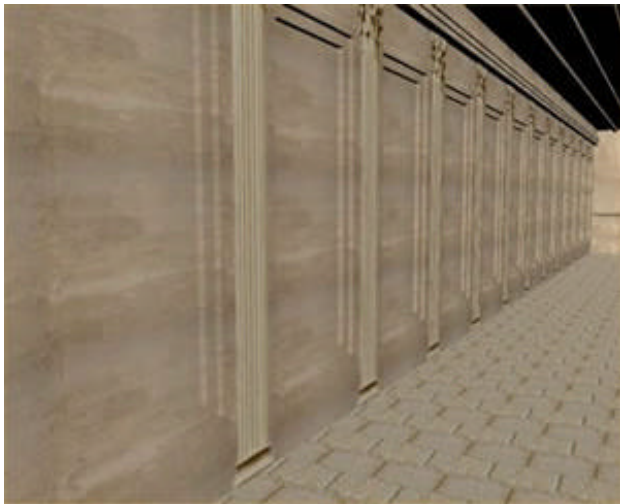


Fig. 2 Esta parede tem um padrão geométrico repetido

4.4 Montagem

Logo após a modelação de cada componente do Fórum, foi-se procedendo ao seu exacto posicionamento na cena, em conjunto com as formas geométricas já alojadas. Para esse efeito, na fase anterior foi definida uma sequência de modelação, privilegiando em primeiro lugar os objectos adjacentes entre si, para nesta fase melhor aquilatar a evolução da “construção” do Fórum.

4.5 Texturas

Pela sua importância para o resultado (visual) final, esta fase foi objecto de especial atenção e cuidado, desde a recolha de conteúdos à sua minuciosa manipulação gráfica. Eis uma breve descrição de algumas das texturas mais importantes usadas neste trabalho:

4.5.1 Coluna

Esta foi, sem margem para dúvidas, a textura que mais tempo e atenção mereceu na sua concepção. Basta um simples olhar pelas plantas e desenhos do Fórum para perceber que as mais de 100 colunas existentes no recinto, bem como as características muito particulares do seu capitel¹, poderiam ser condicionadoras do sucesso deste trabalho a dois níveis:

1. Desempenho geral do sistema: devido ao elevado número de colunas existentes;
2. Realismo: uma não correcta representação da ordem coríntia (ordem representativa das colunas do Fórum Flaviano de Conimbriga).

Após um cuidadoso trabalho de pesquisa e com recurso ao uso de transparências, obteve-se, segundo a opinião dos especialistas, uma coluna com um realismo bastante aproximado com a época em causa.



Fig. 3 Coluna do Fórum Flaviano de Conimbriga. Ordem coríntia representada no capitel da coluna

4.5.2 Laje

As pedras com dimensões consideráveis (80cm x 60cm) que revestiam a praça, e não só, têm um importante impacto visual para quem entra no recinto através da entrada principal. Após diversos testes, fotografias (digitais) e manipulação gráfica, foi criada uma pedra cujas características visuais se aproximam, mais uma vez na opinião dos especialistas, com as existentes na época.



Fig. 4 Textura do lajeado da praça

4.5.3 Parede

Esta pedra de aspecto rebocado também tem alguma importância, pela quantidade e dimensão das superfícies geométricas que deveriam ser revestidas com esta textura, que pela grande área de exposição, caracteriza de forma indelével a aparência geral do Fórum, pois onde quer que nos “encontremos” dentro do recinto, é a tonalidade cromática desta pedra que predomina. Mais

¹ Topo da coluna

uma vez e após sucessivos testes e análises se concebeu uma textura representativa da pedra rebocada da época.



Fig. 5 Textura usada como padrão das paredes

4.5.4 Frontão

Por forma a substituir, logo optimizando, umas centenas largas de polígonos, sem negligenciar o realismo, o frontão dos telhados foi obtido por intermédio de uma textura, mais uma vez com o recurso ao uso de transparências.



Fig. 6 Frontão dos telhados do Fórum

4.5.5 Gradeamento

Dado não existirem exemplares, mesmo danificados, de uma parcela desse gradeamento, esta foi a única textura concebida de raiz para este trabalho. Também aqui foram utilizadas transparências, para que a visualização entre os elementos cruzados do gradeamento transmitisse um efeito mais realista.

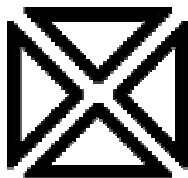


Fig. 7 Textura usada no gradeamento do Pórtico do Fórum

4.6 Coloração

Durante a elaboração das texturas fomos alertados pelos especialistas que, devido à erosão natural ocorrida ao longo dos séculos, as pedras existentes actualmente já não apresentam as tonalidades e cores originais. Assim, após a elaboração das texturas, foi aplicada, com o auxílio de uma ferramenta gráfica, uma coloração que os especialistas julgam ser muito próxima com a existente na época.

4.7 Iluminação

O uso de iluminação visa essencialmente a obtenção de um acréscimo no realismo global da cena, tentando proporcionar ao seu visitante uma melhor percepção de toda a ambiência original. Dentro dos diferentes tipos de iluminação existentes no VRML foram escolhidos os que têm menos implicações no desempenho, no caso o *DirectionalLight* e o *PointLight*, que com um correcto posicionamento na cena conseguem os efeitos pretendidos.

Para não eliminar alguns destes efeitos de iluminação, logo melhorando o efeito visual, o foco do avatar, que por defeito está “ligado”, foi colocado inactivo no momento da sua entrada na cena, podendo a qualquer altura o utilizador alterar este comportamento nas opções de configuração do *browser* VRML.

4.8 Optimização

Esta é a única etapa que não é apresentada na sequência cronológica do processo de desenvolvimento, precisamente porque não tem um cabimento cronológico, dado que esteve presente em todas as fases anteriores. Pela sua importância e pelos resultados obtidos, os aspectos de optimização são descritos na próxima secção.

5. OPTIMIZAÇÕES

Numa análise puramente técnica, este é o ponto chave e mais importante deste trabalho, pois as optimizações utilizadas foram fundamentais para os resultados obtidos.

De seguida serão apresentados alguns dos aspectos de optimização mais importantes implementados neste trabalho.

5.1 Instanciação (Clonagem)

Em vez de utilizar o, em muitas situações, ineficiente *copy/paste*, o VRML possibilita a utilização de uma instância de qualquer forma geométrica existente na cena. Essa propriedade é extremamente útil, pois evita a duplicação desnecessária de código, reutilizando as propriedades do(s) objecto(s) já definidas.

Aproveitando as características arquitectónicas do Fórum Flaviano, esta foi das técnicas mais utilizadas no desenvolvimento deste trabalho.

Existem no Fórum vários padrões geométricos que se repetem. A replicação de cada um desses padrões foi obtida precisamente com esta técnica. Por exemplo, a parede do corredor esquerdo segue um determinado padrão geométrico, como se pode constatar na figura 2. Logo apenas foi necessário modelar um desses padrões (figura 8) e a partir dele se obteve a totalidade do corredor.



Fig. 8 Bloco do corredor modelado

Após a conclusão do corredor esquerdo, constatou-se que o corredor direito é idêntico, diferindo apenas numa rotação de 180º acompanhada de uma translação. Assim sendo, todo o corredor esquerdo, que já é constituído por diversos clones, foi ele próprio clonado para dar origem ao corredor direito.

De referir ainda que a instanciação também pode ser usada em objectos que, sendo iguais geometricamente, não possuam as mesmas dimensões. Tal facto, dá a possibilidade de otimizar com este processo, mesmo onde aparentemente não seria possível fazê-lo. Exemplo disso é apresentado na figura seguinte. O pequeno objecto situado no início e ao centro da grande escadaria do templo é um clone de um dos pódios laterais.



Fig. 9 Objecto clonado através de uma mudança de escala

5.2 Colisões

Por serem efectuados num espaço tridimensional, os testes de detecção de colisões são geralmente pesados. Para simplificar esse processo duas medidas foram implementadas, fazendo uso das propriedades do VRML:

- Desactivação dos testes de colisão das superfícies com as quais não é possível colidir. Como apenas é permitido ao utilizador um “andar” gravitacional pelo recinto do Fórum, ele nunca irá colidir com muitas das superfícies existentes no mesmo, como por exemplo os telhados.
- Para facilitar e simplificar os testes de colisão foram definidas formas geométricas menos complexas, como primitivas simples, para serem usadas nos cálculos da detecção de colisões em substituição das formas originais, estas mais complexas e logo geradoras de cálculos mais pesados. Esse método permite uma redução significativa dos testes a efectuar aquando da verificação de uma possível colisão entre o avatar e uma qualquer superfície sólida. Este foi um procedimento usado, sempre que possível, em todo o espaço do Fórum. Na figura 10 pode-se apreciar um exemplo: toda a zona da base do pórtico, assinalada a vermelho na figura, é constituída por inúmeras “peças” individuais. No entanto, os testes de colisão são efectuados apenas a um simples paralelepípedo que abrange toda a sua extensão.



Fig. 10 *Colision proxy* é o nome do campo que permite implementar este procedimento

5.3 Texturas

A importância das texturas no aspecto final do trabalho levou a ter sido dada uma atenção muito especial a este item. Mas, se o realismo é fundamental, as texturas podem ter um impacto negativo, pelas suas dimensões, no desempenho global do sistema, pelo que também aqui urge otimizar:

- Usar texturas para simular superfícies complexas, que de outro modo necessitariam de centenas ou milhares de polígonos para as representar. Apesar dos vários casos existentes neste trabalho, o exemplo mais flagrante e que maior dificuldade gerou foi o capitel das colunas;
- Quase todas as texturas têm dimensões (altura e largura) potências de 2;
- Todas as texturas foram tratadas de modo a terem uma boa relação tamanho/qualidade. Muitas delas não ultrapassam os 2KB, sem que esse facto seja perceptível na qualidade geral do Fórum;
- Para uma maior economia no tamanho das texturas, algumas delas foram tratadas de modo a poderem ser dispostas entre si em mosaico. Um dos melhores exemplos é o lajeado da praça, que foi obtido com uma textura simples com 1.68KB (ver figura 4);
- Usando a mesma textura, logo sem necessidade de carregar outra(s) para a memória, conseguem-se diferentes efeitos visuais, apenas por alteração de alguns parâmetros do próprio VRML. Há diversos exemplos dessa situação, facilmente perceptíveis, recinto do Fórum.

5.4 Billboard

Um *billboard* em VRML é uma entidade geométrica que acompanha o movimento do avatar, de modo a que se apresente sempre de frente para o avatar.

A utilização de *billboards* neste trabalho resumiu-se a uma única entidade geométrica existente no Fórum. No entanto, a sua utilização foi um dos grandes responsáveis

pelos resultados obtidos ao nível do realismo, desempenho e tamanho final do ficheiro VRML.

Essa única entidade geométrica sobre a qual incidiu a utilização do *billboard*, foi a coluna coríntia do Fórum Flaviano de Conimbriga. O seu apurado detalhe geométrico e o elevado número de colunas existente por todo o recinto do Fórum, faziam com que fosse importante otimizar a sua representação.

Como os testes experimentais com esta metodologia se revelaram satisfatórios, essencialmente ao nível do realismo proporcionado, esta foi a técnica usada apenas numa das colunas do Fórum, pois todas as restantes foram obtidas por clonagem (secção 5.1), apesar de haver colunas de diferentes dimensões.

Com vista a um acréscimo no realismo, aplicou-se o *billboard* a uma superfície curva. Assim, partindo de um cilindro sem base nem topo e “cortando-o ao meio”, utilizou-se uma das metades para, não só colocar a textura, como aplicar o próprio *billboard*. Desse modo, houve melhorias na visualização a dois níveis:

- como os cálculos de iluminação são aplicados a uma superfície curva, o efeito proporcionado pela mesma é superior;
- a visualização da coluna, especialmente na zona do capitel, proporciona um efeito mais realista se aplicado numa superfície curva, em vez de plana.



Fig. 11 Efeito do *billboard* aplicado nas colunas

5.5 Número de Polígonos

Durante a modelação de todas as peças usadas neste trabalho, a preocupação com o número de polígonos foi sempre uma constante. No entanto, a redução do seu número não poderia lesar significativamente o realismo pretendido. Assim, mais uma vez, teve de prevalecer algum equilíbrio entre o realismo de cada peça e o número de polígonos necessários para a representar. Por vezes conseguiu-se uma redução significativa do número de polígonos, sem que isso tenha implicado um decréscimo, na mesma proporção, do realismo da peça em causa.

O modelo do Fórum Flaviano de Conimbriga tem perto de 17000 polígonos. Aparentemente, e dado que quase toda a volumetria do Fórum está visível logo no início da visualização, este número seria incomportável para uma boa fluidez do movimento, particularmente quando fosse usado um equipamento informático sem grandes

potencialidades gráficas. Na realidade, tal não acontece precisamente devido às optimizações efectuadas no desenvolvimento deste trabalho.

5.6 Hierarquização

Um ficheiro VRML é constituído por nós, alguns dos quais podem conter outros nós, que por sua vez podem ainda conter outros nós. Esta estrutura interna de um ficheiro VRML, sugere que os mesmos têm, ou podem ter, uma estrutura hierárquica. E este é, precisamente, um dos conceitos importantes no VRML.

Um dos testes realizados pelo *browser* durante a representação de um mundo virtual, é o teste de visualização (*culling*), ou seja, verifica quais os objectos que estão completamente fora do volume de visão do avatar, para assim poder determinar quais deles estão visíveis no momento. O teste consiste em interseccionar o volume de visão com a *bounding box* representativa do objecto e se esses dois volumes não se intersectarem, conclui-se que o objecto não está visível, logo pode ser ignorado durante o *rendering* da cena [Silicon Graphics98 a)]. Como se pode imaginar, este processo de intersecção de volumes é um cálculo pesado em termos de processamento interno, pois envolve cálculos matemáticos, a um nível tridimensional, bastante apurados.

Definindo uma estrutura hierárquica para a disposição dos objectos, todo o processo pode ser simplificado e optimizado na medida em que, os testes de visualização passam a ser efectuados apenas à *bounding box* do nó que está no topo da hierarquia, que pode, e deve, conter vários objectos que lhe estejam associados.

Assim sendo, todo o nosso ficheiro VRML é uma enorme estrutura hierárquica, onde cada objecto engloba outros objectos. Sendo também muito provável que este último, também ele seja parte integrante da hierarquia de um outro ainda maior.

Eis algumas metodologias implementadas para definir a estrutura hierárquica:

- Grandes objectos foram divididos noutros mais pequenos. Por sua vez, estes foram divididos em outros ainda mais pequenos, e assim sucessivamente. Talvez o exemplo mais claro desta situação neste trabalho, sejam os Pórticos do Templo. Como é uma estrutura que envolve toda a largura do Fórum, se fosse modelado e agrupado em conjunto, tornaria ineficiente a visualização, a navegação e a própria modelação. Assim, numa primeira abordagem foi “dividido” em ala esquerda, ala direita e ala central, prosseguindo em cada uma delas esta subdivisão por áreas modulares, como se pode constatar na figura seguinte referente à ala esquerda;

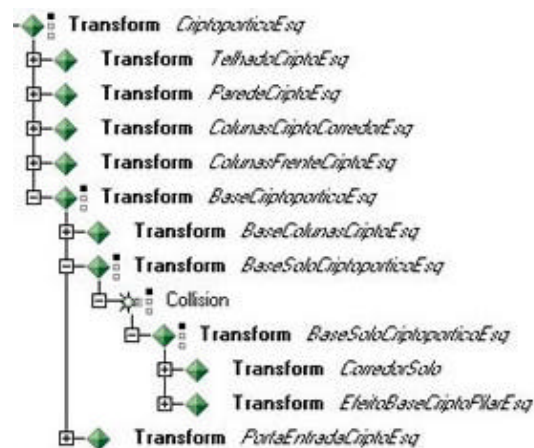


Fig. 12 Estrutura hierárquica expressa no Cosmo Worlds

- Em toda esta estrutura hierárquica existente no ficheiro VRML, qualquer nó de um determinado nível, tem todos os seus filhos próximos entre si, para facilitar e optimizar os testes de visualização e detecção de colisões.

5.7 Primitivas Simples

Dado que os browsers VRML estão optimizados para representar primitivas simples [Hartman96], sempre que possível foram utilizadas estas primitivas na modelação dos diferentes objectos do Fórum, mesmo quando tal não se afigurava necessário. Por exemplo, as paredes dos pórticos do templo, bem como o próprio templo (dado que o seu interior não foi modelado), não são mais que caixas (paralelepípedos). No entanto, houve outros objectos mais complexos, que foram “construídos” com recurso a primitivas simples, ora por intersecção das mesmas, ora pela disposição adjacente entre si. Por exemplo, todos os efeitos existentes nas portas, foram conseguidos através da disposição adjacente de paralelepípedos com dimensões (altura, largura e profundidade) distintas.

No entanto, a técnica de intersecção/sobreposição de paralelepípedos também foi utilizada. Por exemplo, com excepção de uma pequena peça, os pódios foram obtidos dessa forma (figura 13 esquerda), bem como muitos dos ornamentos da praça. Eis dois exemplos:

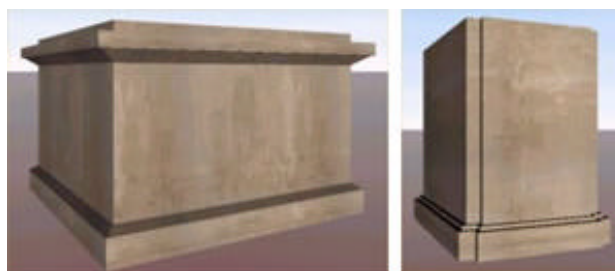


Fig. 13 Peças construídas por intersecção de primitivas simples. Um pódio (esq.) e um ornamento da entrada (dir.)

5.8 Organização Espacial

Uma boa organização espacial no ficheiro VRML implica que a proximidade dos objectos na cena, se reflecta na proximidade com que estão definidos no próprio ficheiro, principalmente em ficheiros muito extensos. Desta forma, optimizam-se os movimentos do *browser* pelo ficheiro durante a representação, limitando esses movimentos ao mínimo, proporcionando assim um *rendering* mais eficiente [Silicon Graphics98 b)].

Este foi um princípio adoptado durante a evolução do ficheiro VRML usado na “reconstrução” do Fórum. A figura seguinte representa a estrutura interna dos nós no Cosmo Worlds. Como se pode verificar, a proximidade entre objectos na cena, manteve-se, sempre que possível, na estruturação interna dos nós usados para a definição desses objectos no ficheiro VRML.

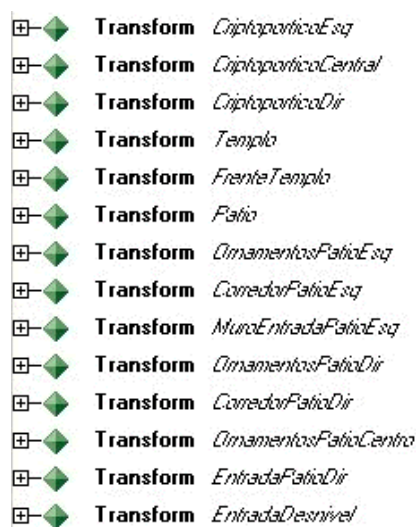


Fig. 14 Posicionamento interno dos objectos no Cosmo Worlds

5.9 Compressão

Actualmente, grande parte dos *browsers* VRML disponibilizam a descompactação do formato gzip, em tempo real, permitindo assim a leitura e visualização imediata do ficheiro compactado sem qualquer dificuldade.

No nosso caso, o processo de compressão foi muito simples, pois a ferramenta de trabalho (Cosmo Worlds) possui um comando que o faz automaticamente e sem dificuldades de maior.

6. RESULTADOS

Eis os resultados deste trabalho:

- **Realismo:** comprovado por especialistas, particularmente pelo Director do Museu Monográfico de Conimbriga;
- **Desempenho:** funciona em qualquer equipamento informático doméstico adquirido nos dias de hoje;

- **Dimensão dos ficheiros:** o principal, e único, ficheiro VRML ocupa apenas 22 KB, e o trabalho na sua totalidade (com as texturas), não ultrapassa os 150 KB.

Pelos resultados finais obtidos, este trabalho pode ser apreciado na Web em: <http://lsm.dei.uc.pt/forum>.

7. CONCLUSÃO

Da análise dos resultados apresentados na secção anterior, podemos concluir que os objectivos traçados para este trabalho foram atingidos na integra. O que nos leva à questão colocada no final da introdução deste artigo. Se com esta linguagem se conseguem realizar trabalhos com uma relação qualidade/desempenho bastante aceitável, com um baixo custo, e sem existirem grandes alternativas, porque não existirão mais projectos em VRML, principalmente na Web?

Em nossa opinião esse facto deve-se precisamente à não conciliação dos três conceitos que eram aparentemente incompatíveis entre si: realismo, desempenho e dimensão do ficheiros; que como acabámos de mostrar não serão assim tão incompatíveis.

Vivendo nós numa sociedade de consumo desenfreado, onde o lucro é quem mais ordena e como tal, tempo é dinheiro, raros são os casos, em quase todas as actividades, em que o método do *best effort* não é o que está presente em maior número de situações. É mais rápido, prático e por vezes também consegue produzir bons resultados. Isto para dizer, inserido no nosso contexto, que é nossa convicção que são poucos os trabalhos de VRML, elaborados de facto em VRML. Ou seja, o que muitas vezes acontece, e este é o primeiro grande erro de quem pretende realizar trabalhos com esta linguagem, é desenvolver a totalidade do mesmo numa qualquer ferramenta de modelação tridimensional, e no final usar os seus motores de exportação (uma grande maioria inclui essa possibilidade) para obter o trabalho final em VRML. As consequências dessa situação são principalmente os ficheiros de grande dimensão e de navegabilidade reduzida, se não mesmo impossível, em equipamentos informáticos domésticos.

Uma outra questão fundamental é a optimização do “código” VRML. Esta requer uma consciencialização profunda e o dispêndio de algum tempo dos criadores no processo de desenvolvimento de um qualquer projecto elaborado nesta linguagem.

Em conclusão, com um pouco de perseverança e alguns cuidados na concepção, principalmente ao nível da optimização, conseguem-se desenvolver bons trabalhos em VRML. Além do mais é uma tecnologia barata, fácil de utilizar, interactiva, standard para a Web, acessível a qualquer um, funciona num qualquer equipamento doméstico, é independente da plataforma, etc., pelo que deixamos este pensamento: VRML... porque não?

8. BIBLIOGRAFIA

[Gonçalves02] Alexandrino Gonçalves. *Reconstrução de ambientes históricos utilizando VRML: o caso do Fórum Flaviano de Conimbriga*. Dissertação de mestrado em Eng. Informática pela Universidade de Coimbra (2002).

[Hartman96] J. Hartman e J. Wernecke. *The VRML 2.0 handbook: Building moving worlds on the web*. Addison-Wesley Publishing Company (1996).

[Silicon Graphics98 a)] Silicon Graphics. *Cosmo Worlds 2.0 User's Guide* (1998).

[Silicon Graphics98 b)] Silicon Graphics. *Cosmo Worlds 2.0 Tutorial* (1998).

Sessão 4

REALIDADE AUMENTADA

VIDA: Visualizador Interactivo de Dados (biomecânicos) Aumentados

José Braz Pereira

Dep. Sistemas e Informática,
EST Setúbal/IPS, Setúbal
jbraz@est.ips.pt

João Madeiras Pereira

Dep. Engª Informática
IST/UTL, Lisboa
jap@inesc.pt

António Prieto Veloso

Dep. Ciências da Motricidade
Faculdade de Motricidade Humana/UTL
apveloso@fmh.utl.pt

Resumo:

Aborda-se o problema da construção e projecção de um modelo virtual tridimensional dos músculos de um atleta em movimento, nas sequências de vídeo captadas durante a execução desse movimento. Para a sobreposição de um modelo virtual do corpo humano, gerado em OpenGL com base nas posições tridimensionais de 21 juntas ósseas de um indivíduo, às sequências de vídeo do seu movimento, apresenta-se como solução o recurso a técnicas de reconhecimento de imagem e determinação da matriz de calibração da câmara através da Transformação Linear Directa Modificada.

Palavras Chave:

Realidade Aumentada; Realidade Melhorada; Computação Gráfica; Reconstrução Tridimensional; Biomecânica.

1 INTRODUÇÃO E ENQUADRAMENTO

A Biomecânica tem recorrido na última década, de forma crescente, à animação de modelos tridimensionais do corpo humano como meio de visualização de dados tanto cinéticos como biológicos. Prova disso é o crescente número de aplicações comerciais com elevado grau de sofisticação, tanto no que respeita aos modelos usados como às funcionalidades disponibilizadas [Vaughan92].

Neste enquadramento, o Laboratório de Biomecânica da Faculdade de Motricidade Humana (LB/FMH) da Universidade Técnica de Lisboa mostrou interesse numa aplicação para visualização da actividade muscular de atletas, através de uma solução graficamente intuitiva.

Para tal, o LB/FMH facultou o acesso aos dados cinéticos e biológicos já recolhidos, assim como ao ambiente de recolha de dados existente no laboratório.

O objectivo da aplicação aqui descrita é definir uma metodologia que perante:

- O ambiente de recolha de dados existente no Laboratório de Biomecânica;
- os dados já recolhidos, nomeadamente: as sequências de vídeo do movimento dos atletas, os dados cinéticos delas extraídos e os dados biológicos correspondentes, e;
- as restrições impostas pelos atletas relativamente aos dispositivos que poderiam envergar.

Permita:

- construir um modelo virtual tridimensional do atleta com o qual o biomecânico possa interagir

após a captação de dados, entendendo-se por interacção com o modelo a capacidade de lhe aplicar qualquer das três transformações singulares: rotação, translação e redimensionamento,

- sobrepor a componente muscular do modelo virtual do atleta em movimento à correspondente sequência de vídeo e,
- servir como uma primeira abordagem para uma futura introdução de um ambiente de Realidade Aumentada (RA) no Laboratório de Biomecânica da FMH.

O primeiro objectivo conduziu à realização do VIDA – Visualizador Interactivo de Dados biomecânicos Animados [Braz99], actualmente em funcionamento no LB/FMH.

Os dois últimos objectivos estiveram na origem da presente versão do VIDA (Visualizador Interactivo de Dados biomecânicos Aumentados).

2 ARQUITECTURA DO VIDA

A presente versão do VIDA, cuja arquitectura se apresenta na figura 1, segue de perto a descrição genérica do funcionamento de um sistema de Realidade Aumentada tal como descrito em [Vallino01], nomeadamente:

- 1 O Mundo Real observado consiste no Atleta em Movimento.
- 2 A imagem do Mundo Real é captada por um Sistema de Captação de Imagem (SCI), que no presente caso consiste em duas ou mais câmaras de vídeo fixas.

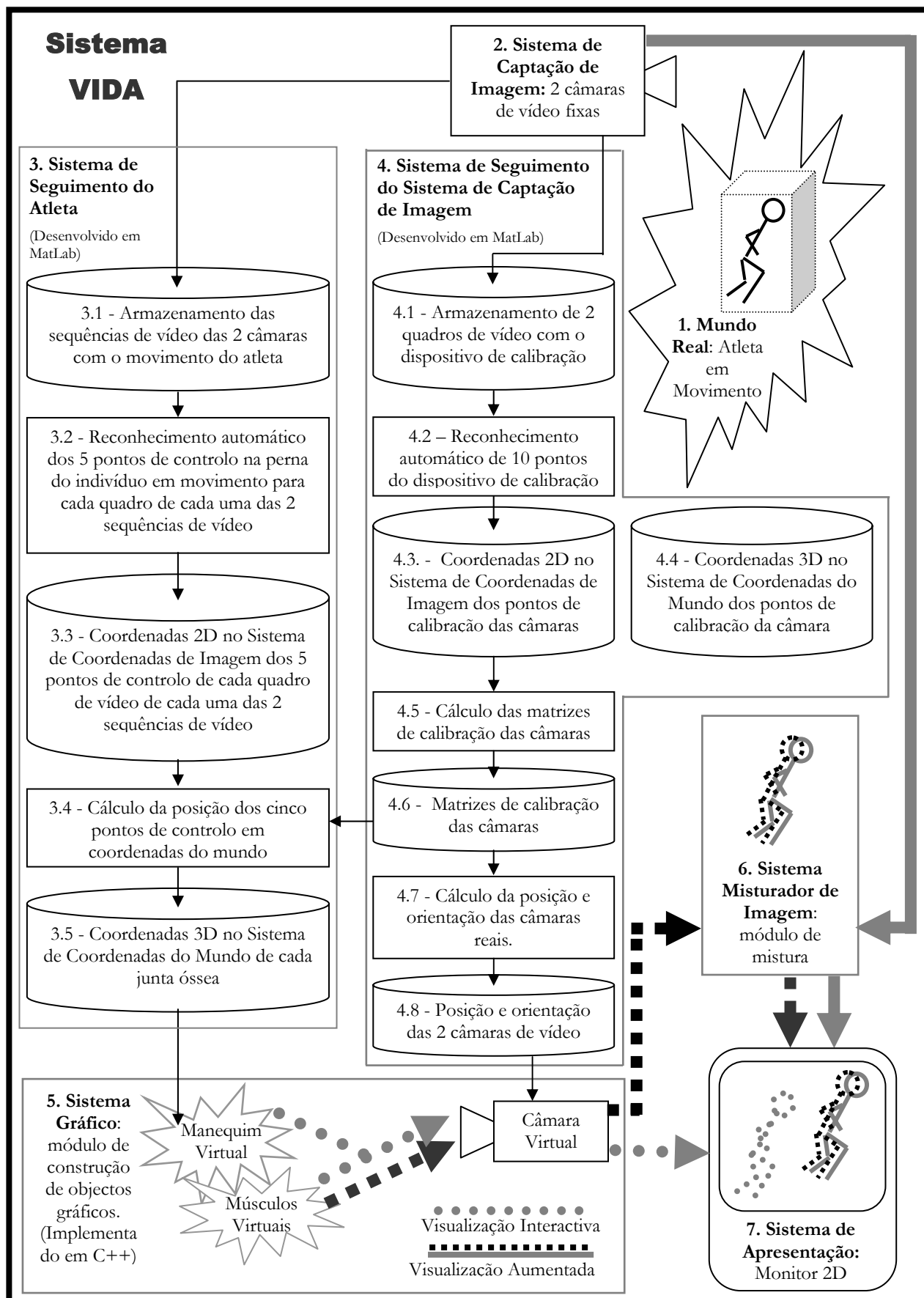


Figura 1: Arquitectura do Visualizador Interactivo de Dados Biomecânicos Aumentados

- Cada uma das câmaras executa a projecção perspectiva do mundo 3D num plano de imagem 2D.
 - Os parâmetros intrínsecos (distância focal e distorção da lente) e extrínsecos (posição e orientação) da câmara determinam o que é projectado no plano de visualização.
- 3 O Sistema de Seguimento do Atleta (SSA), que no presente caso recorre às imagens captadas pelas câmaras do sistema de captação de imagem, permite conhecer em cada instante a posição tridimensional de cada uma das 21 juntas ósseas [Braz99] pertinentes para a geração do modelo virtual do atleta.
 - 4 O Sistema de Seguimento do Sistema de Captação de Imagem (SSSCI) permite determinar os parâmetros intrínsecos e extrínsecos do SCI.
 - 5 A geração do objecto virtual (manequim e músculos do atleta) é executada por um Sistema Gráfico, no presente caso baseado no GLUT - OpenGL Utility Toolkit [Kilgard96].
 - O Sistema Gráfico necessita da informação proveniente do SSA (3) sobre a imagem do mundo real, por forma a reproduzir correctamente o manequim e os músculos virtuais; e da informação proveniente do SSSCI (4) por forma a gerar uma câmara virtual/sintética alinhada com a câmara real.
 - 6 No modo de apresentação aumentada a imagem dos músculos virtuais é “misturada” com a imagem do mundo real, no Sistema Misturador de Imagem, para formar a imagem de realidade aumentada e apresentada no Sistema de Apresentação (7).
 - 7 No modo de apresentação interactivo a imagem do manequim e dos músculos virtuais é apresentada no Sistema de Apresentação (7) sendo a projecção do modelo 3D no plano de imagem 2D controlada através da interacção do utilizador com o Sistema Gráfico (5).

2.1 Limitações

Tratando-se de uma versão destinada a um estudo de viabilidade, a presente versão do VIDA tem como objectivo tratar um caso elementar e tem em consideração as seguintes restrições:

- No que respeita a uma futura implementação de um sistema de RA, não aborda a problemática do seguimento em tempo real, e, consequentemente, a apresentação em tempo real do movimento do atleta.
- No que concerne às técnicas de Visão Computacional, não aborda a problemática do tratamento de oclusões.
- Destina-se a seguir apenas uma das pernas do indivíduo em movimento.

Tendo em mente estes três pressupostos e baseando-nos na Figura 1 podemos descrever o seu funcionamento da tal como se explicita em seguida:

2.2 Captação de Imagem

- O Mundo real (1), i.e., uma das pernas do indivíduo em movimento, é observado por um Sistema de Captação de Imagem (2) composto por 2 câmaras de vídeo fixas.

2.3 Dados de entrada

Os dados de entrada para a aplicação desenvolvida no âmbito do estudo de viabilidade consistem em:

- 2 Sequências de vídeo com a estrela de calibração [Pascoal01], para calibração das 2 câmaras fixas.
- 2 sequências de vídeo com o movimento da perna analisada.
- 1 matriz com as posições de cada um dos 10 pontos da estrela de calibração [Pascoal01] em coordenadas tridimensionais no Sistema de Coordenadas do Mundo

2.4 Seguimento do Sistema de Captação de Imagem

Considerando que as câmaras de vídeo estão fixas durante todo o processo de captação de dados, o Sistema de Seguimento do Sistema de Captação de Imagem (4) só necessita de calcular a posição e orientação das câmaras no início do processo de captação de dados.

- O procedimento de calibração das duas câmaras de vídeo começa com a captação, e armazenamento de, pelo menos, um quadro de vídeo em que sejam claramente visíveis 10 pontos, de que sejam conhecidas as coordenadas no sistema de coordenadas do mundo, doravante denominados pontos de calibração (bloco 4.1 da figura 1).
- Para cada quadro de vídeo, são detectados, com recurso a um algoritmo de reconhecimento de imagem, os 10 pontos da estrela de calibração [Pascoal01] e determinadas as coordenadas de cada um dos 10 pontos de calibração no sistema de coordenadas da imagem e a sua correspondência às respectivas coordenadas no sistema de coordenadas do mundo (bloco 4.2 da figura 1).
- As matrizes com as coordenadas no Sistema de Coordenadas da Imagem (4.3.) e correspondentes coordenadas no Sistema de Coordenadas do Mundo (4.4) de cada um dos 10 pontos de calibração são passadas para o módulo de cálculo da posição e orientação das câmaras reais (4.5).
- Os cálculos necessários para calibração de cada uma das duas câmaras de vídeo, i.e. para a determinação das suas posição e orientação em coordenadas do mundo, são descritos mais à frente na Secção 3.

2.5 Seguimento do movimento humano

O Sistema de seguimento da perna do indivíduo em movimento (3) permite conhecer em cada quadro de vídeo a posição de cada uma das 5 juntas ósseas pertinentes para a geração do modelo virtual da perna.

O seu funcionamento pode ser descrito da seguinte forma:

- As sequências de vídeo captadas pelas 2 câmaras de vídeo são armazenadas em disco (bloco 3.1 da figura 1)
- Para cada um dos quadros de vídeo de cada uma das duas sequências, são detectados e reconhecidos, com recurso a um algoritmo de reconhecimento de imagem os 5 pontos de controlo da perna, coincidentes com cada uma das 5 juntas ósseas (bloco 3.2 da figura 1).
- As coordenadas 2D no Sistema de Coordenadas de Imagem dos 5 pontos de controlo em cada quadro de vídeo de cada uma das 2 sequências de vídeo são armazenados (3.3).
- O cálculo da posição 3D em coordenadas do mundo de cada ponto de controlo para cada quadro de vídeo de cada uma das duas sequências de vídeo (3.4) é descrito mais à frente na secção 3.
- As Coordenadas 3D no Sistema de Coordenadas do Mundo de cada um dos 5 pontos de controlo, armazenadas (3.5), são passadas para o Sistema Gráfico (5) a fim de gerar o modelo virtual da perna seguida e dos respectivos músculos.

2.6 Visualização interactiva

Para o modo de visualização interactiva do Manequim, o Sistema Gráfico (5) do VIDA, com base nas coordenadas 3D no Sistema de Coordenadas do Mundo dos 5 pontos de controlo para cada conjunto de 2 quadros de vídeo sincronizados, vai construir o Manequim e apresentá-lo no Sistema de Apresentação (7). A construção tridimensional do manequim é descrita detalhadamente, mais à frente na secção 3.

2.7 Visualização aumentada

Para o modo de visualização aumentada do movimento do atleta:

- O VIDA começa por calcular a posição e orientação da câmara virtual com base na matriz de calibração da câmara (bloco 4.7 da figura 1).
- O Sistema Gráfico (5) do VIDA constrói a câmara virtual com base nos dados desse cálculo e em seguida projecta o modelo dos músculos no plano de imagem coincidente com o da câmara real.
- A imagem dos músculos alinhada espacialmente é depois misturada com o respectivo quadro de vídeo no sistema Misturador de Imagem (6).

- O quadro de vídeo aumentado com os músculos virtuais é então apresentado num sistema de apresentação (7).

3 REALIZAÇÃO DO SISTEMA VIDA

3.1 Problema

Se pretendermos, de alguma forma, estruturar o problema apresentado pelo LB/ FMH na sua totalidade, podemos afirmar que passa por:

- Detecção automática, em cada quadro de vídeo proveniente de cada uma das duas câmaras usadas, das marcas iridescentes que assinalam as posições dos centros articulares do indivíduo em andamento.
- Cálculo da posição tridimensional no sistema de coordenadas do mundo das acima referidas marcas com base nas posições bidimensionais da imagem de cada marca em coordenadas dos quadros de vídeo.
- Construção de um modelo virtual do indivíduo.
- Cálculo da posição e orientação da câmara de vídeo.
- Síntese de uma câmara virtual coincidente com a posição e orientação da câmara calculada.
- Projectação do modelo criado no plano de imagem.
- Sobreposição da projecção obtida ao quadro de vídeo.

O problema em causa pode ser reduzido ao caso mais simples da detecção de uma mesma marca iridescente em dois quadros de vídeo correspondentes a um mesmo instante no tempo e oriundos de diferentes câmaras com diferentes localizações e posterior cálculo da posição tridimensional da marca, em coordenadas do mundo, com base nas posições bidimensionais das imagens da marca em cada um dos dois quadros, em coordenadas do quadro de vídeo.

3.2 Reconhecimento de Marcas

O Módulo de Reconhecimento de Marcas implementado na versão do VIDA usada para experimentação com vista a uma futura realização de um sistema de RA, aborda a detecção de marcas iridescentes em sequências de vídeo num ambiente laboratorial controlado.

Considerando que o presente trabalho pretende ser apenas um estudo de viabilidade técnica, para o reconhecimento automático das marcas iridescentes recorreu-se a um algoritmo de reconhecimento de marcas desenvolvido propositadamente para as sequências captadas e a elas adaptado, na medida em que são conhecidas metodologias para a solução dos problemas tomados como pressupostos resolvidos, nomeadamente:

- Com o objectivo de limitar a área total de procura das marcas na imagem podemos recorrer à

subtração de duas imagens consecutivas, aplicando em seguida o algoritmo de procura à proximidade do subconjunto da imagem em que foi “detectado” movimento. [Haralick93].

- Com o objectivo de acelerar a procura na imagem de uma segunda câmara, das marcas já detectadas na imagem proveniente das marcas de uma primeira câmara, podemos procurá-las ao longo da recta epipolar [Seo00] [Zhang96].
- Com o objectivo de acelerar a procura de marcas nos quadros subsequentes de uma sequência de vídeo podemos limitar a procura a uma proximidade das coordenadas onde foi detectada a marca no quadro anterior.

O desenvolvimento do algoritmo para reconhecimento das marcas que assinalam as juntas ósseas na perna direita do indivíduo passou pela experimentação de diversos algoritmos, tendo a escolha final recaído num algoritmo de detecção de centróides que, apesar de pouco flexível, foi possível adaptar por forma a detectar de forma correcta as marcas no número de quadros de vídeo consecutivos necessários para a análise do movimento em causa.

Este algoritmo passa por:

- 1 Leitura de um quadro de uma sequência de vídeo.
- 2 Conversão do quadro numa imagem de intensidades.
- 3 Limiarização (*thresholding*) da imagem para conversão em imagem binária.
- 4 Aplicação de um filtro de dimensões para descartar grandes áreas de elevada intensidade.
- 5 Baseado no subconjunto da imagem obtido em (4) aplica-se à imagem de intensidades a transformada de Hough [Jain95] para detecção dos centróides correspondentes às marcas.
- 6 Determinação das coordenadas bidimensionais no sistema de coordenadas da imagem dos centros de massa dos centróides.
- 7 Determinação da correspondência entre os centros de massa detectados e cada uma das marcas.

Desta forma foi possível detectar todas as marcas da estrela de calibração (figura 2), assim como as cinco marcas, nas duas sequências de quadros provenientes das duas

câmaras utilizadas para a posterior reconstrução das posições tridimensionais dos pontos (figuras 3 e 4).

3.3 Reconstrução Tridimensional

Descrição do Método

De acordo com o raciocínio descrito em [Kwon94], simplificada, pode-se relacionar directamente o objecto e a imagem projectada tal como se vê na figura 5. O ponto O do objecto é directamente projectado no ponto I do plano de imagem. O ponto N é o centro de projecção e forma com os pontos O e I uma linha recta, verificando a condição de colinearidade. São definidos dois sistemas de coordenadas: o sistema de coordenadas do mundo [XYZ] e o sistema de coordenadas do plano de imagem [UV]. Os sistemas ópticos do dispositivo de digitalização e do dispositivo de visualização mapeiam



Figura 2: Marcas da estrela de calibração detectadas.

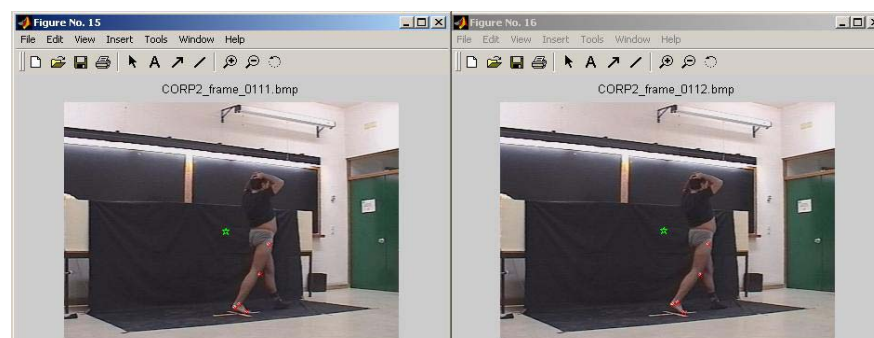


Figura 3: Quadros inicial e final, captados pela câmara 1.

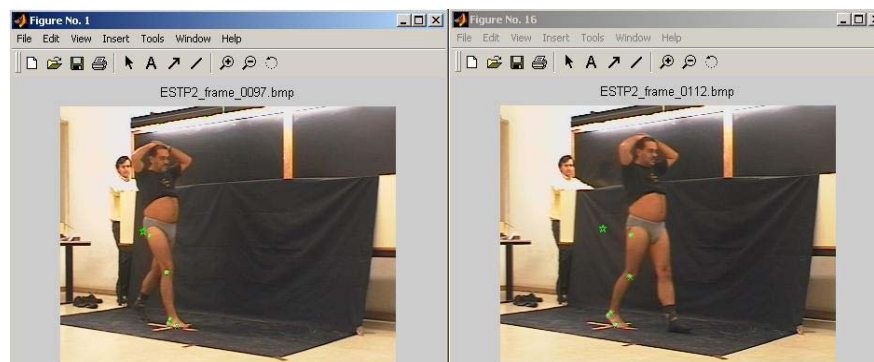


Figura 4: Quadros inicial e final, captados pela câmara 2.

o ponto O, de coordenadas $[x \ y \ z]$ no sistema de coordenadas do mundo, no ponto I, de coordenadas $[u \ v]$ no sistema de coordenadas do plano de imagem. Sendo $[x_0 \ y_0 \ z_0]$ as coordenadas do centro de projecção N, no sistema de coordenadas do mundo, o vector A, desenhado de N para O exprime-se como $[x-x_0 \ y-y_0 \ z-z_0]$.

Adicionando o eixo W, tal como se vê na Figura 5, ao sistema de coordenadas do plano da imagem, por forma a torná-lo tridimensional, as coordenadas em W dos pontos da imagem são sempre 0, e a posição 3D do ponto I torna-se $[u \ v \ 0]$.

Na Figura 5 introduziu-se um novo ponto P, denominado *Ponto Principal*, localizado na intersecção do denominado *Eixo Principal* com o plano da imagem. O Eixo Principal é a linha perpendicular ao plano da imagem e que passa pelo ponto N. Desta forma o eixo principal é de facto paralelo ao eixo W; d, denominada *Distância Principal*, é a distância entre os pontos P e N. Sendo $[u_0 \ v_0 \ 0]$ as coordenadas do Ponto Principal P, no sistema de coordenadas do plano de imagem, as coordenadas do centro de projecção são $[u_0 \ v_0 \ d]$, e o vector B, de N para I, terá como coordenadas:

$$[u-u_0 \ v-v_0 \ -d]$$

De acordo com a condição de colinearidade, os vectores A e B formam uma linha recta única. Se os dois vectores fossem definidos no mesmo sistema de coordenadas, a condição de colinearidade seria equivalente a [1]:

$$B^{(I)} = cA^{(I)} \quad [1]$$

em que c é uma constante de redimensionamento e $B^{(I)}$ e $A^{(I)}$ são os vectores A e B no sistema de coordenadas do plano da imagem. Note-se no entanto que o vector A está descrito no sistema de coordenadas do mundo enquanto o vector B está descrito no sistema de coordenadas do plano da imagem. A fim de relacionar estes dois vectores temos de os expressar num mesmo sistema de coordenadas. Uma forma de o fazer é exprimir o vector A no sistema de coordenadas do plano da imagem.

Seja R a matriz de transformação do sistema de coordenadas do mundo no sistema de coordenadas do plano de imagem, então:

$$A^{(I)} = R \cdot A = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \cdot A \quad [2]$$

em que $A^{(I)}$ é o vector A descrito em termos do sistema de coordenadas do plano da imagem. Substituindo [2] na condição de colinearidade [1], obtém-se [3.1] na forma compacta ou [3.2] na forma matricial:

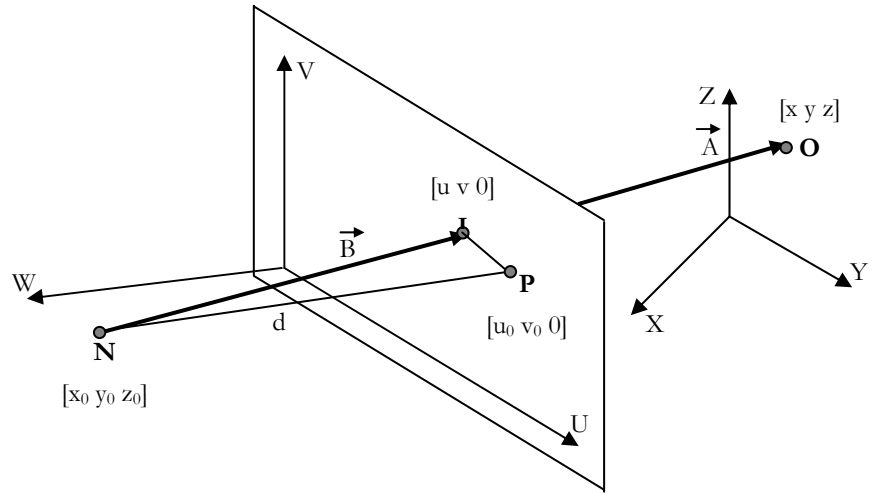


Figura 5: Sistemas de coordenadas do mundo e do plano de imagem

$$B = cRA \quad [3.1]$$

$$\begin{bmatrix} u - u_0 \\ v - v_0 \\ -d \end{bmatrix} = c \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \begin{bmatrix} x - x_0 \\ y - y_0 \\ z - z_0 \end{bmatrix} \quad [3.2]$$

Reduzindo [3.2] à forma de equações e resolvendo em ordem a c obtemos o sistema [4]:

$$c = \frac{-d}{r_{31}(x-x_0) + r_{32}(y-y_0) + r_{33}(z-z_0)}$$

$$u - u_0 = -d \frac{r_{11}(x-x_0) + r_{12}(y-y_0) + r_{13}(z-z_0)}{r_{31}(x-x_0) + r_{32}(y-y_0) + r_{33}(z-z_0)} \quad [4]$$

$$v - v_0 = -d \frac{r_{21}(x-x_0) + r_{22}(y-y_0) + r_{23}(z-z_0)}{r_{31}(x-x_0) + r_{32}(y-y_0) + r_{33}(z-z_0)}$$

Note-se que u_w, v_w, u_{w0}, v_{w0} em [4] são valores em unidades do sistema de coordenadas do mundo e podem diferir até um factor de escala dos valores obtidos pela digitalização uma vez que estes estarão em unidades do digitalizador. Assumindo que as unidades do digitalizador são diferentes das unidades do sistema de coordenadas do mundo, temos de considerar:

$$u_w - u_{w0} = \lambda_u(u - u_0) \quad [5]$$

$$v_w - v_{w0} = \lambda_v(v - v_0)$$

Em que $[u \ v]$ são as coordenadas do ponto I no sistema de coordenadas do mundo em unidades do digitalizador, $[u_0 \ v_0]$ as coordenadas do ponto principal no sistema de coordenadas do mundo em unidades do digitalizador, e $[\lambda_u \ \lambda_v]$ são os factores de escala para a conversão das unidades do digitalizador nas unidades do Sistema de Coordenadas do Mundo.

Substituindo [5] em [4] e rearranjando por forma a determinar u e v obtemos:

$$u = \frac{L_1 x + L_2 y + L_3 z + L_4}{L_9 x + L_{10} y + L_{11} z + 1}$$

$$v = \frac{L_5 x + L_6 y + L_7 z + L_8}{L_9 x + L_{10} y + L_{11} z + 1} \quad [6]$$

onde:

$$\begin{aligned}
 L_1 &= \frac{u_0 r_{31} - d_u r_{11}}{D} \\
 L_2 &= \frac{u_0 r_{32} - d_u r_{12}}{D} \\
 L_3 &= \frac{u_0 r_{33} - d_u r_{13}}{D} \\
 L_4 &= \frac{(d_u r_{11} - u_0 r_{31})x_0 + (d_u r_{12} - u_0 r_{32})y_0 + (d_u r_{13} - u_0 r_{33})z_0}{D} \\
 L_5 &= \frac{v_0 r_{31} - d_v r_{21}}{D} \\
 L_6 &= \frac{v_0 r_{32} - d_v r_{22}}{D} \\
 L_7 &= \frac{v_0 r_{33} - d_v r_{23}}{D} \\
 L_8 &= \frac{(d_v r_{21} - v_0 r_{31})x_0 + (d_v r_{22} - v_0 r_{32})y_0 + (d_v r_{23} - v_0 r_{33})z_0}{D} \\
 L_9 &= \frac{r_{31}}{D} \quad L_{10} = \frac{r_{32}}{D} \quad L_{11} = \frac{r_{33}}{D} \\
 d_u &= \frac{d}{\lambda_u} \quad d_v = \frac{d}{\lambda_v} \quad D = -(x_0 r_{31} + y_0 r_{32} + z_0 r_{33})
 \end{aligned}$$

Em [6], os parâmetros L_1 a L_{11} , denominados parâmetros DLT em [Kwon94], reflectem a relação entre o sistema de coordenadas do mundo e o sistema de coordenadas do plano de imagem, i.e. os valores da matriz de calibração do dispositivo de captação de imagem, no nosso caso, da câmara de vídeo.

Calibração da Câmara

A calibração da câmara em relação a um sistema de coordenadas tomado como padrão, passa assim pela determinação dos 11 parâmetros L_1 a L_{11} .

Rearranjando [6] em ordem aos parâmetros DLT obtemos:

$$\begin{bmatrix} x & y & z & 1 & 0 & 0 & 0 & 0 & -ux & -uy & -uz \\ 0 & 0 & 0 & 0 & x & y & z & 1 & -vx & -vy & -vz \end{bmatrix} \begin{bmatrix} L_1 \\ L_2 \\ \vdots \\ L_{10} \\ L_{11} \end{bmatrix} = \begin{bmatrix} u \\ v \end{bmatrix} \quad [7]$$

ou na forma compacta:

$$X L = U \quad [8]$$

De [7] podemos concluir que, para determinar os 11 parâmetros DLT, necessitamos de conhecer as posições tridimensionais $[x \ y \ z]$, no sistema de coordenadas do mundo tomado como padrão, e bidimensionais $[u \ v]$, no sistema de coordenadas do plano de imagem, de pelo menos 6 pontos de controlo.

Resolvendo [8] em ordem a L :

$$(X^t X)^{-1} (X^t X) L = (X^t X)^{-1} (X^t U)$$

Obtemos:

$$L = (X^t X)^{-1} (X^t U) \quad [9]$$

Construindo as matrizes X e U com as posições tridimensionais das marcas de calibração em coordenadas do mundo e as posições bidimensionais, e calculando [9] através do algoritmo descrito em [Kwon94], obtemos os parâmetros DLT.

Cálculo das Coordenadas 3D das Marcas no Sistema de Coordenadas do Mundo

Uma vez determinados os valores da matriz de calibração, aqui denominados parâmetros DLT, podemos rearranjar [6] em ordem a $[x \ y \ z]$ e obtemos [10] na forma compacta:

$$\begin{bmatrix} L_1 - uL_9 & L_2 - uL_{10} & L_3 - uL_{11} \\ L_5 - vL_9 & L_6 - vL_{10} & L_7 - vL_{11} \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} L_4 - u \\ L_8 - v \end{bmatrix} \quad [10]$$

calculando a pseudo-inversa da mesma forma que em [9], obtemos as coordenadas dos pontos de controlo no sistema de coordenadas do mundo.

Na Figura 6 apresenta-se a posição inicial e final do modelo virtual da perna direita construído com base nas coordenadas tridimensionais oriundas da reconstrução.

3.4 Construção do Modelo Virtual

Na versão do VIDA usada para experimentação e aqui descrita foram seguidos apenas os 5 pontos de controlo necessários para a reconstrução da perna direita do indivíduo em movimento. Na versão actualmente usada no Laboratório de Biomecânica da FMH, as coordenadas tridimensionais de cada marca no sistema de coordenadas do mundo, doravante denominadas são consideradas dados de entrada para o Sistema Gráfico do VIDA (módulo 5 na figura 1).

Modelização do Manequim

O modelo virtual do indivíduo foi construído em OpenGL [Kilgard96] colocando uma esfera nos pontos de controlo correspondentes a juntas ósseas e um paralelepípedo no ponto euclidianamente intermédio a cada junta óssea. Os cálculos necessários são executados em C++ e descritos em seguida.

Animação do Manequim

Para construir e animar o manequim existem como dados de entrada, para cada quadro ou trama, as posições dos 21 pontos correspondentes na sua generalidade às juntas ósseas do corpo humano. A

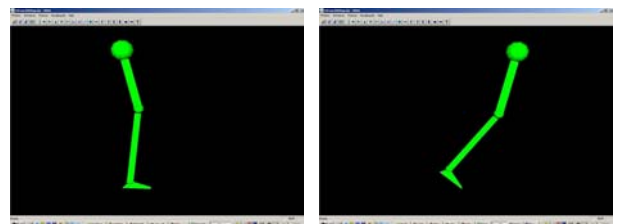


Figura 6: Modelo virtual da perna direita

posição de cada marca, e correspondente numeração, faz parte da especificação de requisitos da aplicação [Braz99].

Cálculos das coordenadas:

Os procedimentos e cálculos executados pelo Sistema Gráfico, e realizados em C++, podem ser descritos na seguinte forma:

1. A posição tridimensional da junta óssea intersegmentar inicial, (x_1, y_1, z_1) na Figura 7, é lida de um ficheiro de MS Excel que contém os dados cinemáticos referentes aos 21 pontos de controlo.
2. É calculada a rotação (ângulo e eixo de rotação) que essa junta tem que sofrer para que um vector U que nela se inicie, atinja a posição da junta óssea seguinte: (x_2, y_2, z_2) na Figura 7.
3. É determinada a distância euclidiana entre a junta óssea em causa (x_1, y_1, z_1) e a que se lhe segue no corpo humano (de cima para baixo) (x_2, y_2, z_2) .
4. As coordenadas da posição da peça óssea são determinadas como as coordenadas em x e em z da junta óssea de início e metade da distância entre as duas juntas ósseas em y : $(x_{OssO}, y_{OssO}, z_{OssO})$

Visualização

Para apresentação do conjunto junta intersegmentar óssea/segmento ósseo (ou peça óssea) procede-se da seguinte forma:

5. Translada-se o centro do sistema de coordenadas para a posição da junta intersegmentar óssea inicial lida em 1) (`glTranslate`)
6. Roda-se o sistema de coordenadas, no ângulo determinado em 2) e em redor do eixo determinado em 2). (`glRotate`)
7. Desenha-se a junta óssea (`gluSolidSphere`)
8. Desloca-se o centro das coordenadas para a posição determinada em 4) (`glTranslate`)
9. Desenha-se a peça óssea que tem início na junta óssea anteriormente desenhada (`gluSolidBox`).
10. Executa-se novamente o ponto 5) agora para a junta óssea seguinte.

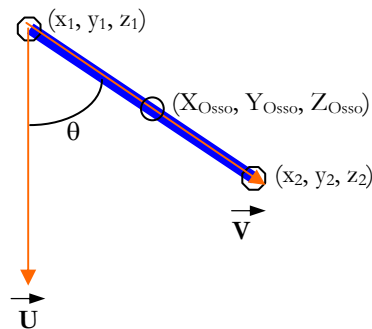


Figura 7: Coordenadas usadas no cálculo da rotação de cada peça óssea por forma a ajustá-la às coordenadas da juntas ósseas inicial e final obtidas por reconstrução tridimensional.

A Figura 7 apresenta os elementos usados nos cálculos de posição e rotação para cada quadro de todas as juntas e peças ósseas. Na referida figura (x_1, y_1, z_1) são coordenadas da junta inicial (p.e. da anca) lidas do ficheiro de dados cinemáticos, (x_3, y_3, z_3) são as coordenadas da junta final (p.e. do joelho) lidas do mesmo ficheiro, (x_2, y_2, z_2) . $(X_{OssO}, Y_{OssO}, Z_{OssO})$ são as coordenadas intermédias entre as juntas ósseas intersegmentares inicial e final e que correspondem à localização da peça óssea (ou muscular) localizada entre elas (p.e. a coxa).

Animação dos Músculos

Sempre que um músculo necessite de mais de um segmento de recta para sua representação, os cálculos executados para posicionamento desse músculo são em tudo semelhantes aos usados para o posicionamento das peças ósseas.

Coloração dos Músculos

Para codificar numa gama de cores os dados correspondentes à indutância da superfície da pele sobre o músculo em causa, procede-se da seguinte forma:

- O valor correspondente ao esforço muscular de um músculo é lido do ficheiro de Excel que contém os dados fisiológicos.
- É determinado o valor máximo.

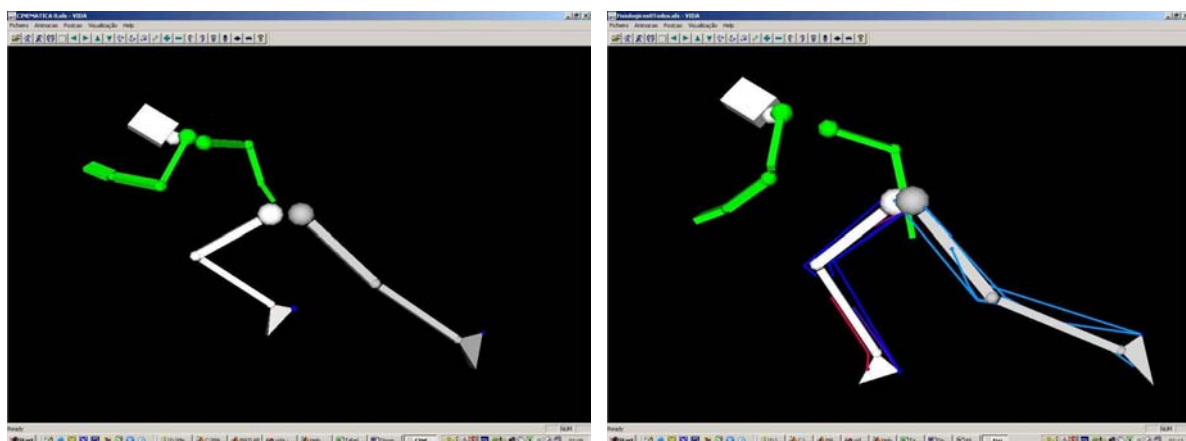


Figura 8: Imagem do modelo virtual do indivíduo em movimento com e sem músculos visíveis.

- Os valores são normalizados entre 0 e 1: $\text{valor}[i] = \text{valor}[i]/\text{valor máximo}$
- A cor é determinada como: Red = $\text{valor}[i]$; Green = 0; Blue = $1 - \text{valor}[i]$;

Na figura 8 apresenta-se um quadro do modelo virtual do indivíduo em movimento, com e sem músculos, construído a partir de dados fornecidos pelo LB/FMH.

3.5 Projectação do Modelo Virtual no Plano de Imagem da Câmara

Para conseguir o correcto alinhamento visual do modelo virtual sobre o quadro de vídeo é ainda necessário determinar a posição da câmara assim como as coordenadas do seu ponto principal.

Cálculo da Localização da Câmara e das Coordenadas do Ponto Principal

A posição da câmara no sistema de coordenadas do mundo pode ser calculada como:

$$L_1x_0 + L_2y_0 + L_3z_0 = -L_4$$

$$L_5x_0 + L_6y_0 + L_7z_0 = -L_8$$

$$L_9x_0 + L_{10}y_0 + L_{11}z_0 = -1$$

$$\begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} -L_4 \\ -L_8 \\ -1 \end{bmatrix}$$

$$\begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} = \begin{bmatrix} L_1 & L_2 & L_3 \\ L_5 & L_6 & L_7 \\ L_9 & L_{10} & L_{11} \end{bmatrix}^{-1} \begin{bmatrix} L_4 \\ L_8 \\ -1 \end{bmatrix} \quad [11]$$

e, desde que os parâmetros DLT sejam calculados com base na Transformação Linear Directa Modificada [Kwon94] as coordenadas do ponto principal $[u_0, v_0]$ no sistema de coordenadas de imagem podem ser determinadas a partir de [6] como:

$$L_9^2 + L_{10}^2 + L_{11}^2 = \frac{1}{D^2} (r_{31}^2 + r_{32}^2 + r_{33}^2) = \frac{1}{D^2}$$

$$u_0 = D^2 (L_1L_9 + L_2L_{10} + L_3L_{11}) = \frac{L_1L_9 + L_2L_{10} + L_3L_{11}}{L_9^2 + L_{10}^2 + L_{11}^2}$$

$$v_0 = D^2 (L_5L_9 + L_6L_{10} + L_7L_{11}) = \frac{L_5L_9 + L_6L_{10} + L_7L_{11}}{L_9^2 + L_{10}^2 + L_{11}^2}$$



Figura 9: Visualização aumentada do movimento do indivíduo

Para os dados fornecidos pelo LB/FMH foram determinados três conjunto de parâmetros DLT, correspondentes às três câmaras usadas para a captação do movimento de um indivíduo, com base nos quais foram sintetizadas as câmaras virtuais alinhadas com as reais o que permitiu alinhar o manequim apresentado na Figura 9 com a sequência de vídeo por forma a apresentar uma visualização aumentada do indivíduo em movimento de que se apresenta um exemplo na Figura 10.

4 CONCLUSÕES E TRABALHO FUTURO

Actualmente a versão em utilização do VIDA permite:

- Observar o esforço muscular de um indivíduo directamente sobre um modelo virtual tridimensional manipulável desse indivíduo.
- Projectar o modelo virtual do indivíduo sobre as sequências de vídeo a partir das quais foi construído esse mesmo modelo, criando um ambiente de Realidade Misturada, intermédio, entre o processo tradicional e o ambiente de Realidade Aumentada desejável.

Não lidando portanto, nem com o tratamento de oclusões nem com o funcionamento em tempo real, imprescindíveis para uma visualização de dados biomecânicos baseada num Sistema de Realidade Aumentada.

A evolução natural do presente trabalho, no que respeita à introdução de um sistema de Realidade Aumentada no Laboratório de Biomecânica da FMH, parece-nos ser no sentido de:

- Permitir a visualização de dados em tempo real.

- Incorporar a possibilidade de visualização através de um HMD Video see-through.

Tarefas que, considerando o trabalho de investigação, os conhecimentos exigidos e o volume de trabalho em causa, seria desejável desenvolver no âmbito de um projecto de investigação e desenvolvimento com financiamento autónomo.

No que concerne a aspectos de funcionamento da aplicação nos quais o Laboratório de Biomecânica da FMH manifestou já especial interesse, como sejam:

- A substituição dos modelos paralelipédicos das peças ósseas por modelos realistas.
- A substituição dos modelos lineares das massas musculares lineares por modelos realistas.
- O cálculo das forças de reacção nas juntas e dos momentos musculares, assim como a sua visualização sob forma de vectores aplicados nos respectivos pontos de origem

Consideramos tratar-se de objectivos facilmente atingíveis a breve trecho, possivelmente no âmbito de um projecto de final de um Curso de Engenharia Informática.

5 BIBLIOGRAFIA

[Braz99] José Braz, “*Relatório Técnico do Projecto da Disciplina de Computação Gráfica II do Mestrado em Engenharia Electrónica e de Computadores*”; Instituto Superior Técnico, 1999.
<<http://ltodi.est.ips.pt/jbraz/Projectos.htm>>

[Haralick93] Robert M. Haralick, Linda G. Shapiro: Computer and Robot Vision Volume II; Addison-Wesley Publishing Company, Inc., 1993. ISBN:0-201-56943-2(v.2)

[Jain95] R. Jain, R. Kasturi, B. G. Schunk, “*Machine Vision*”, McGraw-Hill Series in Computer Science, 1995. ISBN:007-032018-7. pgs 218-223.

[Kilgard96] Mark J. Kilgard, “*The OpenGL Utility Toolkit (GLUT)- API Version 3*”; Silicon Graphics, Inc., November 13, 1996.
<<http://informatica.uv.es/iiguia/2000/AIG/docs/glut3.pdf>>

[Kwon94] Young-Hoo Kwon: *Kwon3D: DLT Method*.
<<http://kwon3d.com/theories.html>>

[Pascoal01] Augusto Gil Brites de Andrade Pascoal: “*Ombro e Elevação do Braço, Análise cinemática e electromiográfica sobre a influência da carga externa e velocidade do braço no ritmo escapulo-umeral tridimensional*”. Dissertação com vista à obtenção do grau de Doutor em Motricidade Humana na especialidade de Ciências da Motricidade, Universidade Técnica de Lisboa, Faculdade de Motricidade Humana, Lisboa, Portugal, 2001.

[Seo00] Yongduek Seo, Ki Sang Hong, “*Calibration-Free Augmented Reality in Perspective*”, in IEEE

Transactions on Visualization and Computer Graphics, Vol. 6, Nº4, October-December, 2000.

[Vallino01] Jim Vallino: Introduction to Augmented Reality; 2001.
<http://www.se.rit.edu/~jrv/research/ar/introduction.html>.

[Vaughan92] Cristopher L. Vaughan, Brian L. Davis, Jeremy C. O'Connor “*Dynamics of Human Gait*”, Human Kinetics Publishers, July, 1992, ISBN: 087-322-37-13.
<<http://isb.ri.ccf.org/data/gaitlab/>>

[Zhang96] Z. Zhang, “*Determining the Epipolar Geometry and Its Uncertainty: A Review*”, Technical Report 2079, INRIA, France, July, 1996.

Image Manipulation through Gestures

José Miguel Salles Dias
Miguel.Dias@iscte.pt

Pedro Nande
Pedro.Nande@iscte.pt

Pedro Santos
Pedro.Santos@iscte.pt

Nuno Barata
Nuno.Barata@iscte.pt

André Correia
Andre.Correia@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

In this work we present a novel free-hand gesture user interface based on detecting the trajectory of fiducial markers attached to the user's fingers and pulse, able to interact a sequence of images of a digital video piece. The model adopted for the video representation, is based in its decomposition in a sequence of frames, or filmstrip. Totally sensor-less and cable-less interfaces, provide the means for a user to intuitively interact through gestures with the filmstrip, within the framework of an Augmented Virtuality usage scenario. By simply gesturing, users are able to select at random, drag, release, delete or zoom image frames, browse the filmstrip at a controlled user-defined rate and issue start, end, stop and play commands to better control the digital video sequence. A fixed video camera monitors the user interaction through gesturing of the mentioned fiducial markers. This scheme enables the system to simplify the more complex problem of marker-less free-hand gesture tracking. Once the markers are detected and recognized in real-time by the computer vision layer, the system obtains the 3D pose (position and orientation) of the marker centres in relation to a virtual camera reference frame, whose mathematical model matches the real video camera. We are specifically interested in obtaining the pose of the left and right hand pulses, left and right thumb, and left and right hand index. By projecting the positions of these poses in the 2D visualization window, simple topological analysis based in the study of the kinematics evolution of distances and angles, can be implemented, enabling gesture recognition and the activation of system functions and, subsequently, of specific gesture-based user interaction for a given active functionality. This interaction will affect the shape, scale factor, position and visualisation of scene objects, that is, of filmstrip frames. For the computer vision layer, our system adopts AR Toolkit, a C/Open GL-based open source library that uses accurate vision based tracking methods to determine the virtual camera pose information through the detection in real-time of fiducial markers. The graphical output is implemented with C++/Open GL. Our proposed system is general, in the sense that it can interact with any filmstrip, obtained "a priori" from a digital video source.

Keywords

Augmented Virtuality, Tangible Interfaces, AR Toolkit, Hand and Finger Gesture, Digital Video Editing, Image Browsing, Filmstrip.

1. INTRODUCTION¹

With our work, we are aiming to provide innovative Mixed Reality technology, applicable for user interaction with images of a digital video sequence. The model adopted for the video representation, is based in its decomposition in a sequence of frames, or filmstrip. Totally sensor-less and cable-less interfaces, used in previous works as Tangible Interfaces [Dias 03a], but adopted here as simple fiducial markers for the underlying vision

tracking system, provide the means for a user to intuitively interact through gestures with the filmstrip, within the framework of an Augmented Virtuality usage scenario. The augmentation of the virtual environment introduced by our system, where image frames from the filmstrip are mapped into a 3D model of a projection screen, enables the user to concentrate in the task at hand and does not add a cognitive seam in the visualisation process. In video or image editing and visualization scenarios, creating a technological environment where a user could experience intuitive gesture-based interaction with a filmstrip, could be something very rewarding and revolutionary for these users.

¹ The test images and filmstrip here presented are part of the Official Trailer of the Minority Report Motion Picture © by DreamWorks available in their corporate site.

This work was in part motivated by Spielberg's "Minority Report" gestural and visual-based HCI language. The paper focuses in describing the basic techniques and in showing the potential of the concept of image manipulation by means of gesturing, and should be seen as an illustration of a technological platform, whose validity evaluation in a given usage context, still needs to be accessed by a proper usability evaluation methodology, such as in [Dias 03b], currently on-going and to be published in the near future. Although the test application described here is related to image editing, the purpose of the paper is to deploy a technique that could be used also in other application scenarios, such as large image database browsing. The system uses hand gesture detection as an HCI² modality, by adopting a trajectory approach and a marker-based technique. This marker-based scheme can be seen as a transitional technology for this type of gestural HCI modality systems, since it simplifies the more complex and general problem of marker-less free-hand gesture trajectory tracking.

In synthesis, in this paper, we address the appropriateness and technical viability of marker-based free-hand gesture user interfaces in Augmented Reality, for the enhancement of visualisation and interaction with digital images in video sequences.

The paper is organised as follows: in section 2, we provide a background and state-of-the-art in the issues of Augmented Reality, Mixed Reality and Tangible Interfaces. In section 3, we present our modular system functions and detail the hardware and software of the developed prototype platforms. Section 4 covers the issues of the User Interfaces adopted in our system. Finally, in section 5, conclusions and future directions of research are given.

2. BACKGROUND

Augmented Reality (or AR) systems and technologies were introduced in 1992 by Caudel and Mizell [Caudel92], in the context of a pilot project, where they were used to simplify an industrial manufacturing process in a Boeing airplane factory. In general these systems provide the means for "intuitive information presentation, which enhances the perceiver's situational awareness and cognitive perception of the real world" [Behringer 99]. This enhancement is achieved by placing virtual objects or information cues into the real world, which is made possible by performing "virtual camera calibration", that is, by computing the virtual camera parameters that match the position and orientation of the observer of the real scene. With this technique, "virtual" objects can then be registered in relation to "real" objects which means that these objects can be seen in the same position and orientation of other "real" objects of the scene, as perceived by the user. This is usually done using optical or video see-through head mounted displays and tracking devices, linked to either standalone computers with 3D graphics capabilities, or mobile wearable computers. Video see-through AR is where virtual images are over-

laid on live video of the real world. A possible alternative is optical see-through AR, where computer graphics are overlaid directly on a view of the real world.

According to Azuma [Azuma 97], AR "1) combines real and virtual environments; 2) is interactive in real-time; 3) is registered in 3D. We can conclude that AR is a challenging multidisciplinary field and a particularly promising new user interface paradigm, integrating computer graphics, computer vision, positional and orientation tracking technologies, wearable computer architectures, psychology, artificial intelligence, ergonomics and others. In view of this integration framework and associated complexity, Milgram and Herman [Milgram 99], have proposed a new taxonomy that supplements the definition of Augmented Reality, by introducing the concepts of Augmented Virtuality (AV) and Mixed Reality (MR). They argue that Real Environments (RE) and Virtual Environments (VE) are, in fact, two poles of the Reality-Virtuality Continuum, being RE the left pole and VE, the right pole. REs include sample data acquired and digitised from real scenarios, such as image, video, ultrasound, X-ray, laser range and light intensity data and others, whereas VEs, refer to computer models of 3D scenarios that can be rendered in real-time. AV includes 3D scenarios that are supplemented by the inclusion of sample data acquired and digitised from real scenarios, through for example, texture mapping techniques. Mixed Reality is thus the overall framework that includes the continuum transitions from RE, to AR, passing through AV and towards VE.

On the user interaction side of the equation within immersive Mixed Reality environments, standard input devices such as keyboards or mice are useless, since they distract the user from the task at hand, thus creating a severe cognitive seam within the interaction process. On the other hand, traditional input devices used in virtual environments, such as the data glove or a 3D mouse with 6 degrees of freedom (6 DOF), introduce undesirable complexity in the tracking technology, or add "strange" gadgets to the user's workspace, with whom she's not daily accustomed. To face this problem, Kato [Kato 01a] have proposed Tangible Interfaces for Augmented Reality, as a new approach for the design of interactions in that context. According to the author, Tangible Interfaces, "are those in which 1) each virtual object is registered to a (tangible) physical object; and 2) the user interacts with virtual objects by manipulating the corresponding tangible object". Tangible Interfaces are described in the literature as being intuitive, since physical object manipulations are mapped to virtual object operations. If the mapping is one-to-one, we classify the interface as a space-multiplexed one. A time-multiplexed interface is the one where a physical object may correspond to different functions, at different points in time.

The related issue of hand and body gestures recognition have been tackled by using two different approaches: an image reference frame-based history approach [Dias 02a], where the bitmaps or pixmaps of the gesture path in the image reference frame are maintained and managed

² HCI – Human Computer Interaction

to derive gestures, or a world reference frame-based trajectory approach [Kopp 03], where the vision system tracks the evolution of the 6 degrees of freedom pose of the hand (its kinematics law) in the world reference frame, to compute the gestures.

Since 1996, AR and its applications, have also been researched by a group of FhG-IGD³, in areas like repair and maintenance of complex facilities, interior and exterior architecture, as well as in medical scenarios, such as needle biopsy and minimal invasive treatment, combining Computed Tomography (CT), Magnetic Resonance (MR) and Ultrasound (U) imaging, with the real scene, enabling new means of interaction of the physician with the patient [Reiners 98]. More recently, ARVIKA [Friedrich02], a large initiative funded by the German Ministry of Education and Research, is developing Augmented Reality technology and applications in the areas of product development, production and service in the automotive and aerospace industries, for power and processing plants and for machine tools and production machinery. This initiative includes some of the world's largest European and global players in the automotive and aerospace industries. ARVIKA efforts are focused in mobile use of AR in industrial applications, involving the fusion of real and virtual information, the delivery of such information as a function of context and multi-modal input/output, including speech, Tangible Interfaces and hands-free operations. The general problem of satisfactory user tracking and object registration is being investigated, using both fiducial markers on the environment and marker-less tracking.

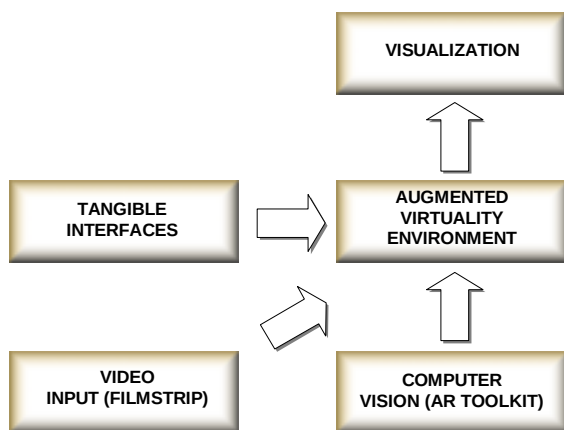


Figure 1 System Architecture.

3. SYSTEM ARCHITECTURE

To deploy our new user interaction paradigm for image frames of digital video, a number of modules comprise our proposed system, referred to as FilmstripWorks (Figure 1).

The **Computer Vision** Module includes a lower layer of Video Capture, which feeds into **AR Toolkit** (the Computer Vision and Image Processing layer [Kato 01b]),

³ Fraunhofer-Institute for Computer Graphics, www.igd.fhg.de

which is able to recognize and track, by computer vision means and in real-time, the 3D pose (position and orientation) of fiducial markers centres, relatively to the virtual camera reference frame, whose mathematical model matches the real video camera capturing live images. In order to provide the required user interaction, a dedicated system module was developed, which comprises real-time **Tangible Interface** Gesture Recognition and Virtual Object Transformations (picking, translation and scaling of image frames). Our system requires the avail-



Figure 2 FilmstripWorks interaction window (top), with multiple image frames that were picked, dragged and zoomed. Detail showing frame numbers and the corresponding time stamps of the filmstrip (bottom).

ability at program start-up of a **Filmstrip**, produced by a third party system. The system then maps the filmstrip band in the lower layer of the visualization window (see Figure 2). The upper zone of this window, corresponds to the visualization and interaction area, where the user can edit (pick, drag, drop, delete, zoom, play the video sequence starting from a frame, browse the filmstrip) various image frames (our scene objects), selected from the filmstrip, using our technique of marker-based free-hand gesture detection. The lower zone or the window, depicts the frame number and the time stamp of each frame of the filmstrip.

4. SYSTEM CONFIGURATION

The typical hardware and software platforms required by our system, are as follows:

- Hardware - CPU: HP Pentium IV 2.5 GHZ; 1 G byte RAM; Graphics card: NVIDIA GFORCE4 128 Mbytes; Webcam (input video with 640x480 pixel);
- Software - MS Visual C++ 6.0 enterprise edition; Ar-Toolkit 2.5.2; Video Input - Direct X 8.1; Graphics - Open GL and Open VRML;

- Indoor Tracking method - Computer vision-based approach (ArToolkit) with sparsely placed fiducial markers.



Figure 3: *Magic Rings* and *Magic Bracelets* enable the manipulation, through hand and finger gestures, of a filmstrip.

5. FREE-HAND AND FINGER GESTURING

Users generally require real-time user interaction in Mixed Reality (and more specifically, in Augmented Virtuality, as in our case). Having this in mind, we have developed Tangible Interfaces as intuitive as possible and we have come up a series of tools, similar to the ones found in [Billingham 01] [Kawashima 01] or [Dias 02b], which are suitable for our user interface requirements. One of these tools is referred as Magic Ring (see Figure 3) and the other, the Magic Bracelet, which have a specific marker attached. As a visual aid, when one of these markers is recognized by the system, an identification text will be displayed on top of it. Instead of registering virtual objects to real ones as in the mentioned works, in the present work we add novel uses for these interfaces, since they are the only physical means of user interaction that will trigger the functionalities of the system. This is achieved by detecting the trajectories of the centres of the fiducial markers attached to the Tangible Interfaces (as provided by ArToolkit), thus deriving the free-hand and finger gestures of the user.

Both the activation of system functions and the tracking of the markers, are performed in the 2D visualization window domain, through simple topological analysis between the projection, in the same window, of the visible markers centre vectors. This analysis consists in

evaluation of simple mathematical relations between the mentioned centre vectors. The current implementation assumes a viewport of 1024x768 pixel that occupies the full screen, with a coordinate system that has its origin at the left upper corner, where the y' axis points downwards. The input video source window of 640x480 pixel is mapped into this viewport.

Given vectors $\mathbf{a}$ and $\mathbf{b}$, corresponding to say, the projected centres in the visualisation window of two fiducial markers, we can easily compute, respectively, their Cartesian distance, $dist(\mathbf{a}, \mathbf{b})$, the minor angle between them, $(\mathbf{a} \wedge \mathbf{b})$, and their mid point, $\mathbf{P} \equiv (\mathbf{P}_x, \mathbf{P}_y)$, through:

$$dist(\mathbf{a}, \mathbf{b}) = \sqrt{(\mathbf{a}_x - \mathbf{b}_x)^2 + (\mathbf{a}_y - \mathbf{b}_y)^2}$$

$$(\mathbf{a} \wedge \mathbf{b}) = \arctan\left(\frac{\mathbf{a}_x - \mathbf{b}_x}{\mathbf{a}_y - \mathbf{b}_y}\right)$$

$$\mathbf{P}_x = \min(\mathbf{a}_x, \mathbf{b}_x) + \left| \frac{(\mathbf{a}_x - \mathbf{b}_x)}{2} \right|$$

$$\mathbf{P}_y = \min(\mathbf{a}_y, \mathbf{b}_y) + \left| \frac{(\mathbf{a}_y - \mathbf{b}_y)}{2} \right|$$

5.1 Activation of System Functions⁴

The activation of a given system function is highly dependent in the degree of proximity between two specified markers associated to the Tangible Interfaces (Magic Ring and Magic Bracelet). This degree of proximity is approximated by the evaluation of $dist(\mathbf{a}, \mathbf{b})$, where $\mathbf{a}$ and $\mathbf{b}$ are the projected visible marker centres in the visualisation window, such that the markers are not mutually occluded within a certain tolerance (which is dependent on the window dimensions). Each projected marker centre will in fact behave as if it was a cursor controlled by a specific finger gesture. Since the user has a total of six markers (A, B, C, D, F and |), he will control a total of six cursors in the visualisation window. In **Table 1** we list the available system functions and the correspondent gestures that trigger them.

⁴ The + operator is, in this context, overloaded to a physical operation of “proximity” between the marker centres positions.

System Functions and Combination of Markers	Activation Gesture
1: Pick a frame or Drop a frame: F + D	
2: Resize a frame (zoom): (B + F) + (A + D)	
Stop resizing a frame: A + B	
3: Delete a frame from the main window: A + D	
4: Start or Stop Browsing through the filmstrip: D + C	
5: Start playing the filmstrip: A +	
6: Stop playing the filmstrip: B + F	
7: Jump to the Start of the filmstrip: F + C	
8: Jump to the End of the filmstrip: B +	

Table 1 Different gestures that trigger system functions.

5.2 Tracking of Gestures

Once a specific system function is activated, such as enabling image browsing in the filmstrip, we need to track the kinematics of the projected markers centres in the visualization window, so that we can map these kinematics laws into parameters of transformation matrixes, which are then applied to objects we intend to manipulate (image frames of the filmstrip).

5.3 Picking and Translation

Picking (or selecting) and translating (or dragging) objects (image frames) are dependent on one or two visible markers. These gestures have an analogy with the action of pickling and grabbing an object with the right index and thumb fingers (**Table 1.1**). The picking action itself is performed in Open GL, with the middle point of two markers centres, which must be inside the picked image frame. The translation option is executed by tracking the trajectory of the middle point between markers centres, or, in the case one of them is not visible, of the marker centre of the one that is visible.

5.4 Image Zooming

The gesture defined for this feature was largely influenced by the practice in cinema and photography of framing a scene, by the definition of the corners of the frame with both hands (**Table 1.2**). By approximating or separating these corners, a previously selected filmstrip frame is reduced or amplified. The scale factor applied to the filmstrip frame, is a function of the variable distance between two characteristic points. The efficiency of this function depends on the number of visible markers. It is activated with the necessary presence of four markers in the screen, but the tracking of the kinematics change of the zooming parameters, does not depend on the visibility of all four. The two characteristic points, **Point 1** and **Point 2**, are evaluated in the following way:

1. With four visible markers: **Point 1** - The middle point between the left index and the left thumb fingers projected marker centres; **Point 2** - The middle point between the right index and the right thumb projected marker centres;
2. With three visible markers: **Point 1** - The middle point between the index and thumb fingers projected visible marker centres, of one hand; **Point 2** - The projected visible marker centre of the other hand;
3. With two visible markers: **Point 1** and **Point 2** - The projected visible marker centre of each hand.

5.5 Image Browsing

With this functionality, the user is able to browse through the filmstrip, at a variable rate of frames per second. The gesture that activates this feature, just approximates the right thumb to the left hand pulse (**Table 1.4**). In the visualization window area, centred on the left pulse marker centre, a “browsing” graphical interaction pops-up, as depicted in **Figure 5**. It's a kind of a circular section with three holes that match the visible marker centres of the left index and thumb fingers and of the left pulse. This graphical interaction widget, is able to rotate relative to its centre, in the clockwise or counter-

clockwise directions. With this interaction a user is able to browse the filmstrip frame by frame or, to play the filmstrip forwards or backwards, at an interactively defined rate, much in the same manner as with a Jog-Shuttle wheel of a non-linear video editor. To support this specification, we need to identify two vectors, from the visible marker centres, $\mathbf{a}$ and $\mathbf{b}$, in visualization window space, compute the difference vector, $\mathbf{b} - \mathbf{a}$, evaluate the angle between this vector and the yy' axis, and study the kinematics change of this angle through time. This technique will enable us to compute both the browsing direction as well as the browsing rate. After being activated, the instantaneous change of the parameters of this system function, re-quires the visibility of two specific markers, but its kinematics change in time, does not require the visibility of none of them, as long as there is no need to change the browsing rate and direction or issue another system function. The algorithm goes like this:

- Let $\mathbf{a}$ e $\mathbf{b}$ be the projected marker centres in study.
- Let yy' be the ordinate axis.
- Let $angle = \arctan\left(\frac{\mathbf{b}_x - \mathbf{a}_x}{\mathbf{b}_y - \mathbf{a}_y}\right)$
- Store the sign of $angle$
- Let $angle = \|angle\|$

Once $angle$ is computed, it is compared against a series of predefined limit angles. From this comparison, different browsing rates will be computed and applied to the filmstrip (a larger angle will trigger a larger browsing rate, a smaller angle can just operate a frame-by-frame browsing). The browsing direction is given by the sign of $angle$ (evaluated before the modulus operation): a positive and negative signs correspond, respectively, to forward (clockwise) and backward (counter-clockwise) browsing.

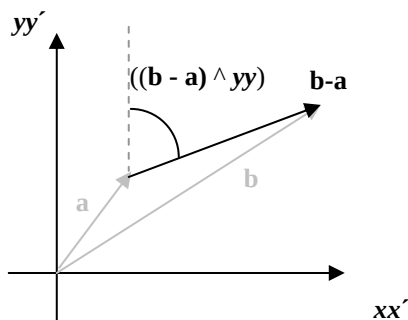
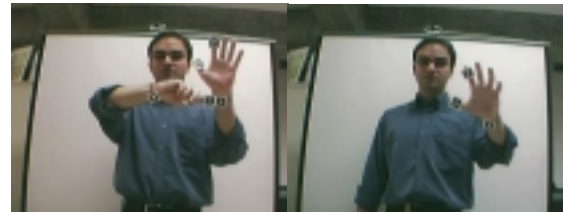


Figure 4 Determining the browsing direction and the browsing rate.

6. CONCLUSIONS AND FUTURE DIRECTIONS

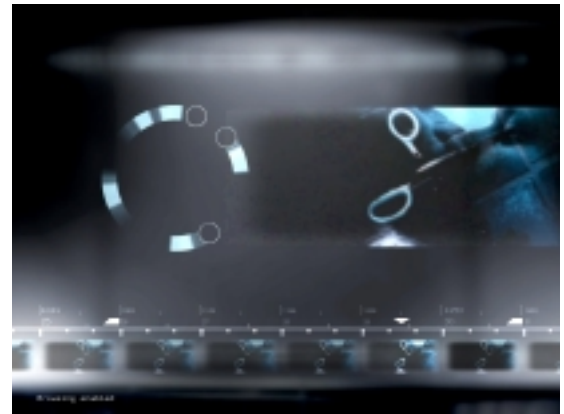
This system presented in this paper describes a general and basic Augmented Virtuality platform that can be ap-

plied in a technical viable way, to interactive manipulation of digital images obtained from a video sequence, previously decomposed in a sequence of frames, or filmstrip. The modular system functions were presented and



(a)

(b)



(c)

Figure 5: Browsing the filmstrip: (a) activation gesture; (b) browsing gesture; (c) visualization result.

several issues relevant for this type of Mixed Reality applications were raised, namely, the system architecture, the gesturing of Tangible Interfaces and the system functions. We have chosen the use case of video editing as a “proof of concept” to show that, given the recent advances in vision tracking algorithms, namely, the deployment of the AR Toolkit library and the availability of faster video cameras, CPU's and 3D graphic boards, marker-based free-hand gesture user interfaces, are technically viable for image manipulation. This does not contradict the fact that current desktop-based systems for video editing with 2D mouse interaction performs perfectly adequately and, in fact, we don't claim that our system will eventually replace the former. By conducting structured usability evaluation with a panel of users, on this type of usage scenario, which is part of our future plans, we are hoping of getting valuable feedback on the issue of the applicability of new gesture-based HCI modality for this type of task, and from there, we can start studying the more general applicability of this technique to other fields.

As a generalisation of our work, we have demonstrated the feasibility of using marker-based tracking, in the framework of AR Toolkit, to simplify the task of hand gesture recognition using a trajectory approach. With our technique we project the 3D coordinates of the visible marker centres to 2D coordinates of a given visualization

window, enabling the use of two Magic Rings and one Magic Bracelet in each hand, to control six new cursors projected on a given visualization window of an AV environment. By simple topological analysis between visible markers centre vectors distances, mid points and angles, we enable gesture recognition which can trigger functions of the system and, subsequently, of specific gesture-based user interaction to operate a given active functionality. This interaction affects the selection, deselection, shape, scale factor, position and visualization of scene objects (filmstrip frames in our case), enabling also filmstrip browsing at a user-defined rate. The gestures that activate and operate system functions, were developed as a compromise between the ease of usage and the limitations of the marker-based approach. Since the system functions and operations are triggered by the relative proximity and orientation of the visible marker centres, special care was given to the fact that the number of markers (six in total), could provide a sufficient number of possible proximity/orientation combinations for the definitions of HCI gestures, that could be conceptually close to the task at hand. When creating these gestures the authors have not yet considered the possible adoption of one the available hand gesture languages used by deaf persons, but this could be a promising line of research for the future. There are currently available research tools for automatic sign language recognition and synthesis (e.g. for the American Sign Language and British Sign language), which include formal methods (such as using the XML markup language MURML) to describe coordination of verbal and gestural constituents in deictic utterances [Wachsmuth 03]. MURML could be applied in our technique, as a standard way to codify the recognized gestures, which would enable a convenient integration with other modalities, such as speech recognition. With the support of MURML, our system could also be used as a platform to recognize, describe and translate different sign languages.

As another result, our gesturing-based control of cursors (just like mouse cursors), opens the possibility to provide user interaction with custom made “augmented” menus, including menu manipulation, menu items browsing and selection [Dias 03a].

Several other future directions can be envisaged for this kind of system, towards the enhancement of the new paradigm of Tangible Augmented Virtuality, for Interactive Visualisation and Editing of Digital Imaging. Concerning the system architecture, enhancements could be made in sense of immersion and the cooperative support of the system, to increase the perception of Augmented Virtuality, that is, of seamless fusion of virtual environments with real-live video and image input, both in standalone and collaborative working scenarios. In fact, as a future evolution, we could support, multiple collaborators (such as in [Dias 02c]), where collaborative image interaction working sessions could take place amongst local and remote users, immersed on different collaborative environment modalities over the Internet 2 backbone. As concluded by preliminary technical tests to the system, we need to adjust the camera tracking methods,

given various indoor lighting conditions. Hybrid indoor virtual camera tracking could also be developed, using a combination of vision-based approaches (both marker-based, as in AR Toolkit, and marker-less), ultra-sound sensors positioned along the workspace area, and accelerometers located on a pair of gloves or on a wearable unit dressed by the user. This approach could be a possible solution for the accurate registration problem, enabling, for example, the user to walk freely within a certain working area, while experiencing augmented virtual worlds.

Related use cases of rich image manipulation, as with gestural-based HCI modality in medical diagnosis applications, such as X Ray image analysis, or large image database browsing, could be well applied by our method, in the direction of the potential generalisation of our system to other fields.

7. ACKNOWLEDGEMENTS

The authors would like to thank Rafael Bastos for advice regarding marker tracking and Marisa Aurélio, Ana Rita and António Calado for the authoring of the filmstrip decomposition code, all from ADETTI, Portugal.

8. REFERENCES

- [Azuma 97] Azuma, R. T., “A survey of augmented reality”, in *Presence: Teleoperators and Virtual Environments*, 6, 355-385, 1997
- [Behringer 99] Behringer, R., et al, editors, *Augmented Reality, Placing Artificial Objects in Real Scenes*, Proceedings of 1st International workshop on Augmented Reality - 1998, A. K. Peters ed., pp xi-xx, 1999.
- [Billinghurst 01] Billinghurst, M., Kato, H., Poupyrev, I., “The MagicBook: A Transitional AR Interface”, in *Augmented Reality: the Interface is Everywhere*, SIGGRAPH 2001 Course Notes 27, 2001
- [Caudel 92] Caudel, T., Mizell, D., “Augmented Reality: An Application of Heads-up Display Technology to Manual Manufacturing Processes”, *Proc. Hawaii International Conference on Systems Sciences*, Maui, Hawaii, IEEE press, pp 659-669, January 1992.
- [Dias 02a] Dias, J.M.S., Santos, P., et al, “MIXDesign, Tangible Mixed Reality for Architectural Design”, SIACG 2002, Universidade do Minho, Portugal, June 2002
- [Dias 02b] Dias, J.M.S., Santos, P., et al, “Tangible Interaction for Conceptual Architectural Design”, “ART02, First IEEE International Augmented Reality Toolkit Workshop”, ISMAR 2002, IGD, Darmstadt, Germany, September 2002
- [Dias 02c] Dias, Cooperative Work Support And Automatic Design Verification In Architecture And Building Construction, eSM@RT 2002, University of Salford, United Kingdom, 19 - 21 November 2002.
- [Dias 03a] Dias, J.M.S. Santos, P., Bastos, R., “Gesturing with Tangible Interfaces for Mixed Reality”, *Gesture Workshop*, Genova, April 2003

- [Dias 03b] Dias, J.M.S. Jorge, J., Carvalho, J., Santos, P., Luzio, J., “Usability Evaluation of Tangible User Interfaces for Augmented Reality, submitted to “ART03, Second IEEE International Augmented Reality Toolkit Workshop”, ISMAR 2003, Japan, October 2003
- [Friedrich 02] Friedrich W. “ARVIKA – Augmented Reality for Development, Production and Service”, Proceedings of the ISMAR 2002, IEEE and ACM International Symposium on Mixed and Augmented Reality, Darmstad, pp 3-4, 30th Sept-1 Oct, 2002
- [Kato 01a] Kato, H., Billinghurst, M., Poupyrev, I., “Tangible Augmented Reality”, in Augmented Reality: the Interface is Everywhere, SIGGRAPH 2001 Course Notes 27, 2001
- [Kato 01b] Kato, H., “Developing Applications with AR-Toolkit”, SIGGRAPH 2001 Course Notes 27, 2001.
- [Kawashima 01] T. Kawashima, K. Imamoto, H. Kato, K. Tachi-bana, M. Billinghurst, “Magic Paddle: A Tangible Augmented Reality Interface for Object Manipulation”, Proc. on ISMR2001, pp.194-195, 2001
- [Kopp 03] Koop, S. et al., “Imitation games with artificial agent: From mimicking to understanding shape-related iconic gestures, University of Bielefeld, Germany, Gesture Workshop 2003, April 2003, Genova, Italy
- [Milgram 99] Milgram, P., Herman, C. J., “A Taxonomy of Real and Virtual World Display Integration”, in Mixed Reality, Merging Real and Virtual Environments, Ohmshda & Springer-Verlag, pp 5-30, 1999
- [Reiners 01] Reiners D., Stricker, D., Klinker, G. “Augmented Reality: Applications”, Computer Graphics Topics, reports on Computer Graphics, 4/98 Vol 10, pp 36-38.
- [Wachsmuth 03] Wachsmuth I. et al, “Deixis in Multimodal Human Computer Interaction: An Interdisciplinary Approach”, University of Bielefeld, Germany, Gesture Workshop 2003, April 2003, Genova, Italy

ARK multi-utilizador

Pedro Pereira

Nuno Matos

Luís Grave

Centro de Computação Gráfica (CCG)

R. Teixeira de Pascoais, 596

4800-073 Guimarães

{Pedro.Pereira, Nuno.Matos, Luis.Grave}@ccg.pt

Adérito Marcos

Centro de Computação Gráfica (CCG)

R. Teixeira de Pascoais, 596

4800-073 Guimarães

Aderito.Marcos@ccg.pt

Universidade do Minho

Escola de Engenharia

Departamento de Sistemas de Informação

Campus de Azurém

4800-058 Guimarães

marcos@dsi.uminho.pt

Abstract

This paper present a monitor based prototype for visualization and interaction of an Augmented Reality (AR) system. It had been recently developed at CCG and demonstrated on the conference SIACG2002 at Guimarães, Portugal.

ARK – augmented reality kiosk - is a set-up based in the prototypes developed in the European project Virtual Showcases where direct interaction has been added. A normal monitor and a half-silvered mirror constitute the usual set-up of the kiosk

By integrating a half-silvered mirror and a black virtual hand, our solution solves the occlusion problem that normally occurs when a user interacts with a virtual environment displayed by a monitor or other projection system. Conceived with low monetary resources this portable solution can be deployed in different application contexts as for instance culture heritage. In this paper it will presented an extension of the solution to a multi-user platform for a Portuguese museum.

Keywords: Augmented Reality, Virtual Reality, Augmented Environments, Visualization, Interaction, multi-user, half-silvered mirror

1. INTRODUÇÃO

Ao contrário da Realidade Virtual (RV), a Realidade Aumentada (RA) não cria um ambiente totalmente artificial. Esta consiste na geração de imagens tridimensionais por computador que são projectadas no mundo real, adicionando assim informação virtual à percepção humana. O potencial aplicacional desta tecnologia estende-se a diversas áreas, desde a indústria ao turismo.

No que diz respeito à área do património e herança cultural em particular, estas tecnologias são actualmente utilizadas na preservação e disseminação de informação histórica e cultural. Considerando a título de exemplo um artefacto arqueológico fragmentado ou deteriorado, é possível com recurso à RA gerar virtualmente a

imagem correspondente na sua totalidade, projectando a mesma sobre o objecto real. Deste modo, o utilizador tem a possibilidade de visualizar o artefacto na sua forma original.

Em termos tecnológicos existem duas abordagens possíveis para criar um ambiente de realidade aumentada: um capacete (see-through Head-Mounted Display) ou um sistema de projecção [Aliaga 1997].

No protótipo *Augmented Reality Kiosk* – ARK, apresentado neste artigo, o sistema de projecção utilizado baseia-se num monitor, tornando esta solução acessível quando comparada com outros sistemas de projecção. Este trabalho surge no contexto do projecto europeu *VirtualShowcases* (www.virtualshowcases.com) que consiste no desenvolvimento de um novo tipo de sistema de visualização aumentada para utilização em ambientes museológicos.

O ARK assume presentemente a forma de um posto mono-utilizador, o qual disponibiliza uma componente de interacção em tempo real. No entanto, esta solução torna-se insuficiente em cenários em que se regista uma grande afluência de público. Perante este facto, o CCG encontra-se a investigar soluções que permitam estender a utilização do ARK para ambientes multi-utilizador.

No que se refere à estrutura do presente artigo, está dividida em três capítulos principais. O primeiro descreve os trabalhos em que se regista uma relação com o ARK, seguindo-se a apresentação do “ARK”, seu *setup* e solução encontrada para resolver o problema do efeito de oclusão que é comum em sistemas de projecção desta natureza. No capítulo “*Augmented Room*” apresenta-se a evolução do ARK para um sistema multi-utilizador.

2. PROJECTOS RELACIONADOS

Como já foi mencionado, o protótipo ARK põe em prática alguns dos princípios da tecnologia da solução inicial *Virtualshowcase*, adicionando a componente de interacção directa de objectos reais e virtuais através de uma solução de baixo custo.

O *setup* original *Virtualshowcase* é construído com base num material transparente (tal como o vidro), laminado com uma folha de prata semi-transparente. As faces laterais do *showcase* simultaneamente, transmitem a imagem dos objectos reais, assim como reflectir as imagens geradas pelo computador. Os objectos reais podem ser sobrepostos com suplementos virtuais, visualizados por imagens 3D ou 2D (estereoscópico ou monoscópico) reflectidas pelo *showcase* virtual (figura 1) [Bimber et al. 2001] [Bimber et al. 2002] [Bimber et al. 2002b].

A restrição deste sistema, no que concerne à interacção, implica que os utilizadores tenham um papel de observadores passivos perante a cena visualizada.

No projecto ARCADE, sistemas de grande projecção como a Mesa Virtual, foram usados em diferentes cenários na área da indústria automóvel e aeronáutica, no desenho e planeamento dos mesmos, possibilitando ao utilizador a interacção com os objectos em modo imersivo [Encarnação et al. 1999].

Este é um sistema de modelação 3D, desenvolvido no Fraunhofer Institute for Computer Graphics em Darmstadt, que visa integrar tecnologia de Realidade Virtual como por exemplo 6D-input, 3D-output e reconhecimento de gestos 3D, numa arquitectura CAD. A combinação destas técnicas permite uma modelação eficiente e precisa num Ambiente Virtual. [Amicis 2001]

Estes sistema, baseado em video tracking, torna-se limitado pelo facto de não acompanhar na totalidade os movimentos do utilizador. Por outro lado, o recurso à Mesa Virtual da BARCOTM, impede que o sistema seja facilmente transportável e implica avultados investimentos.

3. ARK

A solução ARK é apresentada sob a forma de quiosque, onde está instalado um Monitor de 21” invertido e um vidro semi-espelhado. Este último reflecte as imagens projectadas a partir do monitor e ao mesmo tempo é suficientemente transparente para que o utilizador consiga visualizar objectos reais que se localizam no interior do quiosque. A estrutura é de madeira com revestimento negro opaco para reduzir o nível de reflexão da luminosidade exterior, permitindo assim uma maior nitidez da imagem projectada.

A interacção ocorre directamente no interior do quiosque utilizando uma *CyberGlove* como se observa na figura um.

O efeito de realidade aumentada é gerada através da sobreposição do objecto virtual (na superfície espelhada) com o objecto real (debaixo do espelho). O grau de visualização dos objectos reais é controlado através de iluminação artificial, que ilumina apenas os objectos reais debaixo da superfície espelhada.

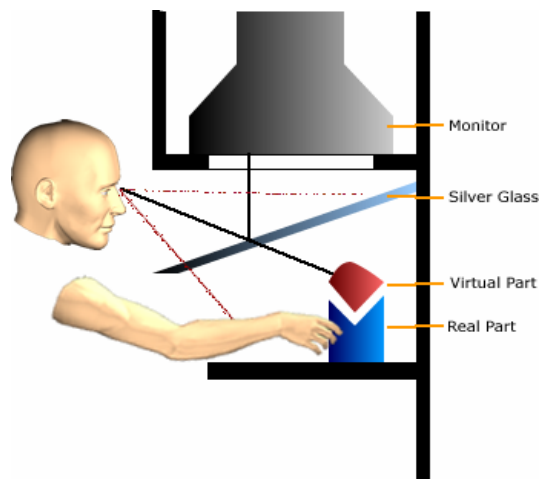


Figura 1: estrutura funcional interação do ARK

O ARK é por isso um sistema adaptável a vários tipos de cenários de utilização, facilitando a representação de cenários mistos (real e virtual) ou puramente virtuais.

O efeito de oclusão pode ocorrer quando a projecção dos objectos virtuais é ocultada por um objecto real ou vice versa [Wloka 1995] [Berger 1997] [Breen et al. 1996] [Balcisoy et al. 2000]. Vejamos um caso simples: uma imagem tridimensional está a ser projectada numa parede e pretendendo-se manipular um objecto 3D interactivamente com a mão. Ao colocar a mão dentro da área de projecção, o objecto virtual é escondido pela mão, mesmo quando o objecto se deva localizar logicamente em frente da mão. Estamos perante o chamado efeito de oclusão.

No ARK, as imagens projectadas no espelho ocultam a mão do utilizador. O problema foi solucionado, com a criação uma mão virtual (Figura 2), que sobreposta com mão do utilizador, cria na imagem projectada o efeito



Figura 2: Mão negra e mão real à esquerda, do lado direito o efeito pretendido depois de aplicado o paradigma da “Mão Negra”

de “buraco negro”. Deste modo a máscara gerada elimina partes dos objectos virtuais e torna a mão do utilizador visível. Por outras palavras temos uma mão completamente preta que sobrepõe-se à mão real [Fuhrmann et al. 1999] [Bimber et al. 2002].

Pelo facto do utilizador interagir directamente com o conteúdo virtual/real, a imagem gerada pode ocultar a mão do utilizador quando o objecto virtual se encontra à frente da mesma. Mas com a utilização do paradigma da mão virtual negra, o utilizador visualiza sempre a sua mão no interior do ambiente aumentado, através do buraco deixado pela mão negra. Na figura três é ilustrado o problema da oclusão que ocorria no ARK.



Figura 3: exemplo do efeito de oclusão ocorrido no ARK

A luva utilizada é uma *CyberGlove™* com 22 sensores espalhados pelos dedos da mão que detectam com grande precisão qualquer tipo do movimento em tempo-real.

A complexidade da fase de calibração é uma das limitações actuais do ARK. A sobreposição da mão virtual (mão negra) com a mão do utilizador tem que ser exacta, a qual é difícil conseguir com o tracking flock of bird utilizado. A precisão do sistema de localização está dependente das condições electromagnéticas das instalações, as quais podem alterar os valor de tracking muito facilmente [Redert et al. 1997] [Summers et al. 1999]. Como acção futura será testar um tracking por vídeo no kiosk.

Usando a *CyberGlove*, o utilizador pode manipular ambos os objectos reais e virtuais, criando um maior efeito de realismo e imersão. O campo de acção da mão é limitado ($0.76 \times 0.63 \times 0.30$ metros), sendo o espaço do quiosque suficiente para os propósitos definidos [Grave et al. 2000].

O sistema de *Tracking* utilizado no ARK integra dois sensores de posição espacial: um na *CyberGlove* e outro nos *ShutterGlasses* (óculos-stereo). Ambos os sensores permitem apurar com precisão a localização necessária para projectar os objectos virtuais na cena aumentada de acordo com o posicionamento e orientação da cabeça do utilizador, bem como a posição da mão e dedos. A componente de localização espacial é designado por *Tracking System Ascension Flock of Birds™*. O computador utilizado é um *SGI Octane* e o *Software* de Realidade Virtual é o *Virtual Design2™* da VRCOM. Optou-se por este motor de render dado que disponibiliza uma API (*Application Programming Interface*) bastante completa que permite com grande facilidade instalar e usar vários dispositivos de

interacção, assim como definir os respectivos comportamentos em resposta à interacção do utilizador. Neste momento está a ser implementada uma solução para PC usando como base o sistema de realidade virtual *Maverik* (<http://aig.cs.man.ac.uk/maverik/>).

O ARK revela-se um sistema intuitivo e de fácil aprendizagem, que faz uso de simples metáforas para interacção para manipular os objectos virtuais. Simulando os movimentos de uma tenaz com os dedos indicador e polegar é conseguido o “drag and drop” dos objectos virtuais. O grau de imersão conseguido com o sistema é total.

O ARK permite apresentações completamente virtuais, ou apresentações mistas com a sobreposição de imagens virtuais sobre objectos reais.

Um exemplo concreto da potencial aplicação do kiosk é na área da prototipagem virtual, onde é permitido visualizar e manipular objectos virtuais em fase de criação.

4. AUGMENTED ROOM

O *augmented room* é uma evolução do ARK numa versão multi-utilizador. Este sistema permite no máximo até oito utilizadores podendo quatro deles interagir com os ambientes aumentados, ficando os restantes quatro condicionados ao papel de observadores.

O nome *Augmented Room* surge pelo facto de este se assemelhar a uma sala com janelas, bem como por permitir a utilização da tecnologia de Realidade Aumentada. Ao contrário de uma sala normal, o utilizador visualiza do exterior para o interior.

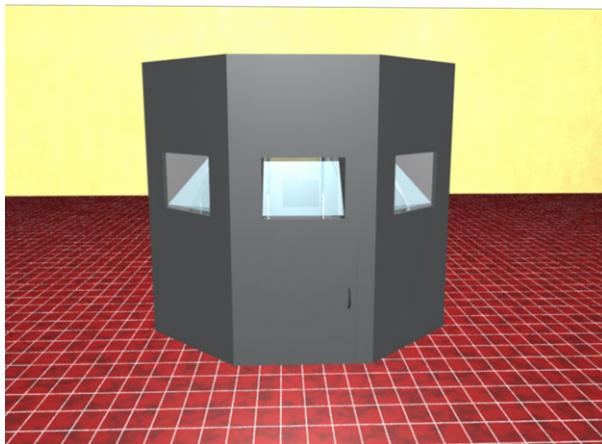


Figura 4: simulação 3D da Augmented Room

Esta “sala” trata-se de uma estrutura em forma octogonal, tendo cada uma das faces uma janela de visualização, através da qual é possível visualizar o seu interior, tal como está ilustrado na figura 4. Tal como no ARK, para visualizar os objectos virtuais em conjunto com os objectos reais, recorreu-se ao vidro semi-espelhado. A possibilidade de regulação do grau

de transparência ou espelhamento também está patente na *Augmented Room*.

As dimensões da estrutura foram projectadas de forma a poder ser exposto uma maquete de uma pira funerária romana no centro da mesma com o objectivo de permitir ao utilizador ver o objecto real e seguidamente visualizar os objectos virtuais. Desta forma a solução estudada prevê a utilização de uma luz central, a qual iluminará com maior ou menor intensidade os objectos reais, diminuindo ou aumentando consequentemente a reflexão das imagens geradas por computador.

Como no ARK, as imagens reflectidas em cada um dos vidros semi-espelhados provêm dos monitores que se encontram na parte superior da estrutura (figuras 6), ou seja, as imagens reflectidas nos vidros são projectadas a partir dos ecrãs dos monitores, não recorrendo deste modo a uma segunda superfície espelhada. Este facto converte-se num problema visto que a falta de uma segunda superfície de reflexão faz com que as imagens reflectidas sejam apresentadas em posição inversa à dos monitores. Para resolver este problema torna-se necessário inverter a imagem do monitor de modo que a reflexão mostre uma imagem e posicionamento correcto da mesma perante os olhos do utilizador. Tome-se como exemplo a palavra “AMBULÂNCIA” colocada de forma invertida na frente do veículo para permitir que a palavra em causa seja bem interpretada no retrovisor do veículo que segue à sua frente. Este paradigma será integrado como solução efectiva no vídeo *stereo* a ser gerado no *Software Alias|Wavefront MAYA 4.5* (figura 7).

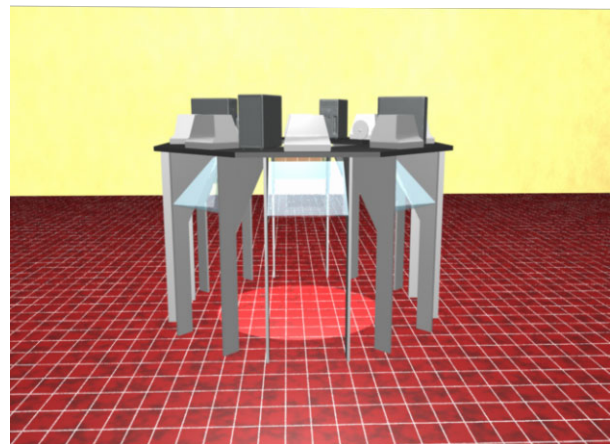


Figura 6: simulação 3D da estrutura sem as paredes

Simultaneamente, os vidros encontram-se inclinados com um ângulo aproximado de 45° em relação ao monitor, para que a imagem reflectida esteja em conformidade com a altura e posição do utilizador. Esta inclinação pode ser alterada para se adaptar ao utilizador podendo variar entre os 30° e 55° em relação ao monitor. A distância a que se encontra o vidro do monitor é também ajustável (ver Figura 6).

O princípio de visualização do *Augmented Room* é idêntico ao ARK. Como esta sistema foi idealizado para espaços de exposições ou entretenimento a componente de interacção não é considerada.

Contudo o sistema é aberto o suficiente para acrescentar interacção. Pelo menos quatro dos postos de visualização é permitido utilizar dispositivos de interacção, como seja por exemplo um joystick. Como o sistema alimenta os oito monitores a partir de 4 PC existem algumas limitações de Hardware em termos de conexão de dispositivos de interacção.

No cenário específico da pira funerária acima referido está-se a desenvolver um jogo virtual interactivo onde os utilizadores podem reconstruir o cenário virtual da "Pira Funerária" com base na apresentação que vêm previamente (figura 7).

Esta apresentação consiste numa recriação do cortejo fúnebre romano onde objectos arqueológicos são apresentados dentro da *Augmented Room*, em stereo activo. Como o cortejo fúnebre é um processo composto por diferentes acções foi desenvolvido um vídeo em stereo activo. Como tal não é possível apresentar uma sobreposição dos objectos virtuais sobre o reais neste caso concreto.

Este sistema pode no entanto suportar cenários de realidade aumentada.

A interacção em grupo também é estimulada pela participação em simultâneo de dois utilizadores, onde um está a interagir e o outro está a dar dicas e estratégias para o jogo.

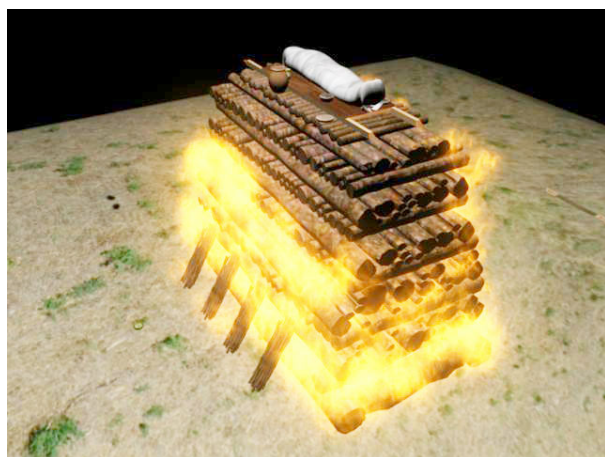


Figura 7: Primeira apresentação para a *Augmented Room*.

O processamento das imagens é efectuado em oito pontos distintos, de forma a garantir a visualização correcta da cena a partir dos oito pontos de visualização possíveis. Os utilizadores têm a sensação de estar a espreitar para dentro de uma sala escura onde se está a ver um ambiente aumentado.

É ainda de referir que para que a imersão seja mais acentuada prevê-se recorrer à visualização em *stereo* activo, dando desta forma a sensação de profundidade.

Num sistema multi-utilizador com vários pontos de vista distintos, é necessário obter a localização da cabeça do utilizador [Redert et al. 1997]. Numa fase posterior, será incluído no sistema um dispositivo para obter o posicionamento do observador. A evolução do ARK para a *Augmented Room*, é um avanço significativo pois instancia-se um sistema mono-utilizador para um sistema multi-utilizador.

5. CONCLUSÕES E TRABALHO FUTURO

A aplicabilidade da RA na área da disseminação do património e herança cultural é já uma realidade podendo vir a ser ajustada a vários cenários histórico-culturais meramente virtuais ou aumentados (objectos virtuais sobrepostos aos objectos reais).

Neste tipo de sistemas tanto a componente lúdica como a educativa podem ser exploradas de forma a cativar os utilizadores pelo modo de visualização e interacção com o sistema.

As soluções apresentadas no artigo ainda se encontram em fase de prototipagem, mas já é possível vislumbrar a mais valia que este tipo de soluções pode trazer. No entanto, para garantir o seu sucesso junto das instituições museológicas é necessário que estas sejam atractivas economicamente a nível de concretização e manutenção, passando sempre por um sistema multi-utilizador.

O trabalho futuro a realizar-se no ARK passa pela implementação de um novo sistema de localização. Contudo o maior desafio com que a equipa CCG se depara é sem dúvida a implementação do primeiro protótipo da *Augmented Room* num museu português. Ainda neste contexto desenvolver algum trabalho de pesquisa no sentido de aperfeiçoar a *Augmented Room* para ambientes de multi-utilizador com possibilidade de interacção em grupo.

6. AGRADECIMENTOS

Os autores agradecem ao Regulamento de Bolsas da Associação CCG/ZGDV – Centro de Computação Gráfica e também ao consórcio do VirtualShowcases, os quais financiaram a realização dos trabalhos descritos neste artigo

7. REFERÊNCIAS

[Amicis 2001] De Amicis R., Fiorentino M., Stork A.; »Parametric Interaction for CAD application in Virtual Reality Environment «; In *Proc. of 12th International ADM Conference on Design Tools and Methods in Industrial Engineering*, Rimini, Italy, 5.-7. September 2001.

[Aliaga 1997] Aliaga, D.G., "Virtual Objects in the Real World", in *Communications of the ACM (CACM)*, Vol. 40. No. 3., March 1997, pp. 49-54.

- [Balcisoy et al. 2000] Balcisoy, S., Torre R., Fua, P., Thalmann, D., "Interaction Between Real and Virtual Humans: Playing Checkers", In *Proc. Eurographics Workshop on Virtual Environments*, Amsterdam, Netherlands, 2000.
- [Bimber et al. 2001] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M., "The Virtual Showcase", In *IEEE Computer Graphics & Applications*, vol. 21, no.6, pp. 48-55, 2001. New York: IEEE Computer Society Press ISSN: 0272-1716.
- [Bimber et al. 2002] Bimber, O. and Fröhlich, B. "Occlusion Shadows: Using Projected Light to Generate Realistic Occlusion Effects for View-Dependent Optical See-Through Displays", In proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02), 2002.
- [Bimber et al. 2002b] Bimber, O., Gatesy, S.M., Witmer, L.M., Raskar, R. and Encarnação, L.M. "Merging Fossil Specimens with Computer-Generated Information", in *IEEE Computer*, September, pp. 45-50, 2002.
- [Berger 1997] M.-O. Berger, "Resolving Occlusion in Augmented Reality: A Contour-Based Approach Without 3D Reconstruction", In *Proceedings of the Conference on Computer Vision and Pattern Recognition*, 1997.
- [Breen et al. 1996] Breen D., Whitaker R.T., Rose E., Tuceryan M., "Interactive and Automatic Object Placement for Augmented Reality", in *Computer Graphics Forum (Proceedings of EUROGRAPHICS'96)*, 15(3):C11-C22, 1996.
- [Encarnação et al. 1999] L. M. Encarnação, A. Stork, D. Schmalstieg, R. Barton III, "The Virtual Table - A Future CAD Workspace", in *Proceedings of Computer Technology Solutions conference (former Autofact)*, Michigan, Detroit, USA, September 13-19, 1999.
- [Fuhrmann et al. 1999] Fuhrmann, A., Hesina, G., Faure, F., Gervantz M.; Occlusion in collaborative augmented environments; *Computer & Graphics*, Vol. 23, N. 6; December 1999 (ISSN 0097-8493).
- [Grave et al. 2000] Grave L., Silva A. F., Escaleira C., Marcos A., "Ambientes Virtuais de Treino para Montagem de Cablagens Eléctricas", in *Proc. of 1º Workshop de Sistemas Multimédia Cooperativos e Distribuídos, COOPMEDIA 2000*, vol. 1, pp. 87-96. Portugal, Coimbra, 2000.
- [Kindratenko 1999] V. Kindratenko. *Calibration of electromagnetic tracking devices*. Virtual Reality: Research, Development, and Applications, 4:139--150, 1999
- [Redert et al. 1997] A. Redert, J.J. van Klaveren and E.A. Hendriks, "Accurate 3D eye Tracking for multi viewpoint systems," *Proceedings IWSNHC3DI*, 224-228, Rhodes, Greece, 5-9 September 1997.
- [Summers et al. 1999] Summers V., Booth K., Calvert T., Graham E., MacKenzie C.L., ; "Calibration for augmented reality experimental testbeds", in *Proc. of the 1999 Symposium on Interactive 3D Graphics*, 1999, Pages 155 - 162.
- [Wloka 1995] Wloka, M. M., Anderson B., „Resolving Occlusion in Augmented Reality”, in *Proc. of 1995 Symposium on Interactive 3D Graphics*, Monterey, Association for Computing Machinery: 5-12.

Sessão 5

COMUNICAÇÕES CURTAS

Técnicas de Realidade Virtual Aplicadas ao Ensino na Área da Construção

A. Zita Sampaio
Dep. Eng^a. Civil e Arq., IST
Av. Rovisco Pais, 1000 Lisboa
zita@civil.ist.utl.pt

P. Gameiro Henriques
Dep. Eng^a. Civil e Arq., IST
Av. Rovisco Pais, 1000 Lisboa
pgameiro@civil.ist.utl.pt

P. Studer Ferreira
Bolseiro POCTI/FCT

Sumário

A presente comunicação descreve a criação e a manipulação de um modelo virtual de interesse didáctico no domínio dos processos construtivos em Engenharia Civil. A aplicação desenvolvida corresponde ao modelo de uma parede dupla de alvenaria, uma das componentes de uma construção corrente. De modo a poder simular-se visualmente a construção da parede, o modelo geométrico gerado é composto por um conjunto de elementos representando cada uma das componentes construtivas. Por recurso a um sistema virtual, são aplicadas sobre o modelo da parede propriedades específicas próprias de um ambiente de simulação virtual. Por interacção directamente com o modelo, é possível controlar a progressão do processo construtivo da parede e aceder a informação relativa a cada elemento, nomeadamente, a sua composição e a fase da sua execução ou montagem em obra, de acordo com determinado planeamento. Demonstra-se, com este exemplo, como a tecnologia de realidade virtual pode ser vantajosamente utilizada na elaboração de material didáctico de grande interesse na formação na área de processos construtivos.

Palavras-chave

Novas tecnologias de ensino, realidade virtual, simulação visual de processos construtivos.

1. INTRODUÇÃO

A presente comunicação descreve um modelo físico evolutivo capaz de simular visualmente as fases construtivas e de admitir interacção nesse processo [Sampaio03]. O modelo virtual foi desenvolvido no âmbito de um projecto de investigação em curso no IST/ICIST - *Virtual reality in optimization of construction project planning* [Henriques02] [Henriques01].

A realidade virtual pode ser descrita como um conjunto de tecnologias que, apoiadas no uso de computadores, simulam a realidade existente ou a realidade projectada. Esta nova ferramenta permite aos seus utilizadores, a participação em ambientes tridimensionais, possibilitando-lhes que interajam com objectos virtuais. Uma das características mais importantes da realidade virtual, e que foi explorada neste trabalho, é ser interactiva [Vince98]. Isto é, a realidade virtual não se limita apenas a uma visualização passiva desse mundo mas permite que o utilizador possa interagir com ele (mover objectos, por exemplo). Adicionalmente, o mundo virtual responde em tempo real a tais acções.

Uma outra propriedade de grande interesse na realidade virtual é a sua capacidade de imersão. Isto é, permite que o utilizador, usando dispositivos específicos, tenha a sensação de se encontrar fisicamente presente num mundo imaginado. Num modelo de edifício mais completo que o apresentado, haveria naturalmente

interesse em desenvolver também esta vertente, a qual requer dispositivos mais complexos que os utilizados neste estudo.

O sistema de realidade virtual utilizado neste trabalho, o sistema EON [EON01], baseia-se numa programação por objectos usando a linguagem JAVA. O modelo tridimensional do cenário pretendido é definido, previamente, num sistema de modelação sendo, posteriormente, inserido no sistema de realidade virtual. Para o propósito desta aplicação, o modelo criado foi definido como um conjunto de objectos de modo a permitir, no ambiente virtual, a imposição de determinadas acções (translações, rotações, ocultação, alteração de textura, ...) a cada objecto do modelo de uma forma individualizada ou a determinado grupo de objectos. Definidas as relações e acções pretendidas para o modelo, o utilizador pode desencadear cada acção, de um modo interactivo, através de um dispositivo mecânico.

Por recurso a técnicas de realidade virtual, é possível obter uma representação animada do processo construtivo da componente de edifício analisada. A par da simulação visual é apresentada, conjuntamente, a informação relativa ao processo construtivo, útil aos profissionais usualmente envolvidos no projecto, nomeadamente, as precedências das actividades, a especificação do material

utilizado na fase visualizada e os seus detalhes construtivos.

Descrevem-se, de seguida, os processos de modelação de uma parede exterior de um edifício corrente e da associação de capacidades virtuais ao modelo criado com vista à exibição interactiva da sua edificação. O modelo de uma parede dupla em alvenaria, incluindo a estrutura de betão armado envolvente e os elementos de vão definidos na parede, foi criado por recurso a um sistema gráfico de modelação tridimensional de uso generalizado em gabinetes de projecto, o sistema AutoCAD [AutoCAD02]. O ambiente virtual foi imposto ao modelo através do sistema de realidade virtual referido [EON01].

2. CRIAÇÃO DO MODELO TRIDIMENSIONAL

Sendo o exemplo de aplicação apenas parte de uma construção, mas significativa na composição da mesma, procurou-se incluir no caso analisado os diferentes tipos de elementos estruturais, estéticos e funcionais.

Numa parede exterior de um edifício corrente é necessário considerar os elementos estruturais das fundações e do pórtico. Foram, assim, modeladas duas sapatas, uma viga de fundação, um pilar e uma viga superior. Os elementos do modelo são representados por paralelepípedos e relacionados de acordo com a sua usual disposição em obra, formando o modelo representado na Figura 1.

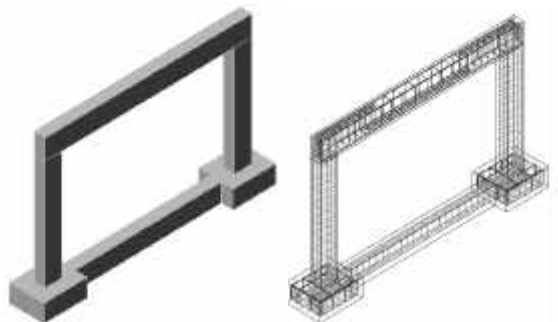


Figura 1: Fases da modelação dos elementos estruturais.

O material destes elementos estruturais é o betão armado. Assim, o modelo da estrutura deverá considerar ainda o modelo das armaduras. Os varões de cada armadura são representados como elementos tubulares de secção circular, com as adequadas dimensões e localização em relação às superfícies dos paralelepípedos envolventes de forma a respeitar o recobrimento regulamentar. Os varões que compõem cada armadura são agrupados formando objectos únicos e não entidades isoladas traduzindo assim, com uma maior aproximação, o modo de actuar em obra.

Na exibição animada do processo construtivo da parede, é permitida a observação em detalhe da pormenorização de armaduras no interior dos elementos estruturais segundo qualquer ponto de vista, possibilitando, assim, por parte do formando a avaliação da dificuldade que ocorre em obra na dobragem e montagem de varões, em especial nas zonas de sobreposição de armaduras (Figura 2).

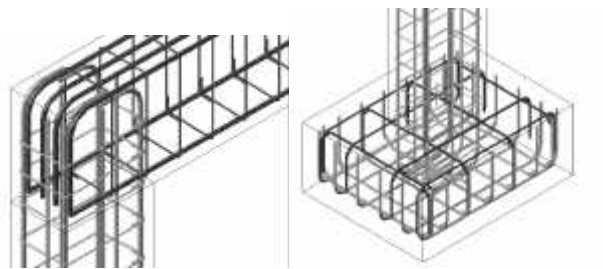


Figura 2 - Acomodação de varões nas zonas de ligação entre elementos estruturais.

O tipo de alvenaria seleccionado corresponde a uma parede exterior e é formada por duplo painel de tijolo cerâmico com uma caixa de ar. A Figura 3 inclui a representação em corte da parede modelada.

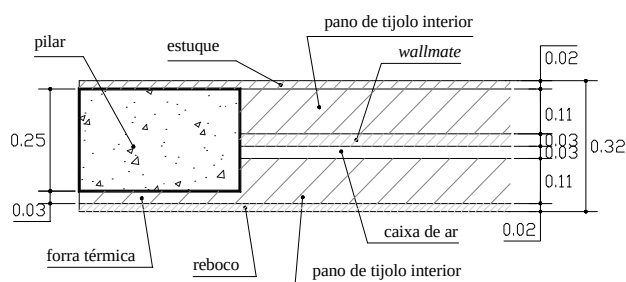


Figura 3: Representação em corte da parede dupla de alvenaria.

A Figura 4 representa o modelo tridimensional da parede composto pelos distintos painéis verticais considerados e pelos elementos estruturais envolventes (representados apenas pelas arestas).

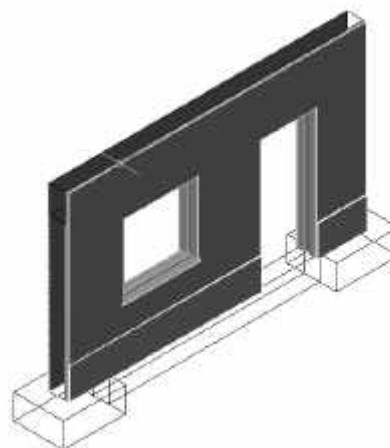


Figura 4: Representação dos painéis verticais do modelo da parede.

Cada painel foi modelado como um elemento individualizado de modo a poder ser adicionado, durante a animação virtual, ao modelo da parede simulando a sua evolução física em obra.

De forma a ilustrar uma situação frequente, admitiram-se inseridos na parede dois elementos usuais de vão, uma porta e uma janela. Na sua modelação foram definidos, como objectos individualizados as distintas peças que

compõem a cantaria, os aros, os elementos móveis e a caixa de estore.

3. MODELO MANIPULADO EM AMBIENTE VIRTUAL

A Figura 5 inclui uma projecção do modelo completo da parede. O modelo foi, de seguida, transferido para o sistema de realidade virtual EON (no formato de ficheiro de desenho com a extensão .3ds).

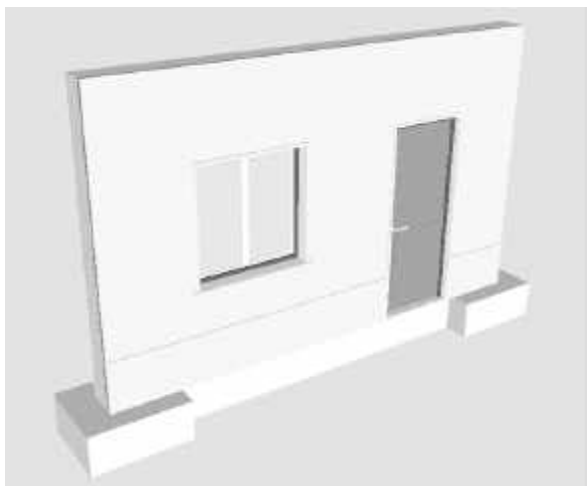


Figura 5: Representação de modelo completo da parede.

A imagem incluída na Figura 6 apresenta o ambiente de trabalho do sistema EON.

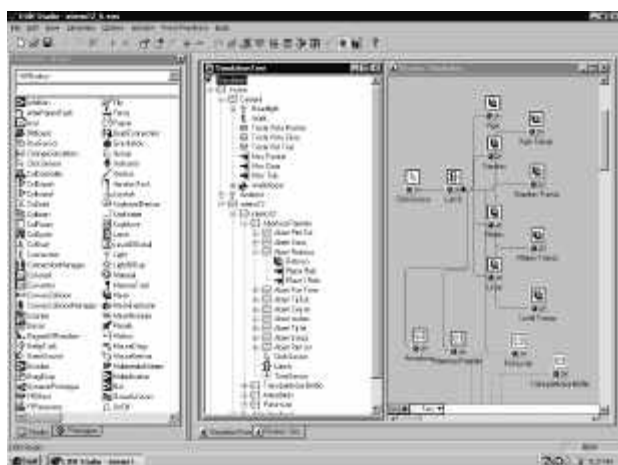


Figura 6: Ambiente de trabalho do sistema EON Studio
3.0.1.

A janela da esquerda contém um quadro de nós ou acções (de movimento, sensoriais, ...) disponíveis no sistema. A janela central é designada por árvore de simulação, pois nela são apresentados os objectos que compõem dado cenário e é o lugar onde são associadas, a cada objecto ou grupo, as acções a impor. É, pois, essencialmente através desta janela que é programada a simulação virtual pretendida. Na janela da direita são estabelecidas as ligações entre os diversos nós definindo-se uma rede. Nela são identificados os nós de origem e de destino da ligação e o modo de desencadear dada acção.

3.1 Apresentação em explosão dos painéis verticais do modelo

Uma representação animada possível de desenvolver no sistema EON, e de grande interesse académico, é a apresentação em explosão dos painéis constituintes da parede. Esta animação permite a observação da configuração de cada um dos painéis e identificar a ordem com que em obra devem ser colocados na formação da parede.

Como cada painel vertical da parede (dois panos de tijolo, forra térmica, placa isolante, reboco, estuque e duas camadas de pinturas) foi modelado como um elemento independente, é possível impor a cada um movimentos de translação individualizados. A animação foi programada de modo que cada objecto se deslocasse de uma posição inicial (todos os elementos justapostos formando a parede) para uma final (apresentação explodida) e que, posteriormente, retomasse a sua posição de serviço. O sentido da translação foi considerado normal à parede. O início da acção é comandado de um modo interactivo.

No sistema EON, a imposição de um movimento de translação a um objecto é obtida associando dois nós (ou acções) do tipo *place*, ao modelo geométrico do elemento a deslocar. Um dos nós irá permitir a caracterização do movimento desde uma posição inicial até a alterada e o outro irá traduzir o movimento de reposição. Na árvore de simulação incluída na Figura 7, pode identificar-se, por exemplo, o grupo designado por *Abert Pint Ext*, composto pelo modelo *Pintura Ex* e os deslocamentos *Place Pext* e *Place1 Pext*.

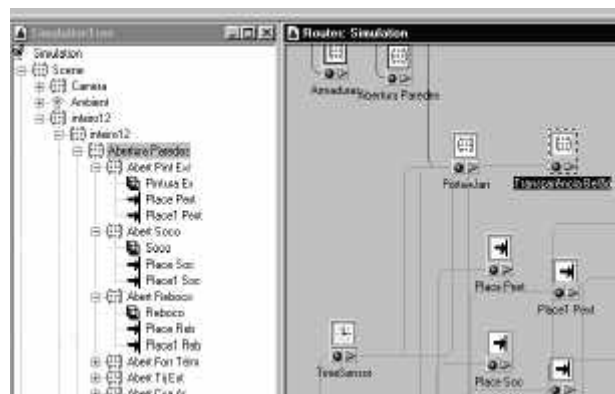


Figura 7: Detalhes do controlo da animação.

A Figura 8 apresenta as duas situações limite da animação programada, a posição inicial e em explosão.

3.2 Visualização da evolução do processo construtivo

Foi programado, no sistema EON, um outro tipo de animação de interesse didáctico, a simulação visual do processo construtivo da parede, de acordo com um determinado planeamento. Para o efeito foram consideradas 23 fases construtivas. A ordem com que os elementos vão sendo exibidos e incorporados ao modelo virtual traduz o modo de evolução física de uma parede em obra.

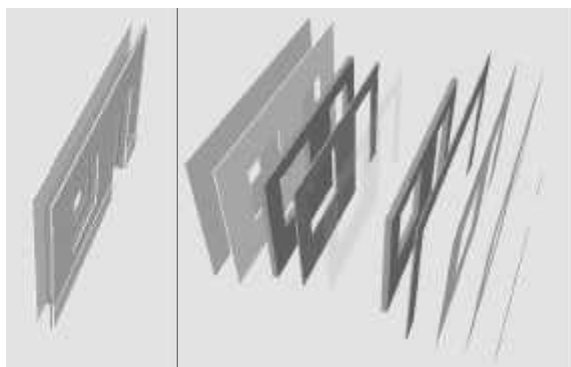


Figura 8: Posições dos painéis em serviço e em explosão.

Em simultâneo com a visualização de cada fase é apresentado um texto contendo dados relativos à etapa exibida, nomeadamente, a sua ordem na sequência construtiva, a descrição da actividade e a caracterização do material do elemento incorporado. Estes dados são colocados, junto ao canto superior direito do visor, numa posição fixa independente do movimento da câmara. Deste modo, o utilizador terá a informação sempre exibida enquanto efectua aproximações, deslocações ou rotações com a câmara sobre o modelo. A Figura 9 apresenta o modelo correspondente a uma fase construtiva, podendo observar-se, à direita e em cima, o texto informativo associado.

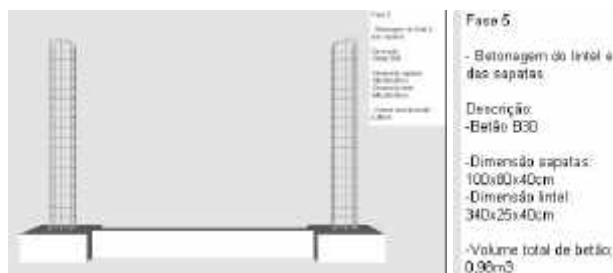


Figura 9: Representação de uma fase do modelo e do texto correspondente.

Incluiu-se sob a janela de visualização da cena virtual, uma barra de progressão da construção (Figura 10). Ao longo da animação a barra vai sendo preenchida com pequenos rectângulos simbolizando a percentagem edificada à altura da fase apresentada, em relação à completa construção da parede.

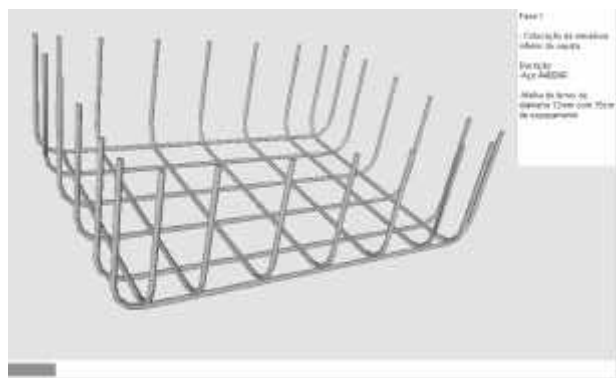


Figura 10: Visualização de fase e barra de progressão.

A animação pode ser apresentada com uma sequência lógica de construção ou com retrocesso do processo (para reapreciação de algum pormenor construtivo de algum passo anterior). Para a observação de uma fase precedente deverá ser seleccionado um rectângulo, sobre a barra de progressão. A correspondente fase é, então, visualizada.

Durante a exibição é possível destacar o elemento incorporado em cada nova fase e observá-lo em detalhe. Esta possibilidade foi introduzida na animação por ser de interesse didáctico. Possibilita a observação de pormenores de forma de cada elemento a colocar em obra e como eles se inserem na parede (Figura 11).

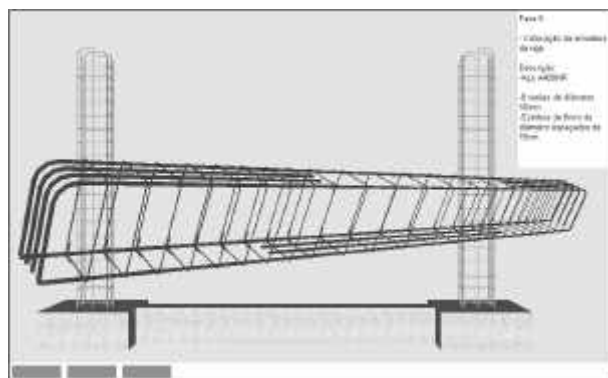


Figura 11: Visualização de elemento destacado do modelo.

Em cada fase é possível observar o modelo com a aproximação e a orientação de câmara que mais se adequa a uma correcta percepção dos detalhes dos elementos construtivos. Por exemplo, é possível observar na Figura 12 os detalhes de forma que os painéis de tijolo e a placa de isolamento devem apresentar de modo a poderem, em obra, alojar a caixa do estore. Para que o modelo possa representar uma situação realista, na execução das componentes do modelo houve a preocupação em definir a sua configuração com suficiente detalhe e rigor.

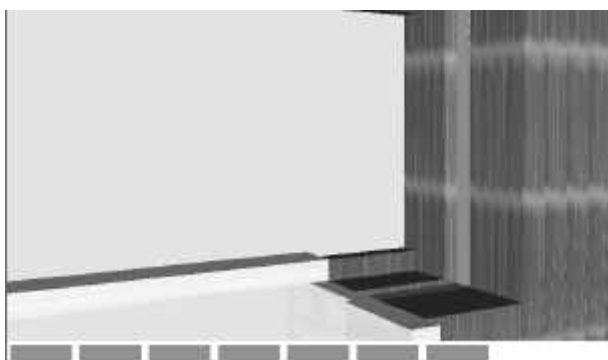


Figura 12: Detalhes da parede na zona da caixa de estore.

Adicionalmente, poderá ser exibida uma apresentação em explosão dos elementos construtivos formados por distintas peças (Figura 13).

O utilizador interage com o modelo impondo o destaque do elemento e, de seguida, a sua apresentação em explosão. As situações incluídas na figura resultam de se

ter efectuado um percurso com a câmara em torno do modelo de forma a seleccionar a perspectiva que melhor esclareça quanto aos detalhes considerados na modelação das peças.



Figura 13: Orientação de câmara sobre modelo com caixa de estore em explosão.

O modelo criado é utilizado em disciplinas afins à construção nos cursos de Engenharia Civil e Engenharia e Gestão Industrial ministrados no Instituto Superior Técnico da Universidade Técnica de Lisboa. Primeiro, a exibição é controlada pelo docente, evidenciando os aspectos que crê serem de interesse durante a exposição da matéria relacionada com os domínios focados no modelo. De seguida, o próprio aluno interage directamente, avança e retrocede na sequência construtiva, destaca elementos e observa. Não se admitiu neste fase que o modelo pudesse exibir distintas constituições da parede, sendo esse aspecto uma evolução a estudar posteriormente.

A atenção e interesse demonstrada pelos formandos incentivam os promotores a uma maior divulgação noutras cadeiras e cursos.

4. OUTRAS APLICAÇÕES

Outras componentes de edifício e distintas situações construtivas podem ser modeladas e manipuladas em cenários virtuais, para objectivos idênticos. Por exemplo, a construção de diversos tipos de cobertura ou a amostragem de tecnologias construtivas aplicadas na execução de paredes de contenção de solos.

5. CONCLUSÕES

A presente comunicação descreve uma aplicação de técnicas de realidade virtual na simulação visual da construção de uma componente de um edifício. Demonstra-se, com o exemplo apresentado, como a tecnologia de realidade virtual pode ser utilizada na elaboração de material didáctico de interesse na formação no domínio de processos construtivos.

O modelo gerado representa uma parede dupla de alvenaria. O formando pode interagir com o modelo virtual de modo a desencadear a sequência construtiva exigida em obra, retroceder nesse processo, destacar

elementos da parede e analisá-los em detalhe, observar como distintas peças de um elemento construtivo se conjugam entre si e se incorporam no modelo, conhecer a descrição de cada fase e possíveis materiais a empregar.

Por recurso a modelos virtuais, os formandos podem aprender novas técnicas construtivas, acompanhar a sequência de operações num determinado planeamento ou analisar as formas detalhadas das componentes de uma obra. Não pretendendo substituir, naturalmente, as visitas *in loco* a obras, os modelos virtuais ajudam a entender os procedimentos relacionados com a construção, que a simples observação de imagens, as quais normalmente apresentam fases finais.

A temática da construção requer usualmente uma explicação do modo de aplicar ou executar dada tecnologia, por fases. À semelhança do modelo desenvolvido também o princípio da exibição sequencial pode vantajosamente ser aplicado.

6. AGRADECIMENTOS

O trabalho desenvolvido insere-se no âmbito do projecto de investigação em curso no IST/ICIST - *Virtual reality in optimization of construction project planning* POCTI/1999/ECM/36300 [Henriques99], financiado pela Fundação para a Ciência e a Tecnologia e participado pelo fundo comunitário europeu FEDER.

7. REFERÊNCIAS

- [AutoCAD02] AutoCAD - User manual, Release 2002, Autodesk, Inc. 2002.
- [EON01] Introduction to working in EON Studio, EON Reality, Inc., 2001.
- [Henriques02] Henriques, P., Sampaio A., Visual simulation in building construction planning, 4th European Conference on Product and Process Modelling, pg. 209-212, Portoroz, Eslovénia, 2002.
- [Henriques01] Realidade virtual e o projecto de construção, Henriques, P.; Sampaio, A. e Braz, H., Congresso Nacional de Construção, pg. 357-363, Instituto Superior Técnico, Lisboa, 2001.
- [Henriques99] Henriques, P., Sampaio A., Bento, J., Braz, H., Programa do projecto de investigação: Virtual reality in optimization of construction project planning, POCTI/1999/ ECM/36300, IST/ICSIT, 1999.
- [Sampaio03] Sampaio, A., Henriques, P., Studer, P., Visualização do processo construtivo de uma parede de alvenaria modelada em ambiente virtual, Relatório ICIST, DTC nº03/03, IST, Lisboa, 2003.
- [Vince98] Vince, J., Virtual reality systems. *ACM SIGGRAPH Books series*, Addison-Wesley, ISBN 0201876876, 1998.

Sistema de Realidade Virtual para tratamento de Fobias

Vítor Cunha
ISEP

R. Dr. Ant. Bernardino de Almeida, 431
4200-072 Porto, Portugal
vmc@dee.isep.ipp.pt

Miguel Leitão

ISEP / INESC Porto

R. Dr. Ant. Bernardino de Almeida, 431
4200-072 Porto, Portugal
jml@dee.isep.ipp.pt

Sumário

Uma fobia pode ser descrita como um medo reconhecido e persistente a um estímulo circunscrito e o consequente evitar desse mesmo estímulo. Uma fobia típica, conhecida como acrofobia, ocorre quando o estímulo receado são as alturas.

O método para tratamento de fobias mais comum e com mais sucesso denomina-se exposição gradual in-vivo. Neste método, o paciente com fobia é exposto a uma sequência do estímulo receado. É conhecido que a exposição a uma correcta intensidade do estímulo conduz a uma habituação e o consequente decréscimo do receio, no entanto, a aplicação deste método pode apresentar dificuldades. Uma alternativa prometedora é a exposição gradual a um estímulo virtual, usando ambientes de Realidade Virtual. A exposição à Realidade Virtual apresenta várias vantagens sobre a exposição in-vivo. O tratamento pode ser conduzido de uma forma privada, no consultório do terapeuta, em vez de ser realizado no exterior, onde as situações reais de fobia se encontram. Assim o tratamento pode ser melhor assistido e realizado de uma forma mais eficiente do ponto de vista económico. Uma outra vantagem é a facilidade de controlar o estímulo virtual, permitindo assim ao terapeuta dosear a intensidade do medo.

Este artigo descreve o sistema de Realidade Virtual (VRFobias) desenvolvido no ISEP e testado na Universidade do Minho. Os requisitos principais de um sistema de Realidade Virtual, necessário no tratamento de fobias, estão descritos. Os resultados obtidos até ao momento e os métodos usados no controlo da intensidade do estímulo também estão descritos neste artigo.

Expressões-chave

Fobia, Terapia de exposição, Tratamento, Realidade Virtual, Ambiente virtual.

1. INTRODUÇÃO

A Realidade Virtual (RV) é um poderoso paradigma da interacção que usa representações geradas por computador para criar no utilizador uma sensação de presença, para que se sintam num ambiente virtual. O tipo de *software* usado nestas representações permite que os ambientes virtuais sejam precisamente manipulados de uma forma que não é possível no mundo real.

Para se conseguir uma interacção com a RV é necessário o uso de hardware dedicado, como por exemplo um *Head Mounted Display* (HMD) e outros dispositivos como luvas e 3D *mouse*. Usualmente o HMD integra um sensor de posição (*tracker*) que continuamente reporta a posição da cabeça aos geradores sensoriais. A sensação visual é obtida através da visualização de imagens em tempo real geradas pelo processo de *rendering*. Este processo de *rendering* obtém a informação de estado da base de dados, especialmente a informação da câmara e outros objectos, gerando imagens consonantes com esta informação. Normalmente este processo depende de *hardware*

hardware dedicado e bibliotecas de *software* especializado para produzir imagens de qualidade razoável no intervalo de tempo disponível. Várias técnicas avançadas e bem conhecidas são usadas para aumentar a velocidade de geração de imagens. Estas técnicas incluem a organização hierárquica dos objectos, distância de *clipping*, *view frustum culling*, e selecção do *level-of-detail*. Devido ao papel dominante da visão no universo humano, este processo tem um importância determinante na sensação de presença num ambiente virtual [Slater94].

1.1 Terapia de Exposição

O tratamento mais comum e bem sucedido no tratamento de fobias é a exposição gradual in-vivo. A terapia de exposição é baseada no princípio de que as pessoas se habituem a situações de desconforto. Neste método, o paciente com fobia é exposto a uma sequência dos estímulos de que tem medo. É conhecido que a intensidade correcta do estímulo leva à habituação e ao decréscimo

decréscimo do medo [Emmelkamp85]. Este processo de habituação ocorre naturalmente para 95% das pessoas.

A exposição consiste em encarar situações recheadas de uma forma gradual e progressiva. É importante continuar a encarar essas situações até que o desconforto diminua. Usualmente o terapeuta elabora uma sequência de situações recheadas de forma individualizada tendo o paciente e o terapeuta que ir aos locais onde essas situações ocorrem.

Este método pode apresentar dificuldades importantes. O estímulo recheado pode ser difícil de obter (os que envolvam animais), dispendioso (aviões). Para além destes factores, o estímulo pode ser de difícil controlo o que não permite o ajuste preciso da intensidade deste. Em situações onde o estímulo recheado só pode ser apresentado fora do consultório, pode-se tornar bastante difícil monitorizar as necessárias variáveis psico-fisiológicas como o CRT ou batida cardíaca, por forma a ajudar o terapeuta a compreender o nível de medo e assim controlar o processo de exposição.

A exposição pode ser feita ao vivo (na vida real) ou em imaginação. Ao vivo, a exposição é usualmente mais eficiente do que o método imaginativo, mas a exposição imaginária é por vezes sugerida como uma alternativa, especialmente quando o estímulo recheado é difícil de obter. Isto pode ser obtido pelo uso de fotografias, imagens, registos áudio, livros ou qualquer objecto que recrie o estímulo recheado.

1.2 Exposição à RV

A exposição à RV pode ser considerada uma nova forma de terapia de exposição imaginária, onde o ambiente imaginado é recriado a partir do mundo real utilizando um sistema de Realidade Virtual.

Numa terapia de exposição à RV, o paciente é confrontado com ambientes virtuais que provocam um medo indesejado. Os estímulos recheados existentes num ambiente virtual são mais facilmente controlados do que os existentes no mundo real. Desta forma, o terapeuta pode variar precisamente a intensidade do medo. A exposição à RV possui mais algumas vantagens relativamente à

relativamente à exposição in-vivo. O tratamento pode ser conduzido, de forma privada, no consultório do terapeuta em vez do mundo exterior, onde as situações de real fobia se encontram. O tratamento pode assim ser melhor assistido e com custos mais reduzidos.

Várias pesquisas, recentemente realizadas, estudaram a funcionalidade desta alternativa. Estas pesquisas abordam diferentes fobias como o medo de aranhas [Carlin97], o medo de alturas [Rothbaum95], claustrofobia [Alcaniz98] e o medo de andar de avião [Banos02].

A maioria destes trabalhos normalmente tenta reproduzir o método normal de tratamento, mas utilizando exposição virtual num ambiente simulado, pelo que não exploram na totalidade o potencial e a versatilidade do ambiente de Realidade Virtual. Esperamos que melhores resultados poderão ser obtidos à medida que novos métodos para melhorar a qualidade dos estímulos recheados sejam definidos.

2. IMPLEMENTAÇÃO

O sistema de RV desenvolvido para testar a terapia de exposição virtual (VRFobias) é baseado numa estação de trabalho SGI Octane MXE que guarda a base de dados do ambiente com descrições de todos os objectos existente no mundo virtual. Esta estação de trabalho também é responsável pelo processo de síntese de imagem que gera as imagens apresentadas ao paciente com fobia.

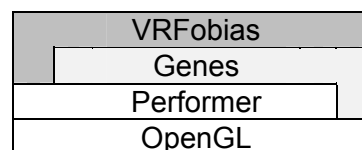


Figura 1: Desenvolvimento do sistema.

VRFobias foi desenvolvido usando o Genes, como se pode observar na figura 1. Genes (*Generic Environment Simulator*) é um simulador de ambientes genérico desenvolvido na instituição. Este tem como camada inferior o *Performer*, um conhecido software de simulação visual da *Silicon Graphics* [SiliconGraphics03].

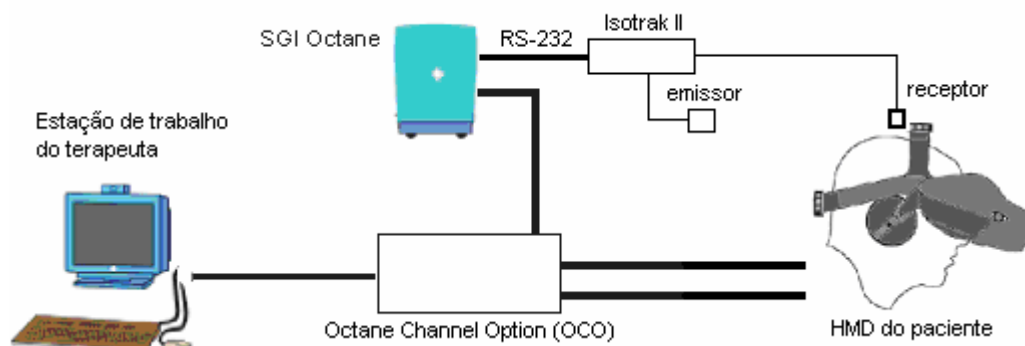


Figura 2: Geração de imagem em vários canais e sistema de tracking.

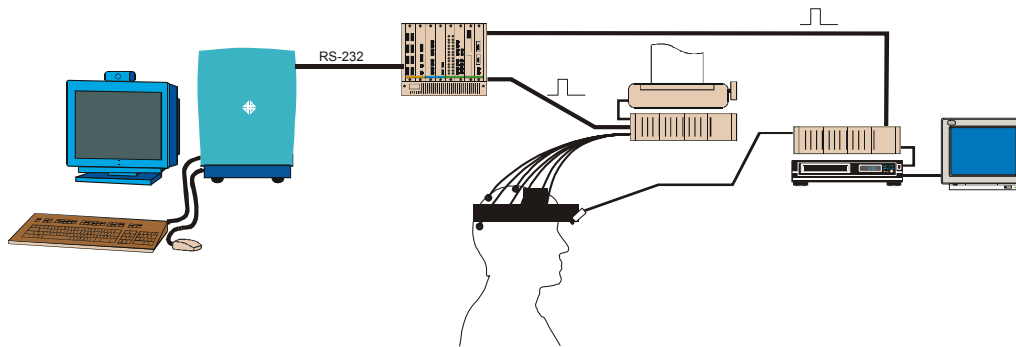


Figura 3: Sincronização com os monitores psico-fisiológicos.

Genes permite a geração de imagens e som em tempo real, em conjunto com alguns controladores de movimento e comportamento. Genes inclui uma poderosa linguagem de descrição de modelos e cenas que integra a geometria dos objectos, luzes e propriedades do som, e características físicas. Genes é também usado em outras aplicações de simulação visual como um realista simulador de condução, Dris [Leitão97].

O sistema necessita de um OCO (*Octane Channel Option*) dedicado a esta função, como ilustrado na figura 2, que permite dividir o *frame buffer* da estação de trabalho em 4 canais independentes.

Com este dispositivo, são gerados simultaneamente três canais de vídeo independentes. Dois destes três canais de vídeo são usados para gerar uma imagem estereoscópica apresentada ao paciente através de um *Head Mounted Display* Virtual Research V6. Respectivamente um para o olho esquerdo e outro para o olho direito. As imagens são obtidas através da utilização de duas câmaras virtuais desfasadas de uma distância inter-ocular regulável. Esta regulação é necessária para a minimização do desconforto do paciente e consequente melhoria da sensação 3D. Devido às limitações do dispositivo de visualização a resolução adoptada foi de 640x480 pixels a 60 Hz por canal [VirtualResearch03].

O terceiro canal está disponível para o terapeuta, que pode escolher interactivamente o seu ponto de vista preferencial. O terapeuta também tem acesso, em tempo real, à informação relativa ao estado do ambiente virtual. Esta informação é guardada na estação de trabalho possibilitando ao terapeuta realizar uma análise posterior.

A orientação das câmaras virtuais, que representa a orientação actual da cabeça do paciente, é actualizada a partir da informação obtida do sensor de posição (*tracker*) Isotrak II, ilustrado na figura 2. Este dispositivo baseia o seu funcionamento na medição contínua da posição/orientação de um pequeno receptor acoplado ao HMD do paciente [Polhemus03]. Estes dados são obtidos por um processo paralelo à aplicação VRFobias.

Nas experiências realizadas para tratamento de acrofobia, o terapeuta tinha controlo interactivo da altura virtual do paciente através de um tradicional teclado. Desta forma as câmaras virtuais usadas para gerar as imagens para o paciente são controlada em 4 graus de liberdade. Um grau de liberdade corresponde à altura da plataforma controlada pelo terapeuta, enquanto que os restantes três referem-se aos ângulos de orientação *heading*, *pitch* e *roll* (HPR), obtidos a partir da informação recolhida do *tracker*.

Todo o sistema é sincronizado com vários equipamentos de monitorização psico-fisiológica através de um módulo de sincronização dedicado, como mostrado na figura 3. Este módulo recebe comandos da estação de trabalho sendo também utilizado em outras aplicações de RV [Leitão99].

O ambiente acústico é simulado, de forma não interactiva, através de uma gravação anteriormente efectuada dos sons do mundo real.

Para que RV seja efectiva no tratamento é necessário assegurar parâmetros como os atrasos associados à leitura de dispositivos periféricos, número de *frames* por segundo (25), qualidade geral dos modelos e texturas e minimização do desconforto provocado pela utilização de um HMD.

É sabido que a terapia de exposição pode ser eficiente com qualquer ambiente ou situação que reproduza um estímulo do mesmo tipo do estímulo receado. Normalmente não é necessário usar exactamente o mesmo estímulo receado. Porém, foi decidido implementar um modelo computacional que reproduzisse um cenário do mundo real existente num arranha-céus local. O modelo computacional foi desenvolvido utilizando texturas obtidas de fotografias do cenário real. A figura 4 permite comparar a vista do mundo real com a do mundo virtual.

Esta relação entre o ambiente virtual e real permitiu a validação do sistema comparando a terapia de exposição in-vivo com a terapia de exposição à RV.



Figura 4: Vista do mundo real (esquerda) e vista do ambiente virtual (direita).

A figura 5 mostra o uso típico do sistema durante uma sessão de terapia de exposição à RV de um paciente com acrofobia.



Figura 5: Terapia de exposição à RV para acrofóbicos.

3. CONCLUSÕES

Várias pesquisas, recentemente realizadas, estudaram a funcionalidade desta alternativa e alguns trabalhos são conhecidos que tentam comparar a exposição à Realidade Virtual com a exposição in-vivo. Estes trabalhos usualmente tentam manter o método normal de tratamento, mas com a exposição virtual num ambiente simulado, não usufruindo de todo o potencial e versatilidade possível nos ambientes virtuais.

Vários tratamentos experimentais, já realizados, permitiram a validação do sistema como uma ferramenta eficiente no tratamento de fobias [Emmelkamp01].

Algumas técnicas inovadoras, orientadas ao controlo da intensidade do medo, estão actualmente a ser implementadas e testadas. Utilizando a informação recolhida dos monitores psico-fisiológicos, esperamos controlar automaticamente a altura virtual do paciente com fobia. Esperamos que melhores resultados sejam alcançados à medida que este novo método de melhorar a qualidade e controlo do estímulo simulado seja desenvolvido e testado. Resultados deste novo método serão brevemente alcançados. Este controlo automático pode ser considerado como um passo na direcção de conseguir um tratamento de fobias mais automático, facilitando a tarefa do terapeuta.

4. REFERÊNCIAS

- [Alcaniz98] M. Alcaniz, R. M. Banos, C. Botella, C. Perpina, A. Rey, & H. Villa (1998). Virtual Reality Treatment of Claustrophobia. *Behaviour Research & Therapy*, 36(2), 239-246.
- [Banos02] R. M. Banos, C. Botella, C. Perpina, M. Alcaniz, J. A. Lozano, J. Osma, & M. Gallardo (2002). Virtual reality treatment of flying phobia. *IEEE Transactions on Information Technology in Biomedicine* 6(3):206-212
- [VirtualResearch03] Virtual Research Systems. V6 Head Mount Display. 18/07/2003.
<<http://www.virtualresearch.com/products/v6.htm>>
- [Polhemus03] Polhemus. The fast and easy digital tracker. 18/07/2003.
<<http://www.polhemus.com/isotrak.htm>>
- [Carlin97] A. S. Carlin, H. G. Hoffman & S. Weghorst (1997). Virtual reality and tactile augmentation in the treatment of spider phobia: A case study. *Behaviour Research and Therapy*, 35, 153-158.
- [Emmelkamp85] P. M. G. Emmelkamp & M. Felten (1985). The process of exposure in vivo: cognitive and physiological changes during treatment of acrophobia. *Behaviour Research & Therapy*. 23, 219-223.
- [Emmelkamp01] P.M.G. Emmelkamp, M. Bruynzeel, L. Drost & C.A.P.G. van der Mast (2001). Virtual Reality Treatment in Acrophobia: A Comparison with Exposure in Vivo. *Cyberpsychology and Behavior*, 4(3), 335-341.
- [Leitão97] J. M. Leitão, A. Coelho & F. N. Ferreira (1997). DriS – A Virtual Driving Simulator. *Proceedings of the Second International Seminar on Human Factors in Road Traffic*, ISBN 972-8098-25-1
- [Leitão99] J. M. Leitão, A. Moreira, J. A. Santos, A. A. Sousa & F. N. Ferreira (1999). Evaluation of Driving Education Methods in a Driving Simulator. *Proceedings of the Graphics And Visualization Education Workshop 1999*. 156-157.
<<http://www.siggraph.org/education/conferences/GVE99/papers/GVE99.Proceedings.pdf>>
- [Rothbaum95] B. O. Rothbaum, L. F. Hodges, R. Kooper, D. Opdyke, J. Williford & M. M. North (1995). Virtual reality graded exposure in the treatment of acrophobia: a case report. *Behavior Therapy*, 26(3), 547-554.

[Slater94] M. Slater, M. Usah, A. Steed (1994). Depth of presence in virtual environments. *Presence: Teleoperators and Virtual Environments*, 3, 130-144.

[SiliconGraphics03] Silicon Graphics. OpenGL Performer. 08/05/2003.
<<http://www.sgi.com/software/performer/>>

Simulação e avaliação de um projecto urbano através dos computadores

Pedro Leão Neto
Faculdade de Arquitectura U P
pleao@arq.up.pt

Abstract

State-of-the-art computer technology has improved the integration of different types of representations and data and has provided new possibilities for manipulating results. Consequently, the use of computers has been widespread in urban design and the planning process. The potential that computer environments brought for describing future developments to the public at large is enormous. Nevertheless, when imagery and realism dominate the communication process, they may obscure important issues that relate more to the underlying logic of urban fabric.

A sample of 23 university students from nonarchitecture courses and a sample of 23 students from architecture (contrasted group design) were used for analysing differences in their evaluation on form and content of a video presentation concerning an urban design project in Lisbon. The results highlighted the power that imagery and realistic computer models have, when used in an unbalanced way, for overshadowing differences between subjects.

Keywords

Urban design; communication; computer technology; imagery and realism

Sinopse

O potencial que a tecnologia informática possui para descrever empreendimentos futuros ao público em geral é enorme. No entanto, a utilização de modelos realistas de computador podem, por si só, ofuscar variáveis importantes do projecto urbano que se prendem mais com a sua lógica de configuração espacial e com as ideias que lhe estão subjacentes.

Foram utilizados dois grupos de contraste, compostos por estudantes universitários de cursos diferentes, para analisar as diferenças em relação à avaliação feita por eles sobre a forma e o conteúdo de uma apresentação em vídeo de um projecto urbano em Lisboa. Os resultados obtidos realçam o poder que a tecnologia informática e a modelação realista possuem para esbater as diferenças entre os sujeitos.

Palavras chave

Projecto urbano; comunicação; tecnologia informática; modelação realista.

1. INTRODUÇÃO

Os métodos de planeamento e de projecto urbano têm evoluído continuamente ao longo dos tempos. As mudanças são, de certa forma, uma resposta às necessidades políticas e económicas da sociedade, reflectindo a influência da tecnologia e da cultura. Desta forma, os computadores estão a influenciar tanto a forma de fazer as cidades como a comunicação entre todos os actores do processo de planeamento.

Alguns autores, tais como Michael Batty (Batty 1995; Batty e Doyle 1998) acreditam nas potencialidades de comunicação ligadas ao estado da arte em tecnologia informática. Tem havido bastante investigação sobre a forma como a modelação e representação computadorizada contribui para realçar de forma mais eficaz a comunicação de projectos urbanos e de arquitectura (Dodge, Doyle 1998; Emmott e Travis 1995; Hall 1998). Existe

também uma grande confiança na Realidade Virtual (RV) para simular adequadamente a forma dos edifícios e para analisar a utilização e interacção das pessoas nesses ambientes virtuais. No entanto, tal como foi mencionado por estes autores, há o problema óbvio de se saber até que ponto o comportamento no ambiente do mundo virtual é passível de ser transferido para o mundo real (Batty, Conroy 1998, p1-2). Alguns autores demonstram grande esperança nas capacidades de visualização em computador. Por exemplo, Tony Hall (1998, p1-2) acredita que a visualização computadorizada é particularmente adequada para comunicar um empreendimento futuro ao público em geral.

Outros autores, pelo contrário, são mais cépticos em relação à tecnologia informática aplicada à criação de simulações realistas para a comunicação de projecto urbano. Eles revelam como o poder das imagens digitais, especialmente das representações muito realistas,

têm sido usadas ou podem ser usadas de forma a que ofusquem todas as outras variáveis que não estão directamente relacionadas com realismo ou percepção espacial (Bosselmann 1997; Koutmanis 1997; McNamara 1998; Vassils 1997). O trabalho destes autores aponta, para além de outros aspectos, que o esbatimento da fronteira entre representação e realidade é algo que está a acontecer e que isso deve ser tornado mais explícito.

Dentro do processo de planeamento e de comunicação do projecto urbano, se pretendemos alcançar um público mais informado e mais consciente, temos que desviar a tendência actual que dá um peso excessivo à utilização de métodos de representação realistas, de forma a que outras questões mais relacionadas com a lógica subjacente ao projecto urbano possam ser comunicadas de maneira eficaz.

De importância primordial para a discussão acima descrita são as questões relacionadas com a correlação *in situ* [que estuda o grau de correlação obtido por diversas representações de edifícios e de espaços urbanos quando comparadas com a experiência real desses objectos arquitectónicos e espaços urbanos]. Na verdade, os resultados que demonstram as preferências das pessoas em relação a vários media indicam também que os métodos combinados podem ser contra-produtivos para a correlação *in situ* (Stamps III 1998; Stamps III 1993b; Nasar 1984; Stamps III 1993a; Stamps III 1993c; Stamps III 1997).

Por outro lado, em resultado de outras investigações, há argumentos que suportam a utilização de modelos combinados e/ou integrados para a comunicação de um projecto urbano (Decker 1994).

Finalmente, a questão das diferenças de familiaridade para com diferentes métodos de representação e em relação a questões de urbanismo e de arquitectura devem ser, de alguma forma, consideradas, quando se pretende estudar a comunicação eficaz de um projecto urbano ao público em geral. Isto porque para se obter um ambiente que ajude ao pleno desenvolvimento e à qualidade do projecto urbano, a comunicação entre todos os actores do processo de planeamento - Técnicos, Políticos, Agentes culturais e Público em geral - é um elemento essencial (Hall 1996; Punter and Carmona 1997; Lang 1994).

2. APRESENTAÇÃO DO VÍDEO E GRUPOS DE CONTRASTE

Depois de ter avaliado um conjunto de possíveis apresentações em vídeo que poderiam ser utilizadas para o presente estudo, foi seleccionado o vídeo da Avenida. O cliente institucional desse vídeo havia sido o Município de Lisboa, o qual tinha como intenção utilizá-lo como suporte de comunicação do projecto da Avenida ao público em geral. Os autores do projecto eram Manuel Fernandes de Sá, LDA e a equipa responsável pela produção e concepção do vídeo era constituída por uma equipa interdisciplinar de arquitectos, urbanistas, engenheiros informáticos e outros especialistas ligados à criação de imagem. A instituição responsável pela produção

do vídeo era o Instituto de Engenharia de Sistemas de Computador (INESC).

A apresentação em vídeo fazia uso de diferentes métodos de representação para comunicar o lugar, tanto antes, como depois do projecto urbano. Música de fundo, sons da cidade ao vivo e texto foram combinados com imagens para construir um vídeo de 13 minutos e foram obtidos diferentes níveis de realismo através da utilização de fotografia, fotomontagem, filme a cores, um modelo a 3D com animação e plantas. Foram aplicadas técnicas diferentes de visualização de imagens, especialmente para decompor o modelo da infra-estrutura submersa em vários dos seus componentes e para integrar o modelo com filme e fotografia.

Os grupos de comparação deste estudo eram compostos por estudantes universitários da Universidade Pública do Porto. Uma amostra de 23 estudantes de arquitectura (4º ano) e uma amostra de 23 estudantes de outros cursos (13 alunos de Geografia e 10 estudantes de Sociologia, todos do 3º ano) foram utilizadas para representar a população de estudantes dos últimos anos dos três cursos acima referidos.

3. MÉTODO

Para conseguir obter a participação dos estudantes o autor contactou os professores dos últimos anos de cada curso universitário para obter a sua permissão para mostrar o vídeo aos estudantes e para lhes ser entregue um questionário. Foi pedido aos sujeitos que preenchessem um questionário sobre a forma e o conteúdo da apresentação em vídeo, que todos viram imediatamente antes de começar a responder às questões. Antes de ver a apresentação em vídeo, foi pedido aos estudantes que lessem o questionário de forma a que quaisquer dúvidas sobre o significado das perguntas pudessem ser esclarecidas de antemão. Depois de ver a apresentação em vídeo e completar o questionário, teve lugar uma discussão aberta.

Para os objectivos deste estudo, tendo em consideração o tipo de dados disponível (ordinais, nominais, não-paramétricos) e o formato das perguntas, o autor utilizou análises de frequências de distribuição, de tabelas de cruzamento de variáveis e testes não-paramétricos. A Técnica de Diferencial Semântica de Osgood foi seleccionada para uma fracção das perguntas que se centravam em diferentes aspectos da apresentação em vídeo e no conhecimento e opiniões das pessoas antes e depois do projecto urbano. Foi obtida uma medição da atitude em relação a cada objecto de 1 a 5, atribuindo-se a classificação de 5 a uma atitude positiva e de 1 a uma atitude negativa. Por exemplo, na pergunta EP1 - O espaço público que é filmado é: - são dadas as 5 opções seguintes: 1- Extremamente aberto, 2- Aberto, 3- Nem aberto nem fechado, 4- Fechado, 5- Extremamente fechado. Foi seleccionado um outro tipo de escala ordinal para quase toda a restante fracção das perguntas. Esta escala ordinal foi aplicada para avaliar o julgamento das pessoas sobre diferentes métodos de representação, diferentes técnicas de visualização de imagens e pares diferentes de escalas adjectivas

opostas. Para cada pergunta, o sujeito tinha de classificar em importância descendente apenas três elementos do conjunto que lhe era fornecido. O número de elementos em cada variável (3, 5, 7 ou 9) para grupos diferentes de perguntas. Por exemplo, na pergunta MV1 - Na sua opinião o que é mais eficaz para a compreensão de uma proposta de projecto urbano? - Uma imagem de cada vez no ecrã; Imagens lado a lado no ecrã; Imagens sobrepostas e integradas no ecrã. Isto significava que a escala era mais precisa para as perguntas que tinham menos elementos para classificar, porque essas tinham um número inferior de elementos com a classificação de zero. O nível nominal foi seleccionado para a restante fracção das perguntas. Estas perguntas tinham como objectivo classificar os que respondiam em dois grupos, em relação a (a) conhecimento prévio do local antes de ver a primeira parte do vídeo e (b) mudança de opinião sobre o local depois de ver a segunda parte do vídeo. Assim, cada uma dessas perguntas implicava uma resposta Sim ou Não.

Os resultados de todas as perguntas foram introduzidos e analisados no SPSS 9.0 para Windows.

4. RESULTADOS

4.1 Diferenças entre os sujeitos

Os resultados mostraram que apesar de todas as diferenças entre os dois grupos de sujeitos - estudantes de arquitectura e estudantes de outros cursos - em relação à compreensão e avaliação da forma e do conteúdo da apresentação em vídeo, a opinião final era a mesma e positiva (Modo e Valor Mediano de 4).

De facto, foram encontradas diferenças significativas em relação a vários aspectos. Por exemplo, os estudantes de Arquitectura atribuíram uma classificação superior a dimensões espaciais abstractas, como configuração do espaço (sig. 0,014: Mann-Whitney) ou dimensões mais formais como relações volumétricas entre os edifícios (sig. 0,023: Mann-Whitney), enquanto que os estudantes dos outros cursos atribuíram uma classificação superior a variáveis espaciais menos formais, como a posição relativa das pessoas em relação aos edifícios (valor baixo de sig. 0,087: Mann-Whitney).

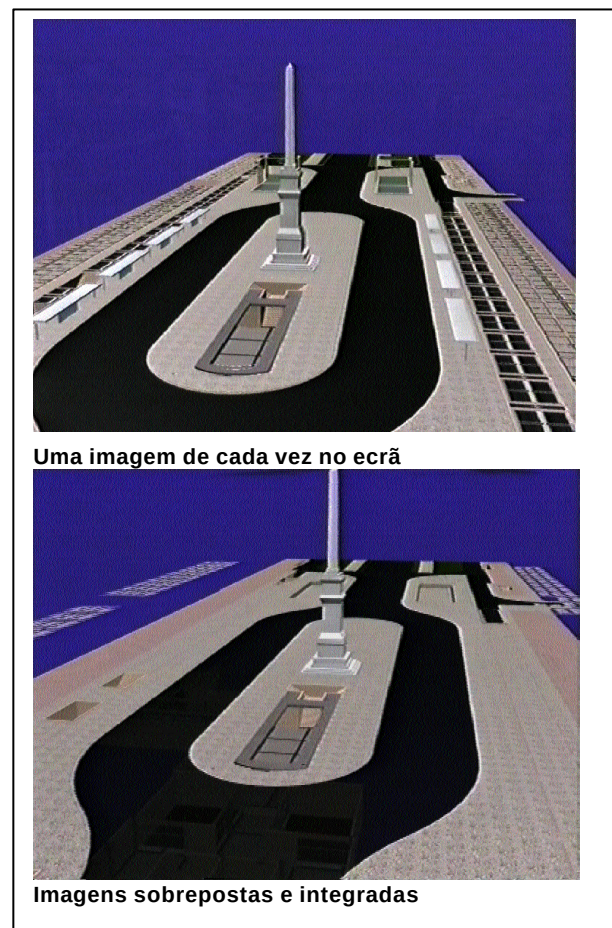
Os resultados indicam também que não há diferenças entre os grupos quanto à classificação do modelo a 3D com animação em primeiro lugar, e fotomontagem em segundo lugar como os mais importantes métodos de representação para comunicar um projecto urbano.

4.2 O modelo a 3D como o método de representação com maior influência

Os resultados indicam que o modelo a 3D com animação é o método de representação com maior influência, quando comparado com todos os outros. Significando isto que o modelo a 3D com animação é um método de representação pluralista e que, geralmente, todos os estudantes lhe atribuíram classificações elevadas, tanto para questões realistas como abstractas.

4.3 Imagens sobrepostas integradas no ecrã

Os resultados indicam que as imagens sobrepostas integradas no ecrã obtiveram a classificação mais elevada em ambos os grupos em comparação com (a) uma imagem de cada vez no ecrã e (b) imagens lado a lado no ecrã (figuras 4.3).



Figuras 4.3

4.4 As perguntas qualitativas e a discussão aberta no final

Os resultados da pergunta qualitativa GO1 (qualquer comentário que o aluno achasse de utilidade para o questionário) indicam que os estudantes de arquitectura são mais críticos sobre o tipo e conteúdo dos métodos de representação utilizados (falta de métodos de representação a 2D, modelo a 3D com detalhe insuficiente), ligando estas críticas às questões de projecto. Os estudantes que não são de Arquitectura também apontaram a falta de detalhe, mas são menos críticos em relação aos métodos de representação e têm tendência para ir directamente para as questões de projecto (segurança, conforto e qualidade de vida na cidade). Isto mostra uma diferença entre os dois grupos em relação aos aspectos que julgam ser mais importantes quando avaliam uma apresentação em vídeo de um projecto urbano.

Uma das questões principais da discussão aberta foi a de considerar, por alguns estudantes, que a apresentação em vídeo era muito manipulativa. Outros afirmaram que a

apresentação em vídeo se baseava demasiado em "imagem", não fornecendo informações suficientes sobre o projecto urbano (conteúdo), enquanto que outros ainda afirmaram que a primeira parte do vídeo foi demasiado longa e que a animação deveria ter tido mais detalhe.

5. DISCUSSÕES E CONCLUSÕES

O autor pensa que o denominador comum que explica o resultado homogéneo para ambos os grupos em relação à avaliação positiva que fizeram do projecto urbano foi a forma da apresentação em vídeo e não o seu conteúdo - o projecto urbano. De facto, houve muitas diferenças entre os dois grupos em relação a um vasto conjunto de variáveis. No entanto, parece que um outro factor ou factores existiram para que a opinião final dos estudantes fosse a mesma. Esse denominador comum, na opinião do autor, foi a forma da apresentação em vídeo e a falta de informação adequada sobre o projecto e o lugar de intervenção (conteúdo). Na verdade, quase todos concordaram que faltavam informações importantes que permitissem uma avaliação correcta do projecto urbano (segurança, estacionamento de veículos, configuração de espaços verdes/construídos/abertos e outras mais).

Estas verificações acentuam também a grande influência que um modelo a 3D com animação pode ter sobre a avaliação e compreensão de um projecto urbano e o perigo de acreditar que o modelo a 3D com animação pode ser um método de representação auto-suficiente, tendo em conta que ele obteve classificações elevadas tanto para questões de projecto 'realistas' como abstractas. Na verdade, o modelo a 3D com animação "compete" com sucesso com a realidade, mesmo quando a ligação entre os dois é ténue. No presente estudo, isso foi claro quando, apesar da artificialidade reconhecida do modelo, os sujeitos atribuíram ao modelo a 3D com animação uma classificação mais elevada do que a fotomontagem para ter uma percepção do lugar como se lá estivesse. Este resultado vem também dar força ao que alguns autores já anteriormente tinham mencionado (Steuer 1992): que outros sentidos humanos que não a visão podem ser utilizados com sucesso para dar aos utilizadores uma experiência convincente do espaço através de um ambiente mediado. O problema com este "encantamento" tecnológico é que não podemos transferi-lo directamente para o real, no entanto a sua influência pode ser determinante. Assim, o modelo a 3D parece ser o método mais influente e preferido, sobretudo por causa das suas capacidades poderosas de apresentação e da sua capacidade para fazer uso activo do sistema háptico [relacionado com a manipulação e a percepção dos objectos através do sentido do tacto e dos estímulos que são produzidos e percebidos no organismo, em especial aqueles que estão relacionados com a posição e o movimento do corpo] dos espectadores, mesmo quando estas propriedades têm uma relação ténue com a realidade.

O autor conclui que o impacto dos media (apresentação em vídeo) foi de tal forma poderoso que levou os estudantes a exprimir uma opinião favorável em relação ao

lugar, depois da proposta de projecto urbano. No entanto, esta opinião não corresponde totalmente a uma avaliação positiva sobre o conteúdo da apresentação - o projecto urbano. As contradições entre a elevada classificação obtida pela avaliação do projecto urbano e as muitas críticas feitas em relação à apresentação em vídeo indicam que os estudantes estavam de certa forma "presos" a uma matriz que descrevia e representava o projecto de uma forma ineficaz e parcial.

Para finalizar, o autor acredita que estes resultados reforcem o que vários autores já referiram. O facto de que a imagética e os modelos digitais, particularmente representações muito realistas, poderem por si ter um peso excessivo e não ajudar a tornar claras questões importantes de projecto, influenciando de forma decisiva a avaliação das pessoas sobre uma proposta de intervenção urbana (Bosselmann 1997; Koutmanis 1997; McNamara 1998; Vassils 1997; Decker 1994). Isto significa que a arquitectura e o projecto urbano são de facto realidades demasiado ricas e complexas para serem compreendidas ou avaliadas com base apenas numa apresentação em vídeo de treze minutos, deste género. O perigo de pensar assim é maior ainda quando se acredita ou se realça em demasia a imagem por si só e quando os modelos de representação pretendem substituir a realidade, tal como indicam os resultados do presente estudo. Em primeiro lugar, uma apresentação em vídeo tem que ser um exercício para promover a discussão e a informação. Em segundo lugar, a utilização de diferentes métodos de representação deverá ser equilibrada.

6. BIBLIOGRAFIA

- Batty, M. (1995) The Computable City Fourth International Conference on Computers in Urban Planning and Urban Management, Melbourne, pp 1-25. <http://www.geog.buffalo.edu/Geo666/batty/melbourn.html>
- Batty, Michael, Conroy, Ruth, Hillier, Bill, Jiang, Bin, Desyllas, Jake, Mottram, Chiron, Penn, Alan, Smith, Andy, Turner, Alasdair (1998) The Virtual Tate CASA - Working Paper Series.
- Batty, Michael, Doyle, Simon (1998) Virtual Regeneration CASA - Working paper series.
- Bosselmann, P. (1997) Representation of Places: Reality and Realism in City Design (University of California Press, London), pp 14-204.
- Decker, John (1994) The validation of computer simulations for design guideline dispute resolution Environment and Behaviour 26 N° 3, pp 421-443.
- Dodge, M., Doyle, S., Smith, A., Fleetwood, S. (1998) Towards the Virtual City: VR & Internet GIS for Urban Planning Virtual Reality and Geographical Information Systems.
- Emmott, S. J., Travis, D. S. (1995). The electronic Agora. In Emmott, S. J., Travis, D. S. (eds.) Information Superhighway: Multimedia Users and Futures (Academic, London).

- Hall, A. C. (1996) *Design Control: towards a new approach* (Butterworth Architecture, Oxford)
- Hall, A. C. (1998). The use of computer visualisation in planning control. In *Exploiting it in planning*.
- Koutmanis, A. (1997) Ongoing research project on architectural representation. University of Technology, (Delft).
<http://130.161.162.100/koutmanis/research/representation>.
- Lang, J. (1994) *Urban Design: The American Experience* (Van Nostrand Reinhold, USA)
- McNamara, K. R. (1998). Review on Peter Bosselman. The Representation of Places, Urban Mail Archives.
<http://www.unimelb.edu.au/infoserv/urban/hma/hurban/current/0788.html>.
- Nasar, J (1984). *Environmental Aesthetics: Theory, Research and Applications*. In Nasar, J. (ed.). *Visual preferences in urban scenes: a cross-cultural comparison between Japan and the United States* (Cambridge University Press, Cambridge) .
- Punter, J., Carmona, M. (1997) *The Design Dimension of Planning: Theory, content and best practice for design policies* (E & FN SPON, London)
- Siza, A. (1999) *The lisbon Charrette*. In Duarte, J.P.; Bento, J.; Mitchell, W.J (ed.). *Foreword* (IST Press, Lisbon) pp 6 -10
- Stamps III, A. E. (1993a) *Postconstruction Validation og Photomontage Simulations Perceptual and Motor Skills* 20 , pp 1335-1338.
- Stamps III, A. E. (1993b) *Use of Photographs to simulate environments: A Meta-Analysis' Perceptual and Motor Skills* 71 , pp 907-913.
- Stamps III, A. E. (1993c) *Validating contextual urban design protocols: replication and generalisation from single residences to block faces Environment and Planning B: Planning and Design* 20 , pp. 693-707.
- Stamps III, A. E. (1997). *Meta-analysis in environmental research. Space Design and Management for Place Making*, Edmond, pp 114-124.
- Stamps III, A. E. (1998) *Measures of architectural mass: from vague impressions to definite features Environment and Planning B: Planning and Design* 25 , pp. 825-836.
- Steur, J. (1992) *Defining Virtual Reality Journal of Communication* 42 (4), pp 73-93.
- Vassils, B. (1997). *Virtual Reality: A communication tool for urban planning*. In Asanowicz, A., Jakimowitz, A. (eds.) *CAAD-Towards New Design Conventions* (Tecknical University of Bialystock, Germany) pp 45-59.

Geração de Ambientes Rodoviários Para Simulação de Condução

Carlos Campos
Univ. M / ISEP
ccampos@ipp.pt

Vítor Cunha
ISEP
vmc@dee.isep.ipp.pt
R. S. Tomé, 4200 Porto, Portugal

J. Miguel Leitão
ISEP / INESC Porto
jml@dee.isep.ipp.pt

Sumário

Este artigo, apresenta uma ferramenta desenvolvida para a geração de ambientes rodoviários, destinada a simuladores de condução, baseando-se essencialmente na informação de traçado longitudinal, e informação de perfil transversal.

Esta informação pode ser obtida directamente de qualquer projecto de vias rodoviárias. A ferramenta permite modelar também a sinalização vertical e a sinalização horizontal, de forma, a ser utilizada por um simulador de condução.

Os modelos obtidos por esta ferramenta são otimizados para a implementação de simulações visuais interactivas, utilizando técnicas como a organização espacial (árvore de objectos), LOD (level of detail), e o Layer (camadas).

A ferramenta desenvolvida, foi já utilizada na implementação de dois ambientes rodoviários para simulação de condução, demonstrando que, permite gerar ambientes de elevada qualidade, reduzindo simultaneamente o tempo necessário à preparação de experiências de simulação de condução.

A ferramenta apresentada foi posteriormente vocacionada para o simulador de condução “Dris”, (“Driving Simulator”).

Palavras chave: *simulação de condução, representação gráfica, representação vectorial, ambiente rodoviário.*

1. Introdução

Hoje em dia, os simuladores de condução, são importantes, em áreas muito diversas, nomeadamente, na psicologia, e na engenharia de tráfego.

Na psicologia, para estudos de comportamentos dos condutores em diversas situações específicas, como por exemplo, ultrapassagens com risco de colisão frontal. Nesta situação é evidente que o risco humano é nulo, em comparação com uma situação real, daí a preferência pelo uso de simuladores de condução.

Em engenharia de tráfego, este é usado, para estudo dos traçados, não só os já implementados na realidade, bem como os que se encontram em fase de projecto, permitindo experimentar o traçado final com condutores reais.

Independentemente do objectivo do estudo, a realização de experiências num simulador de condução exige a prévia preparação dos modelos dos ambientes rodoviários, onde estes podem facilmente atingir várias dezenas de quilómetros. Estas grandes dimensões e os elevados requisitos de realismo, tornam a criação destes ambientes rodoviários, um processo muito complexo e dispendioso de recursos. Para além destes aspectos, é necessária

informação gráfica, e informação vectorial, do traçado longitudinal e traçado transversal das vias.

As dificuldades são devidas não só à enorme quantidade de objectos elementares que é necessária para preencher de forma realista um ambiente, mas também às imposições de compatibilidade do modelo obtido com os sistemas de simulação e visualização a que se destinam. Um modelo de um ambiente virtual com alguma complexidade pode facilmente atingir um número de polígonos que torna impraticável qualquer tentativa de o especificar de forma manual. Para isso usam-se normalmente ferramentas de modelação que, apesar de permitirem uma redução drástica do esforço necessário, não impedem que o processo de modelação seja considerado como trabalhoso.

Um simulador de condução requer a preparação de bases de dados com a descrição de cenas de regiões geograficamente extensas, com estradas correctamente modeladas em função das normas e tradições do país que se pretende simular. Esta especificidade de ambientes dificulta o aproveitamento de desenvolvimentos

realizados noutros países, mesmo quando os problemas de compatibilidade técnica se apresentem superáveis.

A forma como se organizam os diversos componentes de uma cena é também uma questão importante, como é descrito em [Leitão00], o desempenho dos sistemas de síntese de imagem pode ser drasticamente afectado pela qualidade da representação e organização dos ambientes modelados. Para além dos problemas de compatibilidade com o sistema de simulação e visualização que aparecem associados à informação geométrica do ambiente utilizado, sempre que se deseje incluir participantes autónomos virtuais ou avaliar desempenhos de participantes humanos, é fundamental que outros níveis de informação coexistam com a informação geométrica. É vulgar encontrar, nos simuladores de condução destinados a estudo científico, duas bases de dados distintas mas correlacionadas para especificar um mesmo cenário. Uma destas bases de dados destina-se unicamente ao processo de visualização e contém as representações geométricas dos objectos e as suas propriedades visuais. Para a implementação desta base de dados utiliza-se frequentemente um dos formatos de modelação conhecidos que permitem uma organização optimizada para a visualização em tempo real. A outra base de dados contém informação lógica vectorial que facilita a consulta, em tempo real, do ambiente rodoviário. Esta base de dados inclui normalmente representações de alto nível da rede viária, dos veículos, da sinalização e de todos os outros elementos cuja consulta rápida é importante para a simulação e o controlo dos cenários.

1.1 Trabalhos prévios

Existem muitas ferramentas de modelação 3D disponíveis no mundo, destas apenas uma pequena parte delas, é vocacionada para geração de ambientes, e para simulação em tempo real.

O conhecido software comercial de modelação de ambientes virtuais destinados a aplicações de tempo real *Multigen Creator* [MultiGen99] disponibiliza, em opção, um conjunto de ferramentas adicionais específicas para a modelação de estradas e ambientes rodoviários. Este conjunto de ferramentas é baseado em software desenvolvido no âmbito de um simulador de condução construído na Universidade de Iowa

No entanto, é reconhecido que a preparação de ambientes para simuladores de condução possui requisitos específicos, que não podem ser completamente satisfeitos por este ou outro software comercial disponível.

Uma estratégia de modelação de estradas para simuladores de condução mais próxima da aplicada na ferramenta descrita neste artigo, foi apresentada em [Baya98]. Esta estratégia demonstra já grandes possibilidades, no entanto, a implementação realizada não é adequada a todas as realidades rodoviárias e ambientes de simulação.

A estratégia a seguir, consiste em desenvolver uma ferramenta, em que à entrada exista duas fontes de informação, perfil transversal e eixos das vias. E à saída dessa ferramenta, duas bases de dados, representação gráfica, representação vectorial. As fontes de dados à entrada serão de fácil precessão, em que utilizador, consiga manipular facilmente, permitindo de certa forma uma interacção com o ambiente rodoviário gerado. Outra característica, será a portabilidade da ferramenta para outros sistemas.

1.2 Organização do documento

No capítulo 1, apresenta-se uma introdução às finalidades dos simuladores de condução, a dificuldade na criação de ambientes rodoviários específicos para realização de experiências. Breves referências a trabalhos prévios, assim como conceito geral da ferramenta desenvolvida. No capítulo 2, encontra-se descrito de uma forma geral os aspectos mais importante envolventes na implementação da solução, bem como a implementação da ferramenta, tratamento de dados e descreve-se o fluxo de informação. No capítulo 3, são descritos exemplos em que a ferramenta foi utilizada para gerar o ambiente rodoviário, com elevado nível de satisfação, e reduzido nível de complexidade para o preparador da experiência. No capítulo 4, são mencionadas as conclusões finais, a suas vantagens, bem como desenvolvimentos futuros. No capítulo 5, são apresentadas as referências bibliográficas.

2. Implementação

2.1 Definição de eixos das vias

Num projecto de engenharia civil, os eixos das vias, podem ser definidos por um conjunto de: Rectas; Arcos de circunferência; Clotoides. Uma clotoide, ou espiral de cornú (figura 1), trata-se de uma curva paramétrica, definida pela equação: $s.\rho=A^2$, descrita em [Javier03], em que:

- s, longitude do arco (comprimento do arco)
- ρ , raio de curvatura
- A, parâmetro da clotoide

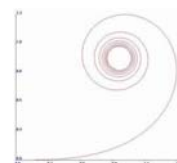


Figura 1 – Clotoide

A partir desta informação é extraído um conjunto de pontos 3D, a união dos vários pontos por pequenos segmentos de recta, dão origem a uma polilinha, constituída por pequenos segmentos de recta, interligados entre si. Na figura 2, pode se observar, a união dos pontos 3D, formando pequenos segmentos de recta, que por sua vez formam a polilinha.

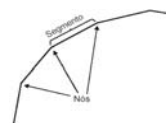


Figura 2 – Polilinha.

Entende-se por polilinha, uma sequência (linha) de segmentos de recta, segmentos estes originados pela

união de pontos 3D, amostrados das: rectas, arcos de circunferência e clotoides.

O preparador de uma experiência de simulação de condução pode utilizar esta ferramenta em conjunto com um vulgar editor de texto para especificar todo ambiente de simulação.

Como se pode observar pela figura 3, temos duas fontes de informação de dados à entrada, que devidamente tratados pela ferramenta implementada, geram duas saídas de dados, de diferentes tipos.

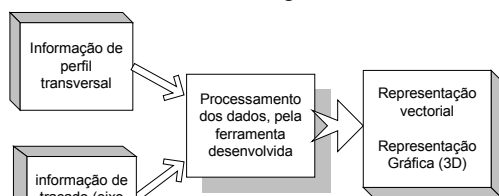


Figura 3 – Diagrama de dados.

2.2 Obtenção dos polígonos

As vias, tal como os outros objectos 3D, serão representadas por uma aproximação poligonal. Ou seja, as vias serão representadas por uma cadeia de polígonos, mais propriamente de triângulos, a toda a largura das vias, como se apresenta na figura 4.

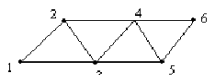


Figura 4 – Representação poligonal por triângulos.

Para a obtenção dos vértices dos polígonos que compõem as vias, utiliza-se cálculo vectorial [JA98], como se apresenta na figura 5. Para cada nó da polilinha, determinam-se os vectores correspondentes às direcções (v_1 e v_2) do segmento a montante e a jusante desse mesmo nó. Usando métodos matemáticos normalizam-se esses vectores, e somam-se um ao outro, obtendo-se assim um vector com a direcção, resultante da soma de v_1 com v_2 . Com o ponto correspondente ao nó, a direcção do vector soma de v_1 e v_2 , o valor da largura da via, descrito no perfil transversal, obtêm-se os pontos p_1 e p_2 . Desta forma, são obtidos todos os vértices dos polígonos da via.

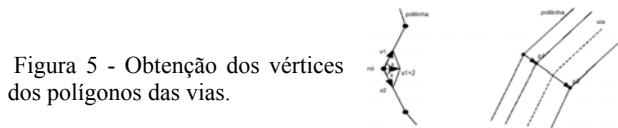
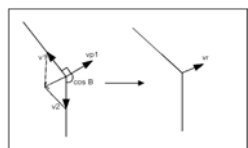


Figura 5 - Obtenção dos vértices dos polígonos das vias.

Quando o vector resultante, tem sentido contrário ao pretendido, essa situação é detectada usando cálculo vectorial, sendo invertida a sua direcção, passando a ter direcção oposta, como se apresenta na figura 6.

Figura 6 - Determinação da direcção correcta do vector director.



No caso de os dois segmentos serem colineares, essa situação também é detectada vectorialmente, sendo corrigida com um vector perpendicular a um dos segmentos, garantindo a correcta representação da via.

Nas extremidades da via, é determinado um vector com direcção perpendicular, no início, ao primeiro segmento (vp_1), no fim ao ultimo segmento (vp_2), calculando-se assim os vértices dos polígonos correctamente, como se pode ver na figura 7.

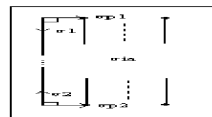


Figura 7 - Determinação dos vértices dos polígonos, nas extremidades das vias.

Cruzando a informação do perfil com o traçado obtêm-se a representação geométrica das vias. A ordem de obtenção dos vértices dos polígonos é importante, pelo facto de o vector normal resultante, em conjunto com o modelo de iluminação utilizado, produzir uma face visível.

A ferramenta implementada, é capaz de criar polígonos ao longo da via, com passo adaptativo [Baya98], ou seja, nas rectas não é necessário uma elevada saturação de pontos em comparação com as curvas, o mesmo acontece com os diferentes raios de curvatura. Em curvas com raios mais pequenos, a saturação de pontos é mais elevada, como se pode observar pela figura 8.



Figura 8 – Variação do intervalo entre pontos.

Na construção das vias, estas são divididas em segmentos mais pequenos, interligados entre si, e organizados no espaço.

Em conjunto com a organização espacial, utiliza-se também o conceito de "LOD", fazendo a selecção entre vários modelos dependendo da distância a que o observador se encontra, em que os vários modelos têm diferentes níveis de detalhe [sgi][Leitão00][Campos02].

2.3 Fluxo de informação

A polilinha que representa os eixos das vias, associado ao perfil transversal, resulta em uma representação gráfica, tal como se pretende.

Na figura 9, é apresentado um diagrama de blocos, para melhor compreender o que foi descrito.

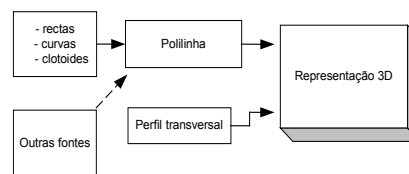


Figura 9 – Fluxo de informação.

Considera-se outras fontes, qualquer conjunto de pontos tridimensionais, que formam uma polilinha.

O perfil transversal é definido num formato próprio criado para o efeito, onde para além da informação de perfil transversal pode ser encontrada também outro tipo de informação, tal como: Textura do mundo; Largura da via, desnível da via (à direita e à esquerda), “offset” lateral; Textura a ser aplicada à via; Toda as linhas existentes, com diversos tipos (descontínua, contínua e mistas), a sua localização, espessura, entre outras; Sinalização vertical; Todo o tipo de objectos envolventes, nomeadamente: árvores, edifícios, postos de abastecimento, etc.

Para a colocação da sinalização vertical, são necessários alguns parâmetros, nomeadamente: posição longitudinal (medida em metros e ao longo do eixo da via), posição lateral (transversal), e ângulo de rotação segundo eixo dos z's, relativamente à via, permitindo ajustar a visualização do sinal vertical.

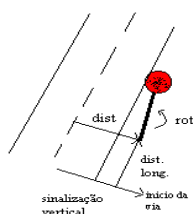


Figura 10- Representação da colocação da sinalização vertical

A figura 10, ilustra a representação do sinal vertical, e o que cada parâmetro corresponde, aquando da sua colocação ao longo das vias. O tipo de sinal a colocar numa determinada posição da via, é escolhido pelo utilizador, cabe a este definir o sinal em conformidade com a característica da via.

A colocação de objectos envolventes, também obedecem a estes requisitos (parâmetros), quando se trata da colocação de um objecto uma vez (uma referência por colocação). Quando se pretende representar o mesmo objecto várias vezes seguidas com um intervalo de repetição constante (uma referência para várias representações do mesmo objecto), além dos parâmetros anteriores, é necessário referir o intervalo de repetição. Deixa de existir a designação, distância longitudinal, que por sua vez passa a corresponder um “offset” inicial, também é definido o fim da repetição.

As linhas de sinalização horizontal, podem ser de vários tipos: contínua, tracejada, ou mista. A todas elas são comuns os seguintes parâmetros: posição lateral, espessura, cor, o início e o fim de representação ao longo da via. Para as linhas contínuas, estes parâmetros são suficientes, para as linhas tracejadas, é acrescida a relação espaço, traço. As linhas mistas são uma composição destes parâmetros, linha contínua e tracejada. As linhas mistas necessitam ainda da distância entre linhas (visto serem compostas por mais que uma linha).

A representação gráfica das linhas de sinalização horizontal, consiste também numa cadeia de triângulos, sobreposta à estrada. Para a representação das linhas de sinalização horizontal conjuntamente com as vias é utilizada uma hierarquia em camadas, “layers”, de forma a facilitar a detecção de visibilidade. Os “layers” são utilizados para definir prioridades de representação, especificando qual o objecto que se sobrepõe a outro. Em sistemas de visualização em tempo real, o uso de “layers”, aumenta significativamente a qualidade da simulação.

A especificação das vias é incluída no ficheiro de descrição mundo (*.wdf), já implementado, sendo este actualizado e sem perda de compatibilidade com versões mais antigas, deste ficheiro.

A compatibilidade, entre o simulador de condução e a ferramenta desenvolvida, implicou refazer alguns módulos já existentes no simulador, nomeadamente, um dos módulos dos veículos autónomos [Leitão00]. A nível da representação gráfica, esta é totalmente independente de todo o resto de funcionalidades, no módulo do simulador é efectuada uma chamada à rotina responsável pela criação gráfica. A nível vectorial, o simulador passa a ter conhecimento directo da informação de perfil transversal, para isso, foi necessário introduzir essa informação nos módulos já implementados.

Todos os conceitos abordados neste capítulo, encontra-se documentados detalhadamente em [Campos02].

Foram desenvolvidas ferramentas acessórias, nomeadamente para a manipulação da polilinha. Podendo essas ferramentas fazer: colagem, translação, rotação, escalamento e “offset” de polilinhas.

3. Exemplos de aplicação

A ferramenta de modelação desenvolvida, foi já utilizada para a implementação de dois ambientes rodoviários distintos, destinados a estudos científicos de simulação de condução.

Em ambos os casos, foi reconhecida a eficiência da ferramenta desenvolvida, para a execução destas tarefas, pelas vantagens oferecidas em termos de rapidez do trabalho de modelação, precisão, qualidade dos modelos constituídos, facilidade de alteração e afinação de versões anteriores do modelo, e compatibilidade com outros subsistemas do simulador de condução.

3.1 Projecto “Haste”

No âmbito do projecto, “HASTE” (Human Machine Interface And the Safety of Traffic in Europe) GRD1-2000-25361, foi especificado um ambiente rodoviário, destinado à realização de vários ensaios experimentais semelhantes, em diversos simuladores de condução distintos.

O ambiente especificado é constituído por uma estrada principal de extensão superior a 29 quilómetros, e algumas secundárias que a interceptam, na figura 11 pode

ser observado um dos cruzamentos, especificados para o projecto “HASTE”.



Fig. 11 - Projecto “HASTE”, cruzamento.

A ferramenta desenvolvida, permitiu a implementação, da versão portuguesa deste ambiente rodoviário, adequada ao simulador de condução “Dris” [Leitão97]. Para além disso, esta ferramenta permitiu também a modelação rápida de uma primeira versão de estrada principal, que foi testada em diversos simuladores durante a fase de especificação.

3.2 Projecto “Estrada nacional 222”

Outra aplicação da ferramenta de modelação de ambientes rodoviários, consistiu na reprodução virtual de um troço com mais de cinco quilómetros da estrada nacional 222. O objectivo deste trabalho, é permitir a realização de experiências paralelas em ambientes reais e virtuais.

O modelo obtido deverá ser utilizado proximamente num estudo de manobras de ultrapassagem com risco de colisão frontal. Para esta implementação foram utilizados dados obtidos do projecto da estrada real, nacional 222.

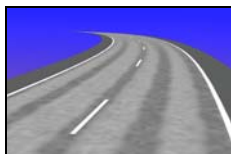


Figura 12 - Vista da estrada nacional 222

Na figura 12, é apresentada uma imagem da estrada N222, ainda em fase de desenvolvimento, e em que se pode observar a existência de uma subida em curva à esquerda.



Figura 13– Mapa de localização.

Na figura 13, encontra-se assinalada a localização, no mapa de Portugal, do troço implementado, da estrada nacional 222.

4. Conclusões:

A criação de modelos de ambientes rodoviários para simuladores de condução, é um processo complexo, devido à necessidade de manipular numerosos modelos de objectos. As ferramentas comerciais de modelação 3D, não são adequadas à optimização deste processo.

Este artigo descreve uma ferramenta de modelação de ambientes rodoviários que permite a modelação

automática de estradas, a partir da informação geométrica do eixo da via e do perfil transversal.

Os modelos obtidos por esta ferramenta são optimizados para simulação visual interactiva, e adequados à integração num simulador de condução, como demonstrado pelas aplicações já concretizadas.

Esta ferramenta permite obter modelos de excelente qualidade e reduzir drasticamente o trabalho de preparação de experiências de simulação de condução, assim como custo, visto não ser necessário a modelação integral do ambiente por parte do utilizador.

A metodologia utilizada permite gerar qualquer tipo de via, como por exemplo: estradas rurais, variantes, auto-estradas, com baixo nível de complexidade para o utilizador, como ficou demonstrado pelas aplicações já implementadas, bem como a sua total portabilidade para outros sistemas.

Está em desenvolvimento uma funcionalidade, capaz de modelar o terreno envolvente, em que os seus resultados poderão ser observados brevemente.

Encontra-se ainda em fase de estudo uma funcionalidade que permite a criação de estradas não planas.

5. Referências:

[Baya98] S. Bayarri, I. Pareja, I. Coma e M. Fernández; Modelação de Carreteras para la Simulación de Conducción; 8º encontro português de computação gráfica, 1998.

[Campos02] Carlos Campos; Geração de Ambientes Rodoviário para Simulação de Condução; Instituto Superior de Engenharia do Porto, 2002

[Leitão97] J. M. Leitão, A. Coelho & Ferreira, F.N. (1997). Dris- A virtual Driving Simulator. Proceedings of the Second International Seminar on human Factors in Road Traffic, ISBN 97-2-8098-25-1.

[Leitão00] João Miguel Queirós Magno Leitão; Agentes Autónomos Controláveis em Simuladores de Condução; Faculdade de Engenharia da Universidade do Porto, 2000

[JA98] Jorge Lagoa, António Ramos; Trigonometria, 1998;

<<http://www.terravista.pt/Ancora/2461/matematica/trigonometria/IndiceTrigonometria.html>>, 2003

[Javier03] Javier Jiménez Shaw ; Solución de la ecuación de la clotoide (espiral de Cornú); <<http://javier.jimenezshaw.com/clothos/download.html>>, 2003.

[MultiGen99] MultiGen-Paradigm, Inc. MultiGen Creator - Premier Tools for Realtime 3D Database Development, Descrição disponível na Internet, em:

<http://www.multigen-paradigm.com/products/pdf_files/Creator_72dpi.pdf>, 1999.

[sgi] Silicon Graphics. OpenGL Performer.

<<http://www.sgi.com/software/performer>>, 2003.

Using Sketches to Edit Electronic Documents

Pedro Miguel Coelho Gonçalo França Faria
Instituto Superior Técnico
Av. Rovisco Pais, 1000 Lisboa
{pmc, gfs}@mega.ist.utl.pt

Daniel Gonçalves Joaquim A Jorge
Computer Science Department, IST
Av. Rovisco Pais, 1049-001 Lisboa
djvg@gia.ist.utl.pt, jorgej@acm.org

Abstract

Calligraphic interfaces mimic the interaction of users with pen and paper, to which most persons are used to from an early age. This makes them a privileged way of interacting with computers in efficient and natural ways. One of the tasks more often made with pen and paper is to write and correct text documents. This is done resorting to several symbols with specific meanings, representing the changes to be made in the document. It should be possible to correct electronic texts in a similar way, using calligraphic interfaces. In this paper, we describe the CaliEdit text editor, currently being developed for PalmOS devices with that goal in mind. In its development, we followed a user-centered design approach. The task analysis phase, with the help of questionnaires, identified the most common symbols for the most common text-correcting tasks. The editor itself uses the CALI shape and gesture recognizer, ported to PalmOS, to help recognize those symbols. Several porting and implementation problems were overcome. Currently, heuristic and usability evaluation studies are underway to help validate the editor.

Keywords

Calligraphic interfaces, gesture recognition, mobile devices, text editing, user-centred design.

1. INTRODUCTION

Calligraphic interfaces are a novel way of interacting with computing devices. They try to mimic the ways in which users interact with paper documents, allowing them to enter and manipulate data simply by drawing or sketching. Several heuristics and constraints can then be used to correctly interpret the users' wishes. This allows a more efficient, streamlined and natural interaction. Indeed, most persons are used to interacting with pen and paper from an early age. Also, paper allows a more flexible interaction than usual computer peripherals, such as keyboards and mice. Furthermore, even computer-unskilled persons can use pen and paper efficiently. This makes calligraphic interfaces suitable for a wide range of users and situations where the limited traditional interaction ways could pose serious problems.

Calligraphic interfaces have been used successfully in several applications in the past. The Java SketchIt project [CAETANO'02] allows dialogue boxes to be created from rough sketches of their composing elements. GIDeS [PEREIRA'03] provides a mechanism for the creation of technical drawings from sketches, with the help of spatial constraints. They have also been used for the retrieval of technical drawings, from rough sketches of what is to be found [FONSECA'03].

Until recently, special hardware such as touch screens or digitizing tablets was required to use calligraphic interfaces. However, with the dissemination of Personal Digital Assistants (PDAs) and the appearance of Tablet PCs, that hardware is now becoming commonplace. Those

devices have interfaces based on direct manipulation of elements on the screen using a special pen or stylus (although often mimicking a mouse). The infrastructure for the use of calligraphic interfaces is already in place.

One of the most usual tasks on those devices is text editing. However, common text editors are based on traditional interaction modes, such as sequential text entry, which requires a cursor to be set on the right position. Also, the edition of the text itself is done, for the most part, resorting to menu options or toolbars. Since PDAs and Tablet PCs are much closer, in terms of portability and format, to paper pads, it would benefit users to allow them to manipulate their texts using in the same way they use when working on paper.

The CaliEdit system is an application under development as a year long undergraduate project at IST that combines calligraphic interfaces with a traditional text editor. It allows users to manipulate texts by directly drawing over it symbols representing the most common editing tasks. It recognizes several common symbols users draw on paper to represent desired changes to a text, and effects those changes. CaliEdit uses the CALI gesture recogniser [FONSECA'00], freely available on the Web [CALI] to identify the symbols.

Users played a central role throughout the development of CaliEdit. In an early task analysis phase, several user studies were conducted to discover what are the most common and natural text-editing symbols. Currently, they have been actively engaged in the project through the performance of heuristic and usability evaluations.

User-centred design is of capital importance due to the novel approach to text editing embodied by CaliEdit.

The following section describes how the relevant symbols were identified. Then, the choice of the platform on which CaliEdit was implemented will be described. In the next section several relevant implementation issues will be discussed, followed by some conclusions and references to current and future work.

2. SYMBOLS

Attempts to find a set of standard symbols for text editing both in libraries and on the Internet proved to be unsuccessful. It soon became apparent that there isn't a standard, universal, symbol set for that the correction of paper documents. However, it was desirable to mimic the usual forms of interaction as closely as possible. Determining the most commonly used symbols was of capital importance. Two questionnaire-based studies specially designed to collect the desired information were conducted. Although the first provided interesting results, sometimes a high level of ambiguity (two or three symbols for the same function) didn't allow conclusive results to be extracted. The second questionnaire was created with those ambiguities in mind in order to solve them. The symbols thus obtained are the ones currently in use in CaliEdit.

Before the design of the questionnaires, the set of functions that CaliEdit should perform was defined. Those functions are the most common in text editing:

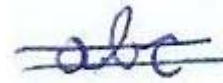
- delete text
- move text
- insert characters in the middle of words
- insert words in the middle of sentences
- insert paragraphs
- insert tabs
- insert spaces

Evidently, this was an open set that could be changed in the course of the analysis of the questionnaires, if some new relevant tasks came up. As it turned out, it no changes were necessary.

The design of the questionnaires was based on the guidelines available on the web site of the Human Machine Interface course at IST [IHM]. They were composed by two small texts. The first text was flawless and the second was a copy of the first in which several common typing and spelling errors were introduced. The user was asked to correct those errors. In theory, the errors in the text would require the use of all the previously chosen editing functions. Furthermore, all errors were introduced in a way in which correcting one wouldn't affect the correction of others. This was done to avoid confusing the users, and because we were trying to identify symbols associated to only one correcting task.

After analysing both questionnaires, the following symbols were found to be the most common when correcting texts in paper documents:

- To delete text



- To move text



- To insert new character



- To insert new words



- To insert a new paragraph



- To insert a space



- To insert a Tab



3. CHOOSING THE PLATFORM

In order to decide in which platform to implement CaliEdit, a comparative study of the three most likely candidates (Palm OS, Pocket PC, and Emacs on a Tablet PC) was performed. This study consisted of an in-depth comparison of all three platforms in order to help choose the one that best fitted the CaliEdit requirements.

The study focused on the text editing capabilities made available by the APIs of the three platforms. The ability to integrate the CALI gesture recognizer into whichever platform was chosen was also studied. The recognizer itself is written in C++ but portability issues could arise, mainly for the PDA platforms, due to their special hardware restrictions. Furthermore, the CALI library receives a set of coordinates, corresponding to the places travelled by the stylus during the drawing of a gesture, returning a list of gesture names, ordered by the probability of corresponding to the given set of the points. The point coordinates must be captured by our application and passed to

the recognizer. Hence, the comparison also took into account the capability of each platform to capture high-resolution screen coordinates.

In this study, the Internet was an invaluable source of information, allowing access to relevant documents, manuals, developer environments and code samples. A technical report where more detailed results of this study can be found is located at <http://mega.ist.utl.pt/~pmc>.

3.1 The PalmOS Platform

The PalmOS is an event driven operating system. As such, all applications have three stages in their execution cycle: the startup, the event loop and the finish. The interaction with the user generates events that are processed in the event loop. Some events are directly related with the pen movements. Using those events it is possible to collect a sample of point coordinates and therefore recognize the gesture/symbol drawn by the user. PalmOS also provides a complete API to manipulate text. Those functions are useful in the implementation of CaliEdit. This platform has some disadvantages mainly because of its small screen size and the need to implement a basic text editor in which to include the calligraphic interface.

Some further experiments were made. A simple application that paints on the screen the points where the stylus touches it was implemented. This small application provided insights on both how to program in the PalmOS system in general, and how to collect and store a set of coordinates to later pass to the recognizer. A version of CALI to the PalmOS system as also developed.

3.2 The Pocket PC Platform

Windows CE is the operating system of the Pocket PC and, like PalmOS, it is an event driven system. Windows CE also provides events that reflect the movements of the pen, and functions that manipulate text. Both platforms are very similar in what can be implemented in them.

The Pocket PC has the same limitations of PalmOS. Additionally, some experimentation showed that it's impossible, apparently, to draw inside a text box. This could pose a serious difficulty in the implementation of CaliEdit, since drawing the symbols provides an important feedback to users and cannot be discarded. Although this problem might have a solution, we didn't spend too many resources to find it.

A Windows CE version of the simple point painting application we created for PalmOS was implemented. This allowed the direct comparisons between the two, to better identify the relevant differences. We concluded that in Windows CE and using the .Net platform to implement applications prototypes can be build much faster than for PalmOS. To integrate the CALI into Windows CE applications, a dynamic link library (dll) was built, allowing applications to easily access the CALI API.

3.3 Emacs on Tablet PC Platform

With Emacs there is no need to implement the text editor. Emacs is extensible thought a special Lisp dialect (Emacs-LISP). To implement CaliEdit, it is fundamental

to know the position of the pen. Emacs-LISP, however, only gives access to the coordinates of the text cursor. Those coordinates are measured in characters. The cursor is considered to be located between two characters and not in a specific pixel. So, with Emacs, the major problem is the inability to obtain the information of the pointer device on the pixel level. Several modifications of Emacs' source code to provide the capacity to collect high-resolution coordinates were tried, but all were unsuccessful. Extending Emacs-LISP with a special-purpose function to obtain those coordinates seems impossible, or, at least, not feasible given the resource constraints of this project.

3.4 The Chosen Platform

Overall, the PalmOS platform seemed the best and was chosen to implement CaliEdit. Emacs on Tablet PC was the initial preferred option (mainly due to the versatility provided by a bigger screen) but had to be put aside due to the inability to capture high-resolution point coordinates. Without this capability we can't collect a good sample that represents the symbols drawn by the user. CALI was ported to both the Pocket PC and PalmOS platforms. So, the choice of the PalmOS system was made mainly due to the difficulties in mixing text and drawings on the Pocket PC. Also, more support is available through the PalmOS community.

4. IMPLEMENTATION

Some of the symbols CaliEdit recognizes had to be different from those collected during task analysis. For example, on paper when a character is missing in the middle of a word, people use an arrow with the missing character on top. While on paper this sounds good, in a text editor it's much simpler to use a tap to place the cursor at the desired position and then simply write the character. On paper users can't directly change the text we can do so on a computer. In order to achieve not only a natural interface but also an efficient one, a compromise between what users can do on paper and what they can do on a computer was made.

All text modifying capabilities of CaliEdit were implemented with the PalmOS API as a basis. The CaliEdit itself collects the coordinates of the points that define a gesture, invokes the CALI recognizer to identify that gesture and then chooses the appropriate API calls. The portion of text to be affected by the gesture is computed automatically from special coordinate sets, such as the gesture's bounding box, and the font metrics.

4.1 Adapting CALI to CaliEdit

The algorithm implemented in CALI recognizes simple geometric shapes like triangles or rectangles. The shapes are recognized independently of changes in rotation, size or number of individual strokes. This library was written in C++. The source files of CALI were used instead of the existing pre-compiled library, since it was necessary to port it to other platforms.

There were some problems porting CALI to PalmOS. One of the biggest resulted from the memory limitations

inherent to that operating system. The application experienced several stack overflows, especially when recognizing ellipses. This turned out to be the result of the large number of interpolated ellipse points calculated by the CALI recognizer. These points allow applications that so desire to replace the rough sketch drawn by the user with a more accurate shape. CaliEdit doesn't need this to happen. As a result, the CALI library was altered to stop doing those calculations. Some memory leaks were also found, made visible by PalmOS's memory limitations. Eventually, all known memory leaks were solved, although some may still remain undiscovered.

4.2 Making CaliEdit a Palm-like Application

At first, CaliEdit's user interface was structured as the ones found on text editors on desktop PCs. It soon became obvious that this wasn't the best way to develop a Palm application. Both the hardware constraints and the ways of interaction are different. A PC-like would be fraught with usability problems and be perceived as strange by a Palm user. Instead, we looked closely at a standard PalmOS text editor, MemoPad, modifying CaliEdit to make it similar to that application.

Overall, the user interface is structured as follows: the first screen presents a list of documents already present on the PDA, from which the user can choose one to modify. Alternatively, a new document can be created with the help of a button. Once a document is selected, the application moves on to the text-editing screen. Again, this is similar to MemoPad. However, two extra buttons inexistent in MemoPad were added to this screen. The first allows users leave the document without saving and the second maintains the original text and saves the modified text in new document, just like the 'Save as...' function in desktop text editors.

Some standard PalmOS functionalities are present in CaliEdit. It is the case of tapping on a word twice to select the whole word, and thrice to select the line the word is on. The usual PalmOS menus (copy, paste, etc.) were also implemented in CaliEdit. Finally, text can be written using Graffiti, as usual for PalmOS applications.

5. CONCLUSIONS AND FUTURE WORK

A text editor where symbols commonly used on paper to signal required corrections can be used was implemented. One of our goals is to achieve a more natural interaction with text documents. Hence, the next steps in our research will be to perform heuristic and usability evaluations [DIX'97]. Those evaluations will help us find interface problems, and will validate the choices made so far regarding its design.

In the heuristic tests, a Palm with our editor will be made available to 4 persons with knowledge in the HCI field.

All errors found by the evaluators will be corrected before the usability tests. In the usability tests we will give users some time to get acquainted with the editor. Then, we will ask them to perform some editing tasks, both using MemoPad then CaliEdit: creating a document, writing something and then saving it, and correcting several errors in a pre-determined text. We intend to interview around 30 persons, half of which will use MemoPad first, while the other half will start with CaliEdit. With this, we hope to prevent biasing the results due to previous knowledge of the texts to correct. Our goal is to show that CaliEdit has better performance than conventional editors like MemoPad. We'll collect information on both the speed and number of errors made by users while performing the tasks. We will also try to discover if users are subjectively more satisfied with CaliEdit than with ordinary editors and with paper and pen. We'll compare the satisfaction levels with the time gained or lost by using CaliEdit. Finally, changes to the interface prompted by the tests will be implemented and validated iteratively.

6. REFERENCES

- [CAETANO'02] A. T. Caetano, N. Goulart, M. J. Fonseca and J. A. Jorge. JavaSketchIt: Issues in Sketching the Look of User Interfaces. AAAI Spring Symposium on Sketch Understanding, 25-27 March, 2002, Palo Alto, California.
- [CALI] The CALI Homepage: <http://immi.inesc.pt/cali/>
- [DIX'97] Alan J. Dix, Janet Finlay, Gregory Abowd and Russel Beale. Human-Computer Interaction, second edition. Prentice Hall, 1997, ISBN 0-13-239864-8
- [FONSECA'00] M. J. Fonseca and J. A. Jorge. Using Fuzzy Logic to Recognize Geometric Shapes Interactively. In Proceedings of the 9th Int. Conference on Fuzzy Systems (FUZZ-IEEE 2000). San Antonio, USA, May 2000.
- [FONSECA'03] M. J. Fonseca and J. Jorge. Indexing High-Dimensional Data for Content-Based Retrieval in Large Databases. 8th International Conference on Database Systems for Advanced Applications (DASFAA '03), Kyoto, Japan, March 2003.
- [IHM] The Human Machine Interface Course at IST site: <http://mega.ist.utl.pt/~ic-ihm/>
- [PEREIRA'03] J. P. Pereira, J. A. Jorge, V.A. Branco, F. N. Ferreira. Calligraphic Interfaces: Mixed Metaphors for Design. Tenth Workshop on the Design, Specification and Verification of Interactive Systems, DSV-IS 2003, Funchal, Portugal, 4-6 June 2003.

Task Analysis and Scenario-Based Design of Calligraphic Interfaces

Filipe Marques Dias, Manuel J Fonseca, Joaquim A Jorge

Department of Information Systems and Computer Science

INESC-ID/IST/Technical University of Lisbon

R. Alves Redol, 9, 1000-029 Lisboa, Portugal

fil@immi.inesc-id.pt, mjf@inesc-id.pt, jorgej@acm.org

Abstract

The great functionality of today's CAD systems enables the manufacturing of very complex models. Yet, great part of this power is unusable, early in the design process. As well, we have found two different approaches in two different car designers that surprisingly lead to the same long time in development. In this paper we identify the reasons for these problems, and propose a solution that bridges the different worlds of conceptual and geometric models, using the affordability of the calligraphic interface. Creating a prototype by drawing it allows impossible situations to be immediately identified and corrected, without further communication overheads, certifying that emotional drawing ideas correspond to the designer's intents.

Keywords

Task Analysis, CAD systems, Calligraphic Interfaces.

1. INTRODUCTION

CAD systems, being in the possession of great functionality, now enable the manufacturing of very complicated models. However, computers are yet to be used in the early stages of product design. This derives from designers not being acquainted with the representation, and needing to leap large conceptual gaps from their mental models to the geometric models.

We also noticed that two different car designers, following two different paths to designing a vehicle, spend the same time to perform the overall activity. In one case, traditional paper and pencil is used as medium for storing and passing information, while in the other case digital support is preferred. Designers execute their work over these mediums, and have to communicate to CAD operators or other engineers. Designers create new drawings rather quickly and so do modellers, yet a car takes several months to go through the phase of design and modelling. Our task analysis provides an answer for this. The communication between designers and CAD operators required for the modelling phase is a bottleneck. Reasons underlying this are related to different interpretations of the drawings inevitable from both parts involved in the communication.

Our proposal to solve these problems is the SmartSketches Project [SmartSk]. Using this system a designer will be able to model in a way close to how he draws, holding a stylus on either a tablet, or directly over the screen.

Input modality is not the only concern to shorten the distance between system and designer. More important user interface issues are taken into account as well. Our system will provide a computer-assisted environment, enabling some automated modelling and simpler and more natural interaction metaphors.

In the remainder this paper we describe the task analysis carried out, as well as the outcome conclusions subjacent to our observation.

2. RELATED WORK

For the modelling process, several systems exist that take care in user interaction to support easier interaction and modelling schemes. The most famous work in this area is SketchPad from Sutherland [Sutherland63], which is the first graphical user interface that allows precise drawing using modelling hierarchy and a constraint-based system. Later, in the 90's, Zeleznik's SKETCH [Zeleznik96] combines gesture and geometric recognition for creating and modifying 3D objects. His approach defines a gesture grammar for creating simple primitives in an orthogonal view. In Teddy [Igarashi99], the modelling of free-form surfaces is allowed with an unprecedented simplicity at the interface level. Silhouettes are the basis of the object modelling process. This approach, then, provides a polygonal mesh that adapts to the object's silhouette drawn. Architecture also benefited from this field when SketchUp [Sketchup3D] was introduced. This system offers operations more advantageous from conventional Boolean methods, like direct manipulation over edges and faces. Pereira *et al.* proposes a system, GIDes [Pereira00], which constructs 3D objects drawn in

isometric perspective projection, with the use of gesture recognition. GIDes defines a gesture vocabulary for a reduced command set of geometric primitives, and operations. A valuable feature presented by this work is an expectation list that suggests to the user the possible operations to perform, during interaction.

Regarding modelling concerns, Terzopoulos and Qin define and present Dynamic Non-Uniform Rational B-Splines (or DNURBS) [Terzopoulos94] as a new modelling scheme. D-NURBS allow the usual indirect manipulation for definition of control vertexes and weights, and a direct manipulation using forces to deform the Splines using physical simulation. Zhang and Qin extend this definition to incorporate hierarchy [Zhang01]. The hierarchy brings the ability to edit fine details in localized regions, resorting to local knot insertion, avoiding the insertion of unnecessary patches.

Our system will require the ability to produce high quality modelled curves and surfaces, not being suited for modelling using the physics simulation directly. However, recognition of stroke gestures and a suggestive way of reducing the command set by dealing with ambiguity are advantages to be explored.

3. TASK ANALYSIS

In this section we briefly describe the visited partner's workflow, tasks they perform as well as the relationship between the tasks and the people involved. For further detailed information we refer the reader to [Dias03].

3.1 Site Profiles

Italdesign Giugiaro provides services to the automotive companies, developing all the design phases of a new vehicle, from the first styling idea to the engineering of the project, and finally to the manufacturing of the running prototypes. Italdesign Giugiaro has always been investing important resources in new technologies in order to improve the end-product quality, to reduce costs and times of production and to optimise processes.

FIAT Centro Stile is an advanced styling research centre, providing services to the FIAT Group. Their research focuses on the concept design phase, providing style and car designs for the main manufactures of the FIAT Group. FIAT Design Centre employs half a dozen designers and as many modellers who work in small teams of two and three on new vehicle designs. At certain stages of car designs physical modellers will be called in to produce scale models of cars being designed.

3.2 User Profiles

Typical users of the system can be divided in two classes: Designers and Computer Aided Styling (CAS) engineers. There is also an interesting proportion of one-to-one between the two kinds of users at every visited site.

Designers' main activity is the creation of free form sketches and drawings. They typically have a degree in art or design, are familiar with sketching and painting and possess good to excellent manual skills. Designers



Figure 1 - Perspective sketch

do not necessarily work on computers. Those that do only use paint software and not CAD systems.

CAS engineers are those that create 3D models out of sketches and hand drawings performed by designers. Typically, they have a degree in mechanical engineering or in design with a technical approach. They are able to produce 3D geometric models and surfaces with a good realistic quality.

4. WORKFLOW

In this section we look at the flow of work at each site describing hierarchical, precedence and temporal relations among major activities and tasks. This information was acquired through internal interviews and workshops organized in the scope of SmartSketches, with consortium partners. The analysed case study consists in the development of a motor show prototype.

4.1 Italdesign Giugiaro

Italdesign Giugiaro workflow starts by creating/analysing the package provided by the customer. This package defines the ranges of freedom for the designer. Next, designers create a perspective sketch of the car that will be the basis for technical sketch. This technical sketch is then assembled in a 4-view drawing, which will be the starting point to the computer aided modelling. After the construction of the 3D model of the car they create a virtual model to present to the client. Finally, at the end of the process they produce a physical 1:1 scale model.

4.1.1 Package creation

The first step in the styling process is the creation/analysis of the package, which consists in a technical drawing fixing the main hard-points that are to remain untouched during the development of the styling. These bonds put by the customer change according to the different carmaker, to the kind of car developed (sport car, sedan, etc.) and to the project type (restyling vs. completely new model). The time needed to develop a complete package drawing can go from a few weeks, if the package is fairly simple, to several months, if all the mechanical requirements are analysed.

4.1.2 Perspective and Technical Sketches

The first creative step is the development of perspective sketches. This is when various styling proposals are studied based on the package requirements, as a technical input, and on the styling intention of the carmaker, as a guideline for the look of the car (as figure 1 illustrates).

Depending on the customer's requirements, this phase can either be very demanding – leading to an official



Figure 2 - Technical sketch

presentation – or simply a way to put down on paper the designers' ideas, for internal use. This phase may even be avoided, especially when an early virtual model presentation is planned.

After these sketches are completed, the Styling Centre board holds an internal meeting to choose the best proposals (usually about 3-5). Customers can be involved in the styling selection from the very beginning, thus giving their impressions and asking for modifications.

The phase of the technical sketches comes next. The number of alternatives is now limited and the package's requirements are more strictly respected. This is the first phase in which the feasibility of the project is studied, mediating between the quality of the project, its aesthetical aspect and its cost.

In this phase, depicted by figure 2, the perspective sketches are re-drawn in the four main views (front, rear, top and side) trying to keep the styling intentions while assuring the project feasibility. The drawing is still a sketch, focused on the rendering effect of the presented solution, but at the same time the technical details and the precision of the four-view representation are provided.

This process allows the development of the engineering of the vehicle already in the styling phase, so that the solutions presented on the technical sketches are usually nearly 80% feasible, with a considerable saving of time in the following phases.

The technical four-view drawing phase is extremely important and is the base for the following model construction. The four-view drawing can be seen as a bi-dimensional representation of a 3D object, whose views are projected on the walls of a box. Figure 3 represents the box "unfolded". This kind of approach will be later used in the virtual model creation phase.

4.1.3 Computer aided modelling

After the final four-view drawing is decided the phase of the mathematic creation of the surfaces starts. The four-views drawing passes from the hands of the designers to

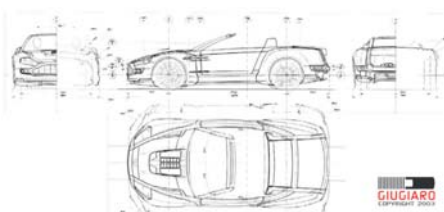


Figure 3 - technical four-view drawing

the Computer Aided Styling (CAS) engineers who create the virtual 3D model working on surfacing software tools on graphic workstations.

In the first phases of the process, it is not necessary to build a model complete with details, as headlamps or refinements. Instead, only the outer shape of the model is created, because the styling details are not fixed and it is most important to give a general idea of the styling intention. Figure 4 illustrates this phase, where most of the surfaces are already defined.

4.1.4 Virtual and physical model presentation

At this point the virtual model is ready to be presented to the customer at the Virtual Reality Centre, which is a room specifically equipped for this purpose.

At the end of the process, the physical 1:1 scale model is built. This physical mock-up is still considered fundamental to evaluate the styling impact of the new vehicle and to give the final validation to the project (this is a need felt by customers used to having the model evaluated by people that don't have the training needed to judge the styling effect in any other way).

After a few possible iterations, a styling master physical model is built with all the possible details (including handles, windscreen wipers, etc.) and painted to give a realistic look. This final model provides the final reference for the styling presentation and is also often used in the wind tunnel to evaluate the aerodynamics of the vehicle.

4.2 FIAT Centro Stile

In this section we describe the concept and design process, resulting from a set of interviews done at Fiat Centro Stile and Elasis.

Elasis industrial designers start considering the carry over components of other Fiat cars and have to perform the 'target setting' phase in order to define the right target for each performance (crash, handling, ride comfort, etc.). Next they develop the geometrical model of the system of components and verify the 'target achieving' phase.

In parallel with the FIAT styling designers of Centro Stile, they have to validate their project with the different style proposals. Designers start to sketch a number of proposals (2D drafts) either directly on a paper with pencil or markers, or with computer painting tools, as shown in Figure 5.

Then, the 2D sketches and the ergonomic scheme in the CAS system are imported in order to compare each style

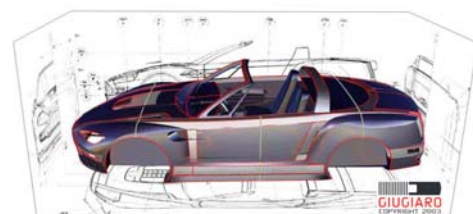


Figure 4 - Construction of surfaces



Figure 5 - Drafting phase

idea to the car package, made by Elasis industrial designers. After that, CAS operators create 3D curves and surfaces, according to the sketches for each style proposal. The best of these proposals are physically created in a 1:3 scale clay model.

The design process goes on per each model until just one proposal is chosen. This proposal will be developed in detail in the CAS system before a clay model on 1:1 scale is built. During this time, the information may be used also for aesthetic analysis (performed in a VR environment), geometrical and functional analysis (translated in CAD data) and style presentations. In parallel, the interior style process is developed.

5. VISITS VIDEO ANALYSIS

We made several visits to project partners' work installations with important outcomes to understand the processes of development and modelling. During these visits we video recorded users in action. This section describes briefly these experiences through the analysis of those videotapes.

As previously seen, the four-view drawing precedes the phase of curve and surface creation. This corresponds to a transition of information from the designers to the Computer Aided Styling engineers, who create the 3D models. Besides the inherent difficulties in communication (transmission of the right ideas) there are also dilatory aspects related to acquiring the technical drawings using scanners, and constructing the 3D box geometry with the 4-views on the walls. We will thus be focusing our video analysis on these tasks (the creation of perspective sketches and the computer aided modelling) because they are the main issues we want to tackle in the current modelling workflow.

The next subsections describe the analysis of the video from a visit to Italdesign Giugiaro's design facilities. The video documents a simulated modelling process where a car is digitised based on four hand drawn technical views. Next we discuss the modelling method used and comment the impact of the number of operations on the overall modelling process.

5.1 Modelling Process

After the four-view drawing is transferred to the CAD program, curves are the first entities to be drawn. The first lines drawn are the general shape of the upper part of the side view. The typical approach is from the general

to the specific, defining and isolating areas as work progresses.

Surface modelling follows the curve definition phase, and its modelling approach is the same. The number of control points is carefully kept to a minimum. A good balance between the surface quality and the surface complexity is fundamental and its achievement is based on the CAD engineer experience.

It often happens that handmade or even computer generated lines from the sketches and the four-view drawing are not clear enough to let the CAS engineer give a good interpretation of the designer's idea. Here, they have to co-operate discussing the modelling direction. The result is that the 3D mathematic model is the best compromise between what has been previously proposed in the sketches and what is realizable in a real 3D object.

5.2 Evaluation tools

A-class surfaces are referred to as being high-quality surfaces yet its exact definition changes depending on the customer. These desired high-quality surfaces' main requirements always consist on geometry, tangency and curvature parameters. To satisfy this desire for high standards, engineers use tools to inspect the shape and the parameters of the surfaces being modelled. Two examples are a tool displaying curvature radii variation details, and another allowing the visualization of how light reflects on the surface and how smoothly that light reflection varies across the surface.

5.3 Analysis

The video describes the modelling of 12 major curves of the car body and 3 surfaces formed by them. Despite the small number of entities modelled, the results prove to be interesting. We counted the following elementary control point (CP) movement operations: drag-and-drop actions on the control-points that change their position, and undo operations on these. Table 1 presents summary statistics for the curves and surfaces observed: minimum, maximum and average number of control points, as well as the number of movements made to control points per curve/surface. When a new control point is added to an existing curve, a point movement is always required¹ in order to place that point in the desired position.

The first surface to be modelled is the vastest of all surfaces, which covers the windshield and ceiling of the car. This surface has 6×4 control vertices and was actively retouched as a result of the high exigency (a total of 54 control point movements). The points were moved in conjunction with the use of both tools mentioned earlier. As a consequence, the order in which the points where moved was grouped with each row of control

¹ This movement is not taken into account on the numbers shown here, because it was considered to be part of the control-point's creation operation. Otherwise, the number of movements should be increased by $cp - 2$, where cp is the total number of control points on the curve.

points perpendicular to the car's orientation. This number is particularly excessive when taking into account that points belonging to an underlying curve are never retouched after the curve has been defined. This means the points moved during surface editing, are those at the interior of the surface. This implies that the 54 movements were actually performed on $6 \times 4 - 16 = 8$ points only, for that first surface, resulting on an average of nearly 7 modifications to the points' position made to each point moved. These figures allow a clear understanding of the importance of the small number of control points on the surfaces and therefore on the curves.

	Min.	Max	Avg. per ent.
Number of curves' CP	3	6	$46/12=3,83$
Curves' CP movements	0 ²	16	$55/12=4,58$
Number of surfaces' CP	15	24	$60/3=20,00$
Surfaces' CP movements	0	54	$54/3=21,33$

Table 1 - curve and surface modelling statistics

6. DISCUSSION

A problem detected during task analysis in an early modelling phase is the difficulty designers and modellers have in explaining details to one another. It would be desirable to have a tool that would allow designers to express their ideas quickly to arrive at a 3D prototype, so that modellers clearly understand the designer's intent. This will not replace the modeller's work since it is meant only to shorten the distance between the interpretations of the designer and CAD operator.

Single control point control over a surface is not the answer for the reduction of time and number of operations performed over a surface while modelling. A higher-level interaction scheme has to be adopted to eliminate the long iterative process of adjusting a surface's shape to the desired position.

7. CONCLUSIONS

A traditional CAD system cannot fulfil the needs of creative designers, since they don't like or understand control points, the mathematical foundation behind the control points and their relation with geometric shapes. Because of this, designers usually show themselves reluctant in using CAD systems to draw complex objects. They prefer to draw on 2D using shading and perspective techniques, in either paper or a paint program, to transmit their ideas and leave the modelling part to CAD operators.

Another drawback designers find on traditional CAD systems is the lack of ability to quickly sketch

prototypes. If this capability is present, they will easily validate the conceivability of their ideas. Two-dimensional drawings can suffer from perspective mismatches or optical illusions that prevent the surfaces from "making sense". If we could quickly create a prototype, impossible surface situations could be corrected immediately after, certifying that emotional drawing ideas correspond to the designer's intents.

Hence, designers, however displeased with the 3D modelling experience, would still like to be able to construct 3D models as a means of facilitating the interaction with the modeller. Thus, empowering a designer with the interaction with a modelling application should be carried out by means of a straightforward interface, like the familiar calligraphic interface.

8. ACKNOWLEDGEMENTS

This work was funded in part by the Portuguese Foundation for Science and Technology, project 34672/99, and the European Commission, project SmartSketches IST-2000-28169.

9. REFERENCES

- [Dias03] F. Marques Dias, A. Ferreira Jr., M. J. Fonseca, J. A. Jorge. Consolidated User Requirements and Task Analysis, May 2003.
<<http://smartsketches.inesc.pt/technicalreports/tr-03May8-af.pdf>>
- [Igarashi99] T. Igarashi, S. Matsuoka, H. Tanaka. Teddy: A sketching interface for 3D freeform design, *Proceeding of SIGGRAPH 99*, August 1999, 409-416.
- [Pereira00] J. P. Pereira, J. Jorge, V. Branco, F. Nunes Ferreira. Towards Calligraphic Interfaces: Sketching 3D Scenes with Gestures and Context Icons, WSCG2000, Plzen, Czech Republic, February 2000.
- [Terzopoulos94] D. Terzopoulos, H. Qin. Dynamic NURBS with Geometric Constraints for Interactive Sculpting, *ACM Transactions on Graphics*, 13(2), April 1994, 103-136.
- [SmartSk] SmartSketches project, available at <<http://smartsketches.inesc.pt>>
- [Sketchup3D] Sketchup3D, available at <<http://www.sketchup.com>>
- [Sutherland63] I.E. Sutherland. Sketchpad: a man-machine graphical communication system. *Proceeding of the Spring Joint Computer Conference*, May 1963, 329-346.
- [Zelevnik96] R. C. Zelevnik, K. P. Herndon, J. F. Hughes. SKETCH: An Interface for Sketching 3D Scenes. *Proceedings of SIGGRAPH 96*, 1996, 163-170.
- [Zhang01] M. Zhang, H. Qin. Hierarchical D-NURBS Surfaces and Their Physics-Based Sculpting. *Proceedings of International Conference on Shape Modelling and Applications (SMI2001)*, May 2001.

² 0 corresponds to a curve that was copied from another.

Polygon Detection from a Set of Lines

Alfredo Ferreira Jr. Manuel J. Fonseca Joaquim A. Jorge

Department of Information Systems and Computer Science

INESC-ID/IST/Technical University of Lisbon

R. Alves Redol, 9, 1000-029 Lisboa, Portugal

alfredo.ferreira.jr@inesc-id.pt, mjf@inesc-id.pt, jorgej@acm.org

Abstract

Detecting polygons defined by a set of line segments in a plane is an important step in the analysis of vectorial drawings. This paper presents an approach that combines several algorithms to detect basic polygons from a set of arbitrary line segments. The resulting algorithm runs in polynomial time and space, with complexities of $O((N + M)^4)$ and $O((N + M)^2)$ respectively, where N is the number of line segments and M is the number of intersections between line segments. Our choice of algorithms was made to strike a good compromise between efficiency and ease of implementation. The result is a simple and efficient solution to detect polygons from lines.

Keywords

Polygon Detection, Segment Intersection, Minimum Cycle Basis

1. INTRODUCTION

Unlike image processing, where data consist of raster images, the proposed algorithm deals with drawings in vector format, consisting of line segments. This requires completely different approaches, such as described in this paper.

To perform polygon detection from a set of line segments we divide this task in four major steps. First we detect line segment intersections using the Bentley-Ottmann algorithm [13]. Next step creates a graph induced by the drawing, where vertices represent endpoints or proper intersection points of line segments and edges represent maximal relatively open subsegments that contain no vertices. The third step finds the Minimum Cycle Basis (MCB) [16] of the graph induced in previous step, using the algorithm proposed by Horton [12]. Last step constructs a set of polygons based on cycles in the previously found MCB. This is straight-forward if we transform each cycle into a polygon, where each vertex in the cycle represents a vertex in the polygon and each edge in the cycle represents an edge in the polygon.

In sections 2 and 3 we describe the four steps of our method. Section 4 presents the whole algorithm, followed by experimental results in section 5. Finally in section 6 we discuss conclusions and future work.

2. INTERSECTION REMOVAL

In a vector drawing composed by a set of line segments there might exist many intersections between these segments. To detect polygonal shapes we have to remove proper segment intersections, thus creating a new set of

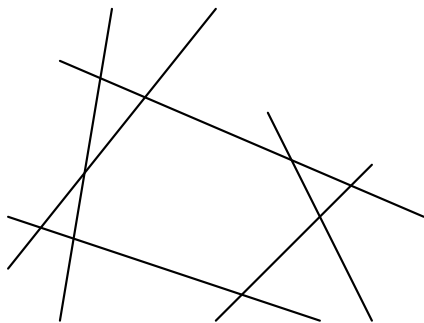
line segments in which any pair of segments share at most one endpoint.

2.1. Finding line segment intersections

The first step of our approach consists in detecting all M intersections between N line segments in a plane. This is considered one of the fundamental problems of Computational Geometry and it is known that any algorithm, within the model of algebraic decision tree, have a lower bound of $\Omega(N \log N + M)$ time to solve it [3, 5].

In [1] Balaban proposes two algorithms for finding intersecting segments, a deterministic and asymptotically optimal for both time $O(N \log N + M)$ and space $O(N)$ algorithm and a simpler one that can perform the same task in $O(N \log^2 N + M)$ -time. Before that, Chazelle and Edelsbrunner [5] reached a time optimal algorithm $O(N \log N + M)$ with space requirement of $O(N + M)$. The randomized approach devised by Clarkson and Shor [6] produced a algorithm for reporting all intersecting pairs that requires $O(N \log N + M)$ time and $O(N)$ space.

In 1979 Bentley and Ottmann proposed an algorithm that solved this problem in $O((N+M) \log N)$ time and $O(N+M)$ space [13]. This algorithm is the well-known Bentley-Ottmann algorithm and after more than 20 years it is still widely adopted in practical implementations because it is easy to understand and implement [15, 11]. In realizing that this is not the most complex part of our approach, we decide to use the Bentley-Ottmann algorithm, since its complexity is pretty acceptable for our purposes and its published implementations are quite simple.

Figure 1. Set Φ of line segments

2.2. Removing line segment intersections

The next step of our approach is to remove all proper intersections between line segments, dividing each intersected segment in sub-segments without proper intersections, only sharing endpoints. In order to find and remove intersections, performing at once the first two steps of our approach, we use a robust and efficient implementation of the Bentley-Ottmann algorithm, described by Bartuschka, Mehlhorn and Naher [2] that computes the planar graph induced by a set of line segments. Their implementation, represented in this paper by COMPUTE-INDUCED-GRAPH, computes the graph G induced by set Φ in $O((N+M) \log N)$ time. Since this algorithm is quite long we choose not to present it here. We refer our readers to [2] for a detailed description.

In this implementation the vertices of G represent all endpoints and proper intersection points of line segments in Φ , and the edges of G are the maximal relatively open sub-segments of lines in Φ that do not contain any vertex of G . The major drawback of this implementation lies in that parallel edges are produced in the graph for overlapping segments. We assume that Φ contains no such segments. Considering, for example, the set Φ shown in Figure 1, COMPUTE-INDUCED-GRAPH will produce the graph G , depicted in Figure 2, where each edge represents a non-intersecting line segment.

3. POLYGON DETECTION

Detecting polygons is similar to finding cycles on the graph G produced in the previous step.

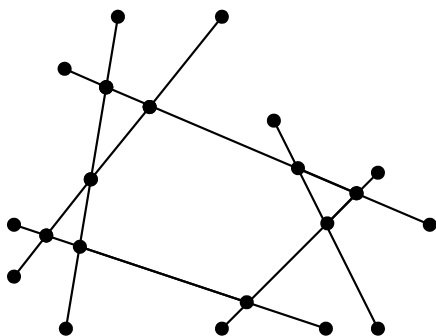
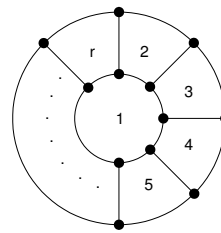
Figure 2. Graph G induced by Φ 

Figure 3. A planar graph with a exponential number of cycles

3.1. All Cycles of a Graph

The first known linear-time algorithm for listing all cycles of a graph was presented by Syslo [16]. This algorithm requires $O(V)$ space and $O(V \times C)$ time, where V is the number of vertices and C the number of cycles in G . Later Dogrusöz and Krishnamoorthy proposed a vector space algorithm for enumerating all cycles of a planar graph that runs in $O(V^2 \times C)$ time and $O(V)$ space [8]. Although asymptotically slower, this algorithm is much simpler than Syslo's and is amenable to parallelization. Unfortunately, the total number of cycles in a planar graph can grow exponentially with the number of vertices [14]. An example of this situation is the graph presented in Figure 3. In this case, the number of cycles, including the interior region numbered 1, is $O(2^r)$ with $r = k/2 + 1$, where k is the number of vertices, since one can choose any combination of the remaining regions to define a cycle [8]. This is why it is not very feasible to detect all polygons that can be constructed from a set of lines. In this paper, we choose just to detect the minimal polygons, those that have a minimal number of edges and cannot be constructed by joining other minimal polygons.

3.2. Minimum Cycle Basis of a Graph

Considering that we just want to detect the minimal polygons this can be treated as searching for a Minimum Cycle Basis (MCB). So, the second step of our approach consists in obtaining a MCB of graph G . A cycle basis is defined as a basis for the cycle space of G which consists entirely of elementary cycles. A cycle is called elementary if it contains no vertex more than once. The dimension of the cycle space is given by the *cyclomatic number* $\nu = E - V + P$ [9, 4], where E is the number of edges and V the number of vertices in G and P is the number of connected components of G .

Horton presented the first known polynomial-time algorithm to find the shortest cycle basis of a graph, which runs in $O(E^3V)$ time [12] or in $O(E^4)$ on simple planar graphs [10], which is the case. While asymptotically better solutions have been published in the literature, the Bentley-Ottmann algorithm is both simple and usable for our needs. The pseudo-code of this algorithm is listed in MINIMUM-CYCLE-BASIS and shortly described below. A further detailed description of this algorithm and concepts behind it can be found in [12].

The ALL-PAIRS-SHORTEST-PATHS finds the shortest

MINIMUM-CYCLE-BASIS(G)

```

1   $\Gamma \leftarrow \text{empty set}$ 
2   $\Pi \leftarrow \text{ALL-PAIRS-SHORTEST-PATHS}(G)$ 
3  for each  $v$  in  $\text{VERTICES}(G)$ 
4  do for each  $(x, y)$  in  $\text{EDGES}(G)$ 
5      do if  $\Pi_{x,v} \cap \Pi_{v,y} = \{v\}$ 
6          then  $C \leftarrow \Pi_{x,v} \cup \Pi_{v,y} \cup (x, y)$ 
7              add  $C$  to  $\Gamma$ 
8   $\text{ORDER-BY-LENGTH}(\Gamma)$ 
9  return  $\text{SELECT-CYCLES}(\Gamma)$ 

```

paths between all pairs of vertices in graph G and can be performed in $O(V^3)$ time and $O(V^2)$ space using Floyd-Warshall or Dijkstra algorithms [7]. ORDER-BY-LENGTH orders the cycles by ascending length and can be implemented by any efficient sorting algorithm. This is a non-critical step because it has a $O(V \log V)$ upper bound in time complexity, which is insignificant in comparison with other steps of this algorithm.

In SELECT-CYCLES we use a greedy algorithm to find the MCB from Γ set of cycles. To do this Horton [12] suggests representing the cycles as rows of a 0-1 incidence matrix, in which columns correspond to the edges of the graph and rows are the incidence vectors of each cycle. Gaussian elimination using elementary row operations over the integers modulo two can then be applied to the incidence matrix, processing each row in turn, in ascending order of the weights of cycles, until enough independent cycles are found.

This step dominates the time complexity from other steps, since it takes $O(E\nu^2V)$ time. Knowing that G is always a simple planar graph we can conclude that as a whole the MINIMUM-CYCLE-BASIS algorithm has a worst case upper bound of $O(E\nu^2V) = O(E^3V) = O(E^4)$ operations and a space requirements of $O(V^2)$.

Figure 4 shows an example of Γ , the set of cycles resulting from applying the MINIMUM-CYCLE-BASIS to graph G shown in Figure 2.

3.3. Polygon construction

The last step of our approach consists in constructing a set Θ of polygons from the MCB. An algorithm to perform this operation can easily run in $O(CV)$ time, where C is number of cycles in MCB. Such an algorithm is listed in POLYGONS-FROM-CYCLES which returns a set Θ of polygons.

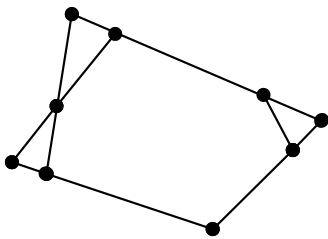


Figure 4. Shortest cycle basis Γ of graph G

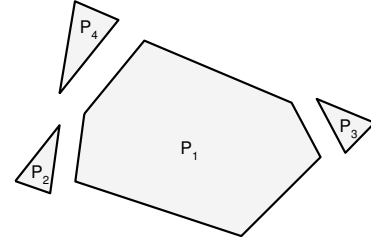


Figure 5. Set Θ of polygons detected from Φ

POLYGONS-FROM-CYCLES(Γ)

```

1   $\Theta \leftarrow \text{empty set}$ 
2  for each  $C$  in  $\Gamma$ 
3      do  $P \leftarrow \text{new polygon}$ 
4          for each  $v$  in  $\text{VERTICES}(V)$ 
5              do add vertex  $v$  to  $P$ 
6              add polygon  $P$  to  $\Theta$ 
7  return  $\Theta$ 

```

Figure 5 illustrates the resulting set Θ of polygons generated by applying POLYGONS-FROM-CYCLES to Γ depicted in Figure 4.

4. ALGORITHM OUTLINE

We can now outline DETECT-POLYGONS. This algorithm is able to detect a set Θ of polygons from a initial set Ψ of line segments. To perform this task we pipeline the algorithms referred in previous sections for line segment intersection removal, MCB finding and cycle-to-polygon conversion.

DETECT-POLYGONS(Ψ)

```

1   $G \leftarrow \text{COMPUTE-INDUCED-GRAPH}(\Psi)$ 
2   $\Gamma \leftarrow \text{MINIMUM-CYCLE-BASIS}(G)$ 
3   $\Theta \leftarrow \text{POLYGONS-FROM-CYCLES}(\Gamma)$ 
4  return  $\Theta$ 

```

As referred in section 2.2, COMPUTE-INDUCED-GRAPH runs in $O((N + M) \log N)$ time and $O(N + M)$ space. The SHORTEST-CYCLE-BASIS runs in $O(V^4)$ operations and has a space requirement of $O(V^2)$, making this the critical step in the complexity of this algorithm, since the POLYGONS-FROM-CYCLES just needs $O(CV)$ time.

Since the number V of vertices in the graph is no greater than the sum of line endpoints ($2 \times N$) with detected intersections M , we can then conclude that the proposed algorithm has time and space complexities of $O(V^4) = O((N + M)^4)$ and $O(V^2) = O((N + M)^2)$, respectively.

5. EXPERIMENTAL RESULTS

The algorithm proposed in this paper was implemented in C++ and tested in a Intel Pentium III 1GHz 512MB RAM computer running Windows XP. We tested the algorithm with sets of line segments created from simple test drawings, technical drawings of mechanical parts and hand-sketched drawings. Table 1 presents the results obtained from these tests.

Lines	Intersections	Nodes	Edges	Time (ms)
6	9	21	24	10
36	16	58	68	50
167	9	169	177	3986
286	47	389	376	8623
518	85	697	679	36703
872	94	1066	10050	128995
2507	10	2407	2526	1333547

Table 1. Results of algorithm tests

Based on these results we conclude that performance is acceptable for on-line processing in sets with less than three-hundred lines like hand-sketches or small-size technical drawings. If the line set have about 2500 lines the algorithm will take more than twenty minutes to detect the polygons. Still this remains a feasible solution for batch processing of medium-size technical drawings.

6. CONCLUSIONS and FUTURE WORK

The proposed algorithm is used for polygon detection in vector drawings to create descriptions based on spatial and topological relationships between polygons. Other use is detecting planar shapes in sketches. Both applications have been implemented as working prototypes used for shape retrieval and architectural drawing from sketches.

The algorithm presented here detects in polynomial time and space, all minimal polygons that can be constructed from a set of line segments. This approach uses well-known and simple to implement algorithms to perform line segment intersection detection and to find a MCB of a graph, instead of using more efficient but less simpler methods.

Indeed there is considerable room for improvement in the presented algorithm, namely through the use of more recent, complex and efficient algorithms. Further work may be carried out regarding the detection and correction of rounding errors resulting from finite precision computations.

7. ACKNOWLEDGMENTS

We thank to Professor Mukkai S. Krishnamoorthy from Rensselaer Polytechnic Institute, New York, for his very helpful suggestions.

This work was funded in part by the Portuguese Foundation for Science and Technology, project 34672/99 and the European Commission, project SmartSketches IST-2000-28169.

References

- [1] Ivan J. Balaban. An optimal algorithm for finding segment intersections. In *Proceedings of the 11th Annual ACM Symposium Comp. Graph.*, pages 211–219. ACM, 1995.
- [2] Ulrike Bartuschka, Kurt Mehlhorn, and Stefan Naher. A robust and efficient implementation of a sweep line algorithm for the straight line segment intersection. In *Proceedings of Workshop on Algorithm Engineering*, pages 124–135, Venice, Italy, September 1997.
- [3] M. Ben-Or. Lower bounds for algebraic computation trees. In *Proceedings of the 15th Annual ACM Symposium Theory of Computing*, pages 80–86. ACM, 1983.
- [4] Augustin-Louis Cauchy. Recherche sur les polyèdres. *J. Ecole Polytechnique*, 9(16):68–86, 1813.
- [5] Bernard Chazelle and Herbert Edelsbrunner. An optimal algorithm for intersecting line segments in the plane. *Journal of the ACM*, 39:1–54, 1992.
- [6] Ken Clarkson and Peter W. Shor. Applications of random sampling in computational geometry, ii. *Discrete and Computational Geometry*, 4:387–421, 1989.
- [7] Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. MIT Press, McGraw-Hill, 2nd. edition, 1990.
- [8] U. Dogrusöz and M. Krishnamoorthy. Cycle vector space algorithms for enumerating all cycles of a planar graph. Technical Report 5, Rensselaer Polytechnic Institute, Dept. of Computer Science, Troy, New York 12180 USA, January 1995.
- [9] Leonhard Euler. Elementa doctrinae solidorum. *Novi Commentarii Academiae Scientiarum Petropolitanae*, 4:109–140, 1752.
- [10] David Hartvigsen and Russel Mardon. The all-pairs minimum cut problem and the minimum cycle basis problem on planar graphs. *SIAM Journal on Computing*, 7(3):403–418, August 1994.
- [11] John Hobby. Practical segment intersection with finite precision output. *Computational Geometry: Theory and Applications*, 13(4), 1999.
- [12] J.D.Horton. A polynomial-time algorithm to find the shortest cycle basis of a graph. *SIAM Journal on Computing*, 16(2):358–366, April 1987.
- [13] J.L.Bentley and T.Ottmann. Algorithms for reporting and counting geometric intersections. *IEEE Transactions on Computers*, pages 643–647, 1979.
- [14] Prabhaker Mateti and Narsingh Deo. On algorithms for enumerating all circuits of a graph. *SIAM Journal on Computing*, 5(1):90–99, March 1976.
- [15] Joseph O'Rourke. *Computational Geometry in C*, chapter Section 7.7 "Intersection of Segments", pages 264–266. Cambridge University Press, 2nd edition, 1998.
- [16] Maciej M. Syslo. An efficient cycle vector space algorithm for listing all cycles of a planar graph. *SIAM Journal on Computing*, 10(4):797–808, November 1981.

Corte de Peças de Pedra:

Uma Aplicação da Computação Gráfica

Samuel Silva
Univ. de Aveiro
Aveiro
a19775@alunos.det.ua.pt

Filipe Marques
Univ. de Aveiro
Aveiro
a20094@alunos.det.ua.pt

Manuel Raposo
PROSISTAV
Aveiro
mraposo@oninet.pt

Joaquim Madeira
Univ. de Aveiro — DET/IEETA
Aveiro
jmadeira@det.ua.pt

Beatriz Sousa Santos
Univ. de Aveiro — DET/IEETA
Aveiro
bss@det.ua.pt

Sumário

A indústria de transformação de blocos de pedra em peças decorativas tem evoluído rapidamente, sendo estas realizadas com recurso a máquinas que dispõem de autómatos programáveis, comandos numéricos e aplicações com interfaces de utilizador, que vão desde consolas de operação com duas linhas de texto a computadores industriais com interfaces gráficas. Embora a indústria portuguesa tenha evoluído nas soluções de controlo, melhores interfaces com o utilizador permitiriam uma preparação de trabalho mais rápida e menos erros de programação dos autómatos. Neste artigo descrevem-se os aspectos mais relevantes de uma destas aplicações, que recebeu melhoramentos a vários níveis, no sentido de uma maior usabilidade. Esta aplicação foi desenvolvida em Visual Basic, recorrendo à biblioteca gráfica OpenGL.

Palavras-chave

Corte de pedra, modelação tridimensional, visualização, OpenGL

1. INTRODUÇÃO

A indústria de transformação de blocos de pedra (mármore, granitos e outras) em peças decorativas como lareiras, balaústres ou colunas, tem evoluído de forma rápida, sendo as tarefas manuais substituídas por máquinas mais ou menos pesadas consoante as dimensões das peças a trabalhar. Apesar do meio agressivo em que têm que trabalhar, essas máquinas já não dispõem de autómatos programáveis, comandos numéricos e interfaces com o operador que vão desde consolas de operação com duas linhas de texto a computadores industriais com interfaces e aplicações gráficas sofisticadas.

A indústria portuguesa produz algumas centenas de máquinas por ano, estando também a evoluir nas soluções de controlo, mas peca ainda pela utilização de interfaces com o utilizador pouco atraentes e incompletas em termos da informação disponibilizada. A melhoria destas interfaces poderia permitir uma preparação de trabalho mais rápida, menos erros de programação dos autómatos e uma maior usabilidade.

O funcionamento deste tipo de máquinas é, em termos mecânicos, semelhante ao de um *plotter* de grandes dimensões com um disco diamantado (a “caneta”), um eixo vertical que controla a profundidade de corte e dois

eixos horizontais que controlam a posição do disco sobre o bloco de pedra.

As aplicações existentes nesta área são aplicações de controlo para máquinas do tipo torno (para colunas, balaústres ou esferas) e máquinas de ponte (para peças que não sejam de revolução) com 2 a 8 eixos de posicionamento, e têm como base do sistema de controlo um autómato industrial programável (PLC) e um computador industrial de painel com ecrã tátil (PC com *touch-screen*).

O PLC é programado numa linguagem própria e o PC usa habitualmente o sistema operativo *Windows*, uma aplicação de comunicação com o PLC e uma aplicação em *Visual Basic*, através da qual o operador comunica com a máquina, introduzindo as informações necessárias para a produção e efectuando a verificação do estado do processo depois de iniciado.

Quando se pretende que a máquina execute um trabalho cujo resultado seja uma peça com uma determinada forma, essa forma é definida por uma sequência de segmentos de recta e de arcos, compondo o perfil da peça final. Estes perfis ficam armazenados no PC de cada máquina, podendo o operador reutilizá-los mais tarde.

O trabalho apresentado consistiu no desenvolvimento de uma aplicação com as características descritas acima. Para tal, recorreu-se à linguagem de programação *Visual Basic* e à biblioteca gráfica *OpenGL* para modelação e visualização bidimensionais e tridimensionais. Esta biblioteca apresenta duas grandes vantagens: por um lado é independente do *hardware*, pelo que pode ser usada em diferentes plataformas e, por outro, é *freeware* podendo ser usada sem o pagamento de licenças [Woo94]. A sua utilização a partir do *Visual Basic* é feita recorrendo a uma *type library* [TLB02].

Relativamente à aplicação anterior foram introduzidos melhoramentos, não só através da alteração de procedimentos e de diálogo com o utilizador, no sentido de melhorar a usabilidade [Mayhew92] [Dix98], mas também através da integração de funcionalidades completamente novas como:

- ferramentas auxiliares de desenho adicionais (como *zoom* e *pan*),
- representação tridimensional das peças,
- simulação de corte com compensação da ferramenta,
- impressão parametrizável,
- ajuda ao utilizador.

O desenvolvimento desta nova aplicação foi limitado pelas características inerentes à plataforma usada (PC's não muito evoluídos e *touch-screen*) e pelo modelo conceptual da interface de utilizador da aplicação anterior, com a qual os utilizadores estão muito familiarizados.

2. DESCRIÇÃO GERAL DA APLICAÇÃO

Nesta secção é descrita a funcionalidade da aplicação, realçando-se os aspectos considerados mais relevantes.

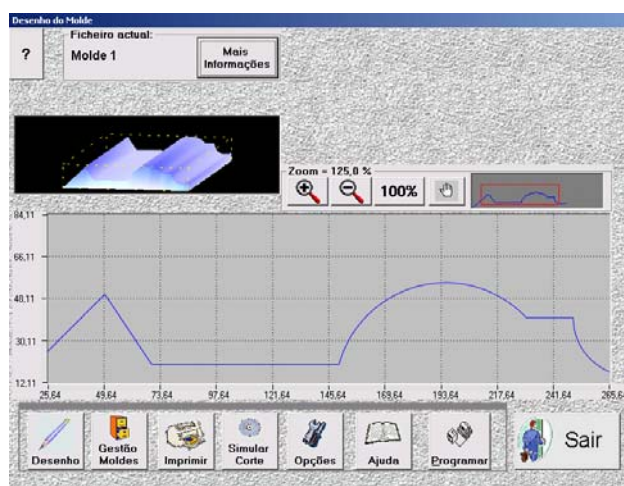


Figura 1 - Aspecto geral da interface de utilizador

2.1 Aspecto geral da interface de utilizador

O aspecto geral da interface de utilizador correspondente à introdução de um perfil é mostrado na figura 1. Como

se pode observar, existem diversas áreas onde é apresentada informação e oferecida funcionalidade ao utilizador: a área de desenho do perfil, os botões de *zoom* e de *pan*, a área de representação tridimensional da peça, a área do menu principal, a área de informação sobre a peça e a ajuda.

2.2 Desenho do perfil

É fornecida ao utilizador uma representação do perfil que está actualmente a desenhar. A zona de desenho é rectangular e está dimensionada por valores de escala. A unidade de medida é o *cm*. Sob o desenho do perfil está uma grelha para facilitar a visualização.

Os segmentos que constituem um perfil são de dois tipos: segmentos de recta ou arcos de circunferência. A opção por apenas estes dois tipos de elementos gráficos foi feita para simplificar o processo de desenho e a compreensão por parte dos utilizadores. Cada segmento é definido estabelecendo inicialmente as coordenadas dos seus pontos inicial e final, e seleccionando o tipo de segmento pretendido. No caso da definição de arcos, é possível especificar, também, a cota máxima (ou mínima) que o arco atinge.

É também possível realizar alterações em segmentos previamente introduzidos, como, por exemplo, a alteração do raio de um arco, o estabelecimento de concordância entre um arco e o segmento que o precede, a alteração de arcos mantendo a sua cota máxima (ou mínima) e a alteração das coordenadas de início ou fim de um segmento (recta ou arco). O segmento no qual se estão a realizar alterações é representado a cor diferente (opcionalmente também com outra espessura de linha) dos restantes.

Foi criada também uma ferramenta de ampliação/redução (*zoom*) em torno de um ponto do perfil. A selecção do ponto em torno do qual se efectuará o *zoom* é feita com o auxílio de "*gravidade*": se o ponto seleccionado estiver na vizinhança de um vértice do perfil, a ampliação é realizada em torno desse vértice e não do ponto seleccionado. Para auxiliar o operador, é feita também a representação, na barra de *zoom*, da zona de perfil que está actualmente a ser ampliada.

A janela de ampliação pode ser também deslocada sobre o perfil (*pan*), tendo para tal que se seleccionar um qualquer ponto da área de desenho e deslocar o perfil na direcção desejada.

É ainda fornecida a possibilidade de inverter um perfil na horizontal (o que estava do lado esquerdo passa para o lado direito e vice-versa) e reflectir um perfil em torno de um eixo vertical, para produzir uma peça simétrica a partir de meio perfil. Estas são operações habitualmente efectuadas pelos operadores.

Para permitir uma recuperação rápida de erros, característica muito desejável numa interface de utilizador [Mayhew92], é possível anular a última alteração efectuada no perfil.

2.3 Representação Tridimensional da Peça

Para proporcionar ao operador uma percepção do resultado final do trabalho, foi desenvolvida a possibilidade de visualizar tridimensionalmente a peça correspondente ao perfil que está actualmente a introduzir, usando dois tipos de visualização, com distintos níveis de detalhe.

2.3.1 Pré-visualização tridimensional da peça

Esta representação tridimensional de baixa resolução e pequeno tamanho do modelo da peça está localizada logo acima da representação do perfil (figura 1). Tem como finalidade fornecer ao operador uma ideia das linhas principais da peça à medida que vai introduzindo os segmentos.

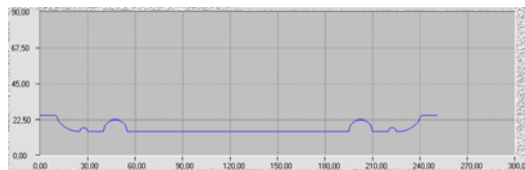


Figura 2 - Exemplo de um perfil traçado por um operador

Consideremos o perfil mostrado na figura 2; a partir deste perfil podem obter-se três peças diferentes (mostradas na figura 3) de acordo com a operação sobre ele efectuada: extrusão (a), revolução (b) e modelação de coluna com quatro faces (c).

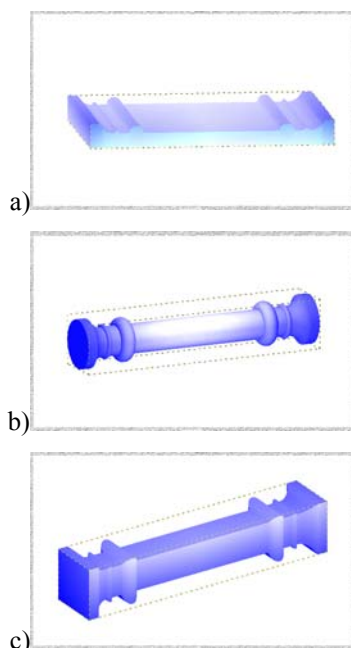


Figura 3 - Peças obtidas a partir do perfil da figura 2: a) peça de extrusão, b) peça de revolução, c) coluna com quatro faces

2.3.2 Modelo tridimensional da peça em pormenor

O operador tem à sua disposição (numa *frame* própria) uma visualização tridimensional da peça com elevada resolução e tamanho.

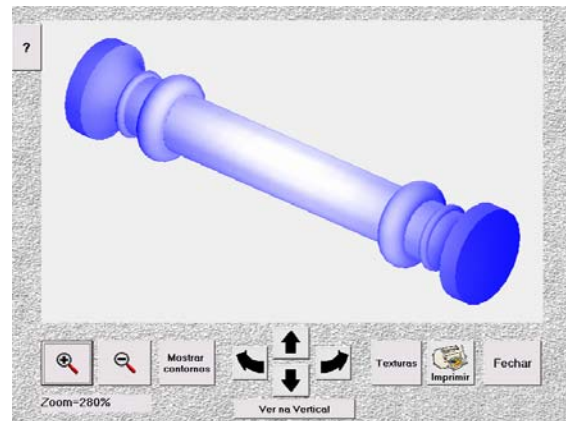


Figura 4 - Visualização tridimensional da peça

São fornecidas ferramentas de ampliação e redução. Para além disso, o operador pode seleccionar a visualização da peça com ou sem os contornos dos diversos segmentos, e na vertical ou na horizontal. Pode movimentar a peça recorrendo para tal a cursores ou seleccionando um qualquer ponto na área de representação e deslocando a peça na direcção desejada (figura 4).

Finalmente, pode aplicar texturas às peças para lhes conferir um aspecto mais realista, a partir de qualquer imagem que se associe à aplicação.

2.4 Simulação de Corte

A simulação de corte fornece ao utilizador a previsão do resultado do corte do molde pelo automático. Os parâmetros da simulação são a espessura do disco de corte e o avanço entre cortes. Através da análise das simulações de corte, o operador pode inferir quais os parâmetros que permitirão produzir a peça com o melhor compromisso entre qualidade e tempo de execução.

2.4.1 Processo de corte

O automático recebe do PC um perfil e executa cortes a intervalos regulares programáveis, posicionando o canto esquerdo do disco, em cada ponto de corte, à cota do perfil nesse ponto.

2.4.2 Compensação da ferramenta

Se o perfil fosse enviado para o automático directamente, sem qualquer tipo de processamento, o resultado do corte seria incorrecto, devido ao processo de corte acima descrito. Em zonas em que a cota fosse crescente ao longo do perfil, o automático iria cortar pedra em áreas onde não seria suposto fazê-lo. A figura 5 ilustra esse efeito.

Para resolver este problema, em vez do perfil original, é enviado para o automático um perfil compensado, que garante que em nenhuma situação é cortada pedra abaixo da linha do perfil original. O perfil compensado é calculado a partir do perfil original e de uma dada espessura do disco de corte.

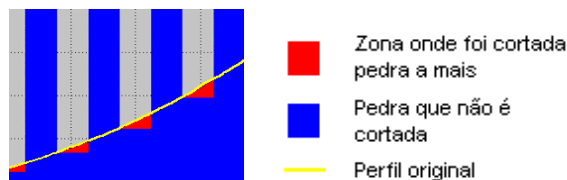


Figura 5- Efeito do corte, na pedra, sem compensação de ferramenta

Este processo de correcção tem sido executado manualmente exigindo, para tal, uma grande experiência dos operadores da máquina na fase de desenho. Esta correcção automática permitirá aliviar os operadores dessa tarefa, garantindo resultados independentes do operador.

2.4.3 Simulação

A partir dos parâmetros de corte (espessura de disco e avanço entre cortes) é apresentada ao utilizador uma simulação do corte do perfil num bloco de pedra. Na figura 6 é apresentado um exemplo de uma simulação de corte.

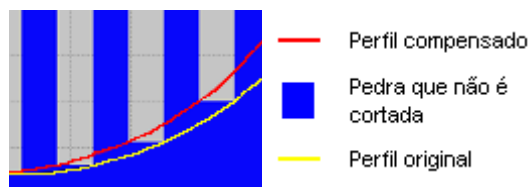


Figura 6- Simulação em que o corte foi executado a partir do perfil compensado

Em algumas máquinas, como é o caso dos tornos, a pedra pode sofrer uma operação final de rectificação para eliminar os “degraus” deixados pelo processo de corte.

2.5 Impressão

O utilizador tem a possibilidade de imprimir informação vária relacionada com o modelo da peça:

- perfil (com ou sem ampliação),
- simulação de corte (com ou sem ampliação),
- representação tridimensional (com ou sem ampliação, com ou sem textura aplicada),
- tabela de pontos (lista dos segmentos que compõem o perfil).

A introdução desta funcionalidade foi motivada pelo facto de em ambiente industrial a informação circular em papel e, também, para funcionar como processo de arquivo, permitindo visualizar uma peça que se procure e reintroduzir um perfil, caso necessário.

2.6 Biblioteca de moldes

O utilizador pode gravar moldes numa biblioteca de moldes e mais tarde proceder à sua reutilização. Cada molde é composto pelo perfil da peça e vária informação adicional.

Para seleccionar um dado molde da biblioteca é necessário percorrer a lista de moldes. À medida que se vai percorrendo esta lista, vai sendo mostrada a informação relativa ao molde que está actualmente seleccionado na lista. Essa informação consiste na representação do perfil, na pré-visualização do modelo tridimensional da peça e de informação adicional sobre o molde, como, por exemplo, o cliente a que se destina e a data de realização.

2.7 Ajuda ao utilizador

Foi desenvolvida uma ajuda ao utilizador, procurando-se que esta fosse tanto quanto possível facilmente acessível, precisa e completa, consistente, flexível e que obstruísse tão pouco quanto possível o trabalho do utilizador [Dix98].

Este desenvolvimento foi limitado pelo facto de o único dispositivo de entrada utilizado na plataforma ser o *touch-screen*, o que à partida exclui a utilização de tipos de ajuda muito usados em plataformas do tipo *desktop*.

Existem na interface duas formas de ajuda ao utilizador:

- Um manual *online* (também disponível em papel), onde o utilizador pode encontrar uma descrição da funcionalidade da aplicação e explicações acerca da sua utilização.
- Ajuda contextualizada, isto é, informações sucintas acerca dos elementos que compõem a interface.

3. CONCLUSÕES

A aplicação desenvolvida é uma inovação na indústria portuguesa de transformação de blocos de pedra, esperando-se que possibilite a realização de peças decorativas de uma forma mais rápida e eficaz, o que se traduzirá, para as empresas, em ganhos de produtividade.

A aplicação desenvolvida está a ser testada pelos utilizadores alvo e por técnicos que instalam a aplicação e a explicam a estes utilizadores, por forma a que sejam identificados aspectos passíveis de serem melhorados, quer ao nível de nova funcionalidade a ser incluída quer ao nível de problemas de usabilidade [Nielsen93].

4. REFERÊNCIAS

- [Dix98] Dix, A., J. Finley, G. Abowd, B. Russell, *Human Computer Interaction*, 2nd. Ed., Prentice Hall, 1998
- [Mayhew92] Mayhew, D., *Principles and Guidelines in Software User Interface Design*, Prentice Hall, 1992
- [Nielsen93] Nielsen, J., *Usability Engineering*, Academic Press, 1993
- [TLB02] *VBOpenGL type library v. 1.2*, October 2002 <http://is6.pacific.net.hk/~edx/tlb.htm> (Junho 2003)
- [Woo94] Woo M., J. Neider, T. Davis, *The Official Guide to Learning OpenGL*, Release 1 (1994), Addison-Wesley Publishing Company

GIDes++

Joaquim A Jorge
Dep. Eng^a. Informática, IST
Av. Rovisco Pais, 1000 Lisboa
jorgej@acm.org

Nelson F Silva
Instituto Superior Técnico
Lisboa
nelson.faria@netcabo.pt

Tiago D Cardoso
Instituto Superior Técnico
Lisboa
tiagocardoso@netcabo.pt

Abstract

GIDes ++ is a gesture based calligraphic 3D modeller using expectation lists, and extends the notion of natural and efficient interface. A brand new look and a new modeller kernel empower the modelling capabilities. A measures layer enables the user to visualize and change measures, giving a new dimension in precise and accurate modelling. Adding new modalities like handwriting and speech, GIDes ++ present a natural and intelligent multimodal interface enabling quick models prototyping creation. Visually, much effort is being made to create a more appealing and elegant application, but keeping always in mind the usefulness of these visual effects.

Keywords

Interaction Techniques, 3D Modeling, Sketching, Gesture Interfaces, Calligraphic Interfaces, Handwriting Recognition, Voice Recognition.

1. INTRODUCTION

Intelligent Multimodal Interfaces have come a long way and are currently being applied with success in many areas. Providing an interface that enables a user to interact with a system in a more natural way is one of the main goals of this upcoming research subject.

Looking at the usual workflow of any manufactured piece one can see that most of the creative work is done by designers, the following stage consists of having a specialized CAD technician to take the product of this conception phase, which is usually drawn in freehand, and port it to an exact computer three-dee model.

By providing a more natural and pleasant, interface instead of the traditional WIMP ones found in commercial packages, we hope to close this gap and “build” a bridge between these two stages, motivating designers to draw free-hand directly on a CAD system, allowing them to quickly prototype models that can easily be changed to follow the so usually strict requirement measures and constraints.

2. BACKGROUND WORK

Using sketching as the main organizing paradigm was the aim of a CAD system called GIDes [Pereira00], which called this approach *Calligraphic Interfaces*.

Replacing direct manipulating by sketching alone posed several very interesting challenges as well as requiring user to learn a given command set in order to draw shapes.

On the most noticeable solutions that resulted from the development of this system were *Expectation Lists*.

These, as shown in the right figure, allowed the system to deal with the ambiguity of the input provided by user's

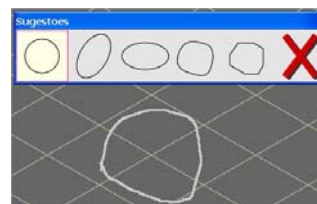


Figure 1: Expectation Lists

structures.

2.1 Manipulating Tools

In order to accomplish Three-Dee edition a Cutting tool was presented. With this tool one could carry out free-hand cuts on faces. There were two kinds of cuts: inside cuts, where the user could draw a close line in a surface, removing the enclosing area; the other type was edge to edge cuts, which were made with a line from an edge to another, eliminating the outside portion of the object. Both operations cut objects through the inverse-normal vector given at the point of the surface where the stroke began.

Other important tools only modified the object position in space, not changing the object at all. With *Positioning* the user picked a point in an object and one in an aim object and the first of these will be translated so that both chosen points coincide. On the other hand in the *Glue tool* the initial object would be glued to the other one. To accomplish this restriction, the initial object would be rotated as well as translated so that the picked face would have the inverse normal vector of the destiny face. If two vertexes were chosen, another restriction would be triggered and an additional rotation would be performed so that both edges leaving those vertexes coincided.

Finally, with the *Adjust tool*, not only one could use it as the Positioning tool, but if the initial object was glued to a face and the destination point was also on that face, the object will translate, but respecting that face's plane, allowing both faces to continue to overlap and respecting the glue constrain.

3. GIDES ++

When recreating GIDes, it was decided to change the strictly imperative language paradigm of C and encode this new version in an Object Oriented Language like C++. This would not only have a good impact on the development of GIDes++ but also in the following versions as it would be all much more organized and gaining from all Object Oriented paradigm advantages.

Visually, a lot was changed too. One tried to create a more pleasant visual ambient which could be more catchy and enjoyable for the user. The first differences noticeable reside on colour changes, like the background and *userstroke* and remaining lines. Some other changes were introduced in the visual indication resulting from user interaction. As for the *Expectation List*, one concluded that imposing the upper-left position on the screen of this had many disadvantages and could turn the experience of selecting a suggestion into an annoying process. Consequently, it was changed so that the user could move it around like a floating frame and every time this list would appear, it would be in the last place left. Objects were also part of the "face-lifting". It was wished to be created a use experience that resembled somehow computer games, which try to be attractive and visually appealing, involving the user in a deeper level. As part of this plan, some graphic effects were added enriching not only the application's aspect, but also the visual information received by the user.

The first effect applied was Motion Blur to camera view changes. When a determined view angle is chosen an animation is performed, changing the camera's orientation and zooming on the desired entity. During the animation, the well known motion blur effect provides realism since humans see real life in this manner and gives better notion of how the camera is moving and in what way is the scene being watched. Another effect was applied to object selection. When an object is selected, a glow effect appears, highlighting the object. This is not only stunting and elegant, but this glow as the particularity of glowing even when the object is behind another. A sort of ghost object appears when the object hides at the back of another, enlightening the user where the object is, what orientation does it have and how is he shaped. This information can be important when using real time object manipulators, like real time rotation or translation, since we can follow the object in every time, allowing us to adjust the object even when it is under another.

When doing boolean operations, this effect could be of a great help too, as one can watch an object part inside the other, seeing, if the case is a subtraction or an intersection, what will result from that operation.

4. MEASURES LAYER

Being a sketch-based modeller, no precision in the object creation input can be assured. Unlike other CAD systems, where dimensions of objects are defined as they are being constructed, in GIDes ++ 3D primitives are assembled from strokes which can't guarantee any precision when it comes to measures. Even with the help of some kind of ruler or a grid with defined unit size, the user can not be accurate enough.

One efficient way to define proportions, but not real measures, is to use support lines with geometric constructions. Even though this is of great help and has an enormous potential, it doesn't unravel the entire problem.

Firstly, how would the user change to this layer? Would there be any explicit information that would point us out how to do so? And then, the other logic questions, how could an interface that would both maintain the naturalness be created, following the paper sketch analogy but always having in mind that the less activity the user needs to do, the better!

As for the layer switcher, an explicit command was chosen. These operations had to be kept in an individual and independent layer since many of the commands performed in this layer were just like normal drawing commands or selection, and if the user was allowed to change measures in the drawing layer, commands could be mistaken. Besides, as has been stated, one intends for users to first develop a prototype and only then concentrate on measures, this way his focus is solely on creativity.

One other main reason for this decision was that much of the operations and simple mouse over actions required lots of computation, therefore in some situations the application would stop responding to user's queries, which isn't pleasant to happen when the user is drawing and wants to see everything he is doing in real time. As a result one can change the layer – drawing or measures – by pressing a button on the fixed toolbar, or by issuing a voice command.

Staying in the measures layer is obvious that the user wants to visualize as easily as possible the object's dimensions. The first approach to this problem was to show every edge measure from a selected object. But soon enough one realized that this was too much information and could even be impossible to show it all at once. It was also noticed that if one knows an edge's size, by proportion analogy, the size of the others can be easily speculated and consequently there were no reasons to report more than one measure at a time. Well, this probably isn't entirely true, but knowing which edges were important to inform at a given moment is a much too complex and non-trivial and we would have efforts would have been wasted and other, more important, areas might have been neglected.

The idea was to have a way of easily and quickly displaying an edge's measure permitting the user to verify a vast number of edges in a short period of time. A solution was found, one that is plain, simple and efficient: as the

mouse moves over and edge, automatically that edge – from one vertex to the other – is highlighted and numeric indication appears, stating the edge's length.

Finally, the only thing left was a method to allow measure changes. Initially, the objective was to copy the way designers and architects set measures. This is usually done by stroking two lines, both starting near vertexes and stating the measure in the middle of them. As a result, changing a measure in GIDES++ would be performed in the same way. The user draws a line starting from a desired vertex and then from another and the length between these two vertexes would come into sight.

Posterior to this, the user would specify the new length and the change was performed. As another possibility, the users can simply ignore this measure and continue to perform other action, enabling the visualization of distances between vertexes not linked by an object edge.

Still the user had always to stroke two lines and just then would he be able to make any changes. So and keeping in mind that simple and easy is better, a complementary way of setting measures was implemented. Simply by clicking over an object's edge, the user is allowed to change it. Of course this solution only has some vertexes combinations in mind, but we could always recur to the previous technique for others.

A particular and important moment was when the user would specify the new length. The use of the keyboard was out of the question as this application was, since the beginning, thought to be interacted with non-conventional input like mouse or keyboard.

Showing a little numeric keyboard in a frame with special characters like coma or point wasn't very convincing, stylish or even efficient. The logic way was to turn to handwriting recognition. This was the most natural, pleasant and efficient way to input measures. Not only is it quick, but designers are used to write measures down on paper. Therefore, a grey frame appears in the screen and the user can simply write down the desired number. To recognize handwriting, new Handwriting Recognition is used which is part of Windows XP Tablet PC Edition. The recognizer gives the power to recognize, in an efficient way, human writing. Some tests were performed to test the recognizer liability and success rate.

5. SPEECH RECOGNITION

As stated before using voice in order to invoke commands and switch between layers seemed to be a very feasible way of eliminating the need for a keyboard and even the right mouse button.

At the time it was decided to use an English version, hand-writing recognition that took advantage of the Microsoft Windows Tablet PC recognizers, was already at use, so keeping things simple the speech recognizer from the same software house was used. Very good things about the system had already been heard and applications that use it successfully have, in fact, been seen. One other

advantage was its use of the XML format to describe the grammar providing greater flexibility.

5.1 Grammar

Voice commands could have been applied to all the functionality of the system but the main goal was to remove the need for a right mouse button on the pen. As such most of the commands enable the user to switch tools or layers, and also allow for the opening and saving of files as well as exiting from the program.

One of the main problems that came up was the fact that having no keyboard didn't enable a simple way to engage or disengage speech recognition. As a result the system keeps trying to recognize commands from what the user is mumbling while working. Fortunately one was able to reduce this to a minimum and have had only a few changes of view while working which are of minor effect to the work in progress.

The main solution for this problem was the use of commands with at least two words. Thus for every command regarding and object the word "this" was added and for camera operations the word "view" was used. A special case was for the paste command where the keyword "this" wouldn't make much sense, so both "object" and "objects" were used. The same is applied to the copy command.

For switching layers voice commands were also introduced. These are composed simply of the layer intended, drawing or measures, followed by the keyword "layer".

The command for exiting the program is, as implied by this description, "exit program".

One has to call attention for a special character used in the description of the grammars used by the MSSpeech API which is the "+" sign. By putting this before a given word in a grammar rule the recognizer is requested to have greater confidence before stating that it recognized that given word. This way one can avoid much misrecognition. On the other hand this will require the user to pronounce the words in a clearer way for the system to be able to recognize them with confidence. What was done was trying to reach a balance. For this the "+" has been applied to the main keyword in a given command.

5.2 Accuracy test

MS Speech	U1 - 22 / 22 = 1.000 = 100 %
	U2 - 20 / 22 = 0.909 = 91 %
	U3 - 17 / 22 = 0.773 = 77 %
	Average : 89.3 %

Table 1 – Speech Recognition Accuracy

It is important to state that these tests were performed in a quiet environment. Most designers and modellers work in rooms with other colleagues so perhaps while one of them is issuing a command it most probably will affect others. Still, since the main purpose was to try to apply a different modality to a system as powerful and demanding as a CAD system these accuracy rates are enough to try to draw some conclusions with the experiments lead with a group of users. Results regarding comments re-

ceived from these testers will be presented further on and discussed.

6. HANDWRITING RECOGNITION

The main goal for *Measures Layer* was for it to allow the user to change the measure of a given model by simply selecting the edge to be altered and, by simply writing the new size, affect the object and make it follow the intended size. For this we a powerful handwriting recognition engine was needed.

Since the target platform when at the start of development of this application was the Tablet PC it was decided to give the inherent recognizer a try. After installing the Windows XP Tablet PC Edition in an ordinary computer one has a powerful handwriting recognizer at hand. The SDK is also freely available and using it isn't very difficult since it need not be event driven, that is to say, one can simply send it the *ink* received in a given window and ask it for the most probable result.

6.1 Accuracy Test

The accuracy tests conducted were done directly in our application. The user had to select an edge and introduce a set of ten different measures and one would determine, by a simple mouse-over, if the edge had the correct new size.

One drawback of the current implementation is that *factoids* are not being used. That is to say that the recognizer doesn't know that users are only writing numbers on the screen, thus it might return letters or symbols. If this happens the edge doesn't change.

The numbers chosen for these tests seemed adequate since they are a mix of several similar looking digits. There are five numbers with no decimal part and others with two decimal digits in order to determine if the dot is a source of problems for the recognizer.

	<i>U1</i>	<i>U2</i>	<i>U3</i>
Rec. Rate	80 %	80 %	90 %
	Avg. Rec. Rate		83.3 %

Table 1 – Handwriting Recognition

Looking at the results of the misrecognitions above of Table 1, one finds that these are very close to the number the user intended to insert. The fact that these tests were conducted on an ordinary PC using the mouse as an input is one of main causes of fault. Even a human could have misinterpreted the numbers which if when drawn by hand with a pen are, sometimes, already difficult to interpret; with a mouse become a real challenge to figure out.

An 83.3% recognition rate is very good taking into account the fact that factoids are still not in use. It is also important to take into account that these tests were performed in a relative early stage of development simply to determine if the recognizer would be enough, thus the small group used.

7. USABILITY TEST

One of the most important steps in software development is to test it with users. Only then can the development team get to know how does the interface work in the real world and what features should be changed, enhanced, added or even removed. The direction of development normally changes and past work is normally reviewed after these tests sessions and usually lead to product optimization in terms of its efficiency and efficacy of its interface.

In this particular case, the main goal was to test the (intelligent multimodal) interface and not the features and functionalities. As a result, it was decided that the task performed by real users should be something more abstract and not centred in a real object, like a chair or table, and should use interface features as extensively as possible.

The construction of a real object (e.g. chair) as task could direct the user attention to other matters and not exclusively to the interface itself. Drawing an abstract object with no real meaning would not distract the users, unlike a know object which could divert their opinions, expressing whether they think the application can easily create that particular object or not. This way one can focus on the feeling and naturalness of the interface for the user.

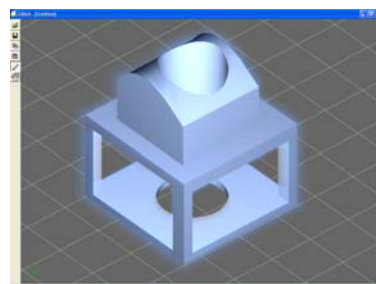


Figure 2 – Test model

This test was performed by a group of users, which had never used GIDeS++ time enough to be considered to have any expertise over this application, and a first user who was an expert in order to get some indicator over the easiness of learning of GIDeS++. In order to have some feedback regarding the use of speech, every user did the task two times: in the first, not using any speech, recurring to the “plan B”, namely toolbars and view alterations in the green axis; And in the posterior attempt, only speech could be used, over the alternatives. With this, a proper evaluation of the speech modality could be reached by comparing the results for every user. Unfortunately, these experiments weren't performed in a screen equipped with a digitizer, but in a normal monitor with a mouse instead of a pen. This can deceive a bit the results as working with a mouse is far more complex than with a pen in this kind of interface. A particular part that suffers much from this is the handwriting input, since trying to write numbers with a mouse is far more time consuming than with a pen.

As for the results, shown next in table 3, the only output was the time needed to finish the task. Initially, we con-

sidered recording the number of mistakes, but as this task was quite straightforward a rather low quantity of errors were expected, thus being this information inconsequential and irrelevant. As no comparison with conventional CAD systems was made the number of clicks – or actions – weren't traced either.

	With Speech Commands			
	U1*	U2	U3	U4
Time	1:43.02	3:07.81	3:07.97	2:06.57
Average: Inexperience Users	-	02:47,5		
Average	02:31,3			

	Without Speech Commands			
	U1*	U2	U3	U4
Time	2:00.51	3:40.19	4:27.29	2:52.21
Average: Inexperience Users	-	03:39,9		
Average	03:15,1			

* Expert User

Table 3 – Usability Tests

As one can observe, all inexperienced users had lower performances than the expert. This was expected, but the gap between both types of users isn't that much, being the expert almost just 33% quicker in the “with speech” test. Taking into consideration the gap between the first user and the others - first time users - one can somehow state that this interface is easy to learn and to get at ease with. Another immediate observation that can be made is the difference in the Average times in both tests. When performing with speech commands, users always finished earlier. Some users even spent 1:20 less in the second trial which is a huge improvement.

One has to take into account that in the “With Speech Commands” test, users were doing the same thing for the second time, therefore, it makes sense for the times to be shorter. The tests had to be done in this way, or else, no conclusion could be taken and, having all users experienced the same test over two different methods, conclusions from users could be more useful and their

comparison between with and without voice would be more founded and easy to make. Even so, looking to the expert user, that had executed many times this task before performing any of the real tests, the difference is still there.

So, it's safe to assert that speech modality is an advantage to this application, allowing users to avoid toolbar navigation and centralizing some of the commands in a way that humans are used to exercise, this is, speaking commands.

During the tests, it was noticed that users lost much time looking for a specific tool, trying to remember in what toolbar context they were in, or pointing the cursor to a determined command area of the screen. Instead, with voice, they could change tool, make view modifications, etc... And prepare them to the next action, moving the cursor (hand) to the point where they would draw next.

All users seemed most impressed with the interface potentials and the mixture of various modalities. Handwriting recognition was probably the one that had the most impact. It is not very common to see this modality in big use nowadays and mainly with so good results and good application.

Speech was noticeable, but it is already very seen on media or films for a long time. One thing most users weren't expecting was that using voice really enhanced their performance and was so pleasant to use.

All users stated that the multimodalities were put to good use and really helped the development and creation of object in GIDeS++. Not only this, but they expressed their satisfaction in using such things and said they were extremely enjoyable to use, commenting that it looked like a sci-fi software, but that really worked.

8. CONCLUSION

The main goal of this project was to try several approaches, in this case modalities, for introducing several improvements to a CAD system in order to make it more natural. This added to the fact that the starting point was an already pretty natural system it makes one feel that the goal was achieved. Even if several issues are still to be covered one certainly was able to show, at least to the testers, that much can be done in order to “camouflage” the inherent complexity of a modeling system.

9. REFERENCES

[Pereira00] Pereira J P, Jorge J A, Branco V, Ferreira F N: GIDeS: Uma Abordagem Caligráfica à Edição, 9º Encontro Português de Computação Gráfica, 2000

Construção dinâmica de um sistema interactivo para visualização do código intermédio do processo de compilação

Paulo Jorge Matos
Instituto Politécnico de Bragança
Campus de Santa Apolónia, 5300 Bragança
pmat@ipb.pt

Pedro Rangel Henriques
Universidade do Minho
Gualtar, 4700 Braga
prh@di.uminho.pt

Resumo

O DOLPHIN é uma framework que suporta o desenvolvimento de compiladores de alta performance, multi-linguagem e multi-arquitectura. Inclui diversas componentes, uma das quais designada por DOLPHIN-Front-End for the Web, que reúne as componentes relacionadas com a web, nomeadamente: um ambiente integrado de desenvolvimento, que entre outras funcionalidades permite visualizar, sob a forma de hiper-texto, a informação ao longo do processo de compilação. Neste artigo é apresentada uma nova solução desenvolvida com o mesmo objectivo, o de visualizar a informação, mas agora recorrendo a uma solução gráfica, interactiva e que é construída dinamicamente a partir de uma descrição da informação feita em XML.

Palavras chave

Visualização gráfica, sistemas interactivos, XML, compiladores.

Abstract

O DOLPHIN is a framework conceived to help the development of high-performance, multi-language and multi-architecture compilers. It includes several components, one of them is designated by DOLPHIN – Front-End for the Web. This one is responsible for all DOLPHIN items related with the web, namely an integrated development environment that allows, between other things, to observe and analyze, as hiper-text, the information along the compilation process. In this article is introduced a new solution developed with this intent (visualize and analyze the information of the compilation process), with an interactive graphical interface, built dynamically from a description of the information done at XML.

Key words

Graphical visualization, interactive systems, XML, compilers.

1. INTRODUÇÃO

O DOLPHIN é uma *framework* que suporta o desenvolvimento de compiladores de alta performance, multi-linguagem e multi-arquitectura [Matos99, Matos02]. É formada por diversas componentes, mas a principal consiste num conjunto de ferramentas e rotinas, que associadas a um modelo de representação de código, permitem o desenvolvimento de compiladores. Dado que a maioria dessas rotinas utiliza soluções algorítmicas de elevada complexidade, cedo surgiu a necessidade de dotar o DOLPHIN de ferramentas para verificar e otimizar o seu funcionamento. O objectivo era permitir monitorizar e observar os efeitos das rotinas sobre o código submetido ao compilador, à semelhança do que é feito no VCG [Sander95].

O normal processo de *debugging* do código não é suficiente nem eficaz nestas situações, isto porque a compilação envolve um conjunto muito vasto de tarefas e lida com grandes quantidades de informação.

A primeira solução implementada para monitorizar e observar os efeitos das rotinas, consistiu em dotar os compiladores desenvolvidos com o DOLPHIN de mecanismos para gerar a informação sobre o processo de compilação [Matos03]. Informação essa que era formatada através de uma linguagem hiper-textual, mais especificamente HTML. Esta solução não só possui os requisitos necessários como permite aceder à informação utilizando um simples browser. O que era uma operação dantesca, passou a ser facilmente exequível, permitindo por exemplo: navegar na estrutura do programa; relacionar diferentes formas de representação (ex. vincular expressões com os respectivos vértices do GFC); aceder a informação anexa associada a determinados elementos da RI (ex. vincular variáveis com a respectiva informação da tabela de identificadores); relacionar elementos de fases distintas do processo de compilação (fundamental para verificar o funcionamento das rotinas); e visualizar informação temporária gerada pelos processos de análise.

No sentido de facilitar o acesso e potenciar a utilização, a solução foi colocada disponível via Web, permitindo através de um browser fazer o *upload* do código fonte, o qual é remetido para uma *Common Gateway Interface* (CGI), que trata de invocar o compilador para gerar os ficheiros HTML. Estes são posteriormente redireccionados para o utilizador.

O resultado foi tão bom que deu origem ao *DOLPHIN -- Front-End for the Web* (DOLPHIN-FEW), o qual inclui entre os seus objectivos, a implementação de um ambiente integrado de desenvolvimento (IDE), para editar e compilar programas, com suporte multi-linguagem, multi-arquitectura e uma forte componente de optimização (características do DOLPHIN), disponível via Web. Serve também para verificar e optimizar o funcionamento das rotinas implementadas no DOLPHIN e é um auxiliar precioso no ensino de tópicos relacionados com o desenvolvimento de compiladores.

No entanto a experiência da utilização do IDE levou-nos a concluir que mesmo esta forma de representação hipertextual não é muito apelativa, nomeadamente quando utilizada para fins pedagógicos. O principal motivo estava relacionado com a quantidade de informação gerada e com a forma como esta é exposta ao utilizador. Tornava-se cada vez mais claro a falta de mecanismos que melhorassem a interactividade com o utilizador. Surgiu assim o trabalho apresentado neste artigo. Trata-se de um interface gráfico, interactivo, que é construído dinamicamente. É uma solução nova que utiliza diversas tecnologias, designadamente: *Schemas*, *eXtended Markup Language* (XML) e *Macromedia Flash*¹.

Apesar de especificamente desenvolvida para representar a informação do processo de compilação, esta solução pode também ser utilizada, sem alterações, para visualizar outras fontes de informação capazes de gerar XML.

Na secção seguinte descreve-se a implementação desta solução e na terceira secção, as conclusões.

2. O SISTEMA DE VISUALIZAÇÃO

Dada a quantidade de informação envolvida e à interactividade que requerida para a nova solução, era importante que o esforço de computação fosse transferido o máximo possível para o lado do cliente (browser). Este receberia a informação e tratava de gerir a visualização e a interactividade com o utilizador.

Após a análise de várias soluções, a opção recaiu sobre a utilização conjunta de XML com *Macromedia Flash*. Esta solução permite transferir toda informação para o cliente, em formato XML, depois utilizando a API disponibilizada através do Flash, do *Document Object Model* proposto pelo W3C (<http://www.w3.org/TR/REC-DOM-Level-1/>), constrói-se um objecto que representa o documento XML. Esse objecto pode ser manipulado pelo *Action Script* (a linguagem do *Macromedia Flash*) para construir a representação gráfica.

A Figura 1 representa a arquitectura da nova solução. O utilizador submete, através do IDE, o ficheiro de código fonte a uma CGI. Esta invoca o compilador para gerar os ficheiros em XML e, posteriormente, redirecciona para o browser a aplicação em Flash, com o endereço do ficheiro principal. A aplicação, ao ser executada no browser, faz o download dos ficheiros XML e constrói a representação gráfica.

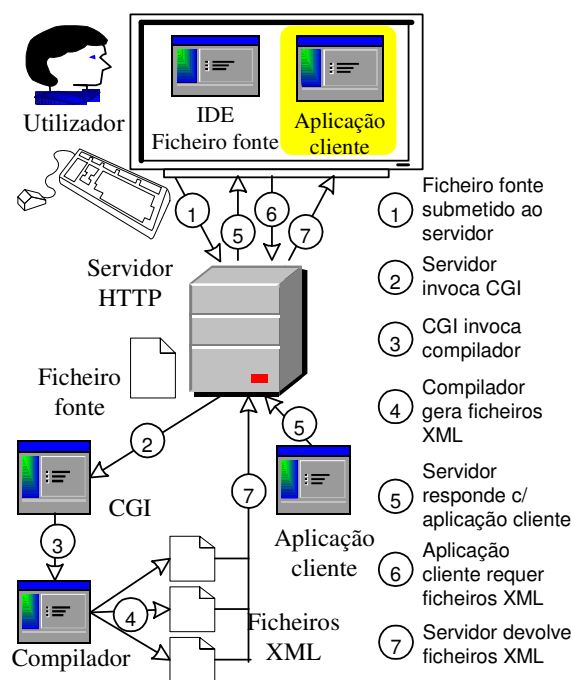


Figura 1. Arquitectura da nova solução.

2.1 Schema para a RI do DOLPHIN

A primeira tarefa foi definir o formato dos documentos XML. Dado que o DOLPHIN utiliza objectos (classes em C++) para construir a RI, cada um desses objectos deu origem a um elemento XML, que é representado pelo identificador do objecto. Cada elemento contém outros elementos, um por cada uma das variáveis do objecto. Esses elementos internos são representados pelo mesmo processo, isto é, a designação do elemento corresponde ao identificador da variável e o conteúdo corresponde ao respectivo valor. Quando as variáveis são compostas (arrays, estruturas, classes, etc), então o conteúdo de cada elemento consiste num conjunto de outros elementos.

A Figura 2 representa várias classes utilizadas na RI (*IdentTable*, *FlowNode* e *JumpNode*) e o XML de dois objectos: um instanciado da classe *IdentTable* (*table*) e outro da classe *JumpNode* (*jump*). Este exemplo contém dois casos que merecem um tratamento especial: o *in* e o *table*. O primeiro caso é um *Set* de apontadores para objectos do tipo *FlowNode* (*template* em C++). A conversão desta estrutura segundo as regras anteriores, resulta na representação dos endereços dos objectos *FlowNode*, em vez dos próprios objectos, pelo que na geração do XML o que é utilizado é o objecto apontado e não o apontador. O segundo caso é o da variável *table*, uma tabela de Hash com chaves do tipo *string* e valores

¹ Macromedia Flash é uma marca registada da Macromedia, Inc.

do tipo *Cell* (apontador). Neste caso, a tabela de Hash é percorrida linearmente e por cada entrada da tabela são gerados dois elementos: o *<key>* e o *<val>*, que representam, respectivamente, a chave e o valor de cada entrada.

Houve ainda a necessidade de obter uma solução para resolver o problema derivado da herança entre classes. Nestes casos, a representação XML inclui não só as variáveis da classe do objecto, mas também as variáveis das classes ascendentes. A Figura 2 ilustra um destes exemplos (entre a classe *JumpNode* e a *FlowNode*).

C++	XML
class IdentTable{	<table><th>
THash<char*,Cell*> th;	<key>...</key>
...	<val>...</val>...
};	</th>...</table>
IdentTable table;	
 class FlowNode{	<jump>
TNode tnode;	<tnode>JNODE</tnode>
Set<FlowNode *> in;	<in>
...	<jnode>...</jnode>
};	...
class JumpNode	</in>...
:public FlowNode{	<jp>...</jp>
Jump jp;	...
...	</jump>
};	
JumpNode jump;	

Figura 2. Tratamento da herança entre classes.

Devido à quantidade de informação e ao número considerável de classes utilizadas na RI, surgiram algumas complicações (valores incorrectos, etiquetas não concluídas, etc), pelo que se optou por desenhar um *Schema* para definir e validar a estrutura dos documentos.

```
<complexType name="FlowNode">
  <sequence>
    <element name="tnode" type="TNode"/>
  </sequence>
</complexType>
<complexType name="JumpNode">
  <complexContent>
    <extension base="FlowNode">
      <sequence>
        <element name="jp" type="Jump"/>
      </sequence>
    </extension>
  </complexContent>
</complexType>
```

Figura 3. Schemas das classes *FlowNode* e *JumpNode*.

No desenho do *Schema* cada classe deu origem a um tipo complexo (*ComplexType*). A herança entre classes foi tratada através dos próprios mecanismos de herança dos *Schemas* (através do *restriction base=".."*). A Figura 3 ilustra este procedimento para as classes *JumpNode* e *FlowNode* da Figura 2. É apenas de realçar a utilização de tipos simples (*simpleType*) para representar enumerações, como é o caso do *TNode*, restringindo

assim os valores que podem ser utilizados; e do *choice* para representar uniões (*union's*) e elementos que podem assumir diferentes tipos (polimorfismo). A Figura 4 mostra exemplos destes casos.

Apenas para concluir esta secção, falta acrescentar que o *Schema* definido representa toda a informação sob a forma de elementos. Os atributos, como se explica mais adiante, são apenas utilizados para fornecer informação sobre a forma e comportamento da representação gráfica.

```
<simpleType name="TNode"><restriction base="xsd:string">
  <enumeration value="JNODE"/>
  <enumeration value="CJNODE"/>
  ...
</restriction></simpleType>

<complexType name="Function">
  <sequence>...
  <element name="setNodes"><complexType>
    <sequence minOccurs="0" maxOccurs="unbounded">
      <choice>
        <element name="jNode" type="JumpNode"/>
        <element name="rNode" type="ReturnNode"/>
        ...
      </choice></sequence></complexType>
    </element></sequence></complexType>
```

Figura 4. Enumerações e variáveis polimórficas.

2.2 Os componentes Flash

A aplicação Flash é formada por um corpo principal e um conjunto de componentes. No corpo principal realiza-se o *setup* da aplicação, que consiste em fazer: o download dos ficheiros XML, construir o *objecto* XML e instanciar um objecto do tipo *DefaultObject* (uma das componentes Flash desenvolvidas), o qual inicia a construção da representação gráfica.

Os componentes desenvolvidos estão preparados para interpretar alguns atributos, que condicionam a representação e o comportamento da visualização. Os principais atributos são:

__ID - *String* que identifica o objecto;

__type - *String* que define o tipo de componente. Por omissão é utilizado o *DefaultObject*;

__vtype - *String* que define o estado visual do objecto. Existem três estados possíveis:

popup - Assinala objectos que apenas surgem mediante um evento que ocorre num outro objecto (*onOver*, *onClick*, etc). Estes objectos podem ser convertidos para o estado *standalone*;

standalone - Assinala objectos cuja visualização não depende exclusivamente de outros objectos. Podem ser escondidos, redimensionados, deslocados e convertidos para o tipo *popup*;

root - Assinala o objecto inicial da representação. Está sempre visível e apenas pode ser redimensionado ou deslocado;

__label - *String* que define a *label* do objecto ou o objecto ao qual o elemento deve ficar associado;

`__x`, `__y` - Atributos do tipo real, que representam a posição horizontal e vertical do objecto;

`__visible` - *Booleano* que indica se o objecto está ou não visível.

2.2.1 Componentes genéricos

A aplicação desenvolvida possui componentes de uso genérico, que podem ser utilizados para visualizar um qualquer documento XML, e componentes construídos especificamente para o DOLPHIN. Ambos têm a capacidade de ler os atributos `__x`, `__y`, `__visible`, `__vtype` e `__label`, para definirem respectivamente a: sua posição, se estão ou não visíveis, como se devem comportar (*popup*, *standalone*, *root*), e se devem utilizar alguma *label* específica na identificação do objecto.

O mais genérico dos componentes é o *DefaultObject*, que corresponde sempre ao objecto inicial da representação, e a partir do qual os restantes são construídos. A Figura 5 permite ver o aspecto deste objecto no formato *standalone* (o formato *popup* pode ser visualizado na Figura 6). De notar a existência de dois botões no canto superior direito: um para redimensionar e o outro para fechar o objecto; e de uma *Label* no canto superior esquerdo, normalmente com a designação do elemento raiz do objecto XML, ou então com o valor do atributo `__label` (caso este esteja definido). À frente desta *label* pode ainda existir um *link* para o objecto ascendente (na hierarquia do objecto XML). A restante informação representada diz respeito aos elementos descendentes.

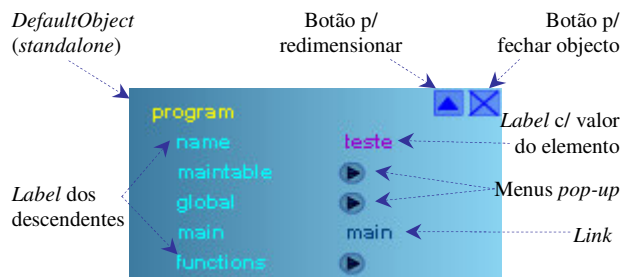


Figura 5. *DefaultObject* no formato *standalone*.

O próprio *DefaultObject* percorre a lista dos elementos descendentes e constrói dinamicamente a representação interna. Cada descendente é representado por um par formado por uma *label*, com a designação do respectivo elemento, e por um objecto do tipo *label*, *link* ou *popup* menu. A *label* é utilizada para representar o valor de elementos do tipo simples. O *popup* menu e o *link* são ambos utilizados para ligações a elementos compostos. Há no entanto diferenças entre eles, para além do aspecto gráfico (um é representado por um botão animado e o outro por texto), o *popup* é utilizado nos descendentes do tipo composto e força a criação dos objectos para esses elementos (passando a respectiva sub árvore do objecto XML). Nestas condições o atributo `__type` normalmente não está definido, ou então está definido com o valor *"sequence"*.

O *link* está sempre associado a elementos do tipo simples, mas que representam ligações a elementos do tipo

composto. Não cria nenhum objecto novo, espera-se no entanto que o objecto alvo seja criado num outro contexto, para poder ser correctamente referenciado pelo *link*. O atributo `__type` é definido com o valor *"link"* e o objecto a referenciar é definido através do atributo `__label` (ex: `<main __type="link" __label="ID1200">main</main>`).

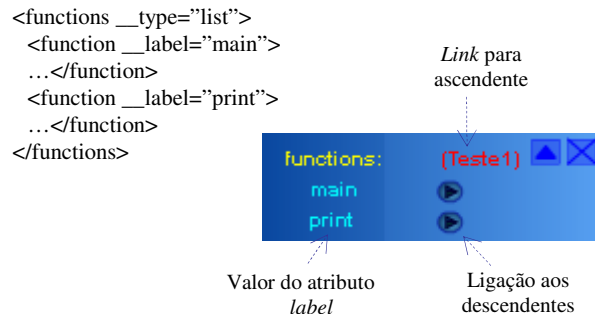


Figura 6. *DefaultObject* para uma lista.

O *link* e o *popup* menu são semelhantes no funcionamento, isto é, quando o apontador passa sobre eles o objecto que representam fica em foco (se estava invisível passa a visível). Caso o utilizador click na tecla esquerda do rato, o objecto passa do estado *popup* para *standalone* ou vice-versa (depende do estado inicial). Ao sair de cima do botão *popup* ou do *link* o objecto volta ao estado inicial, ou ao estado que ficou após se pressionar a tecla do rato.

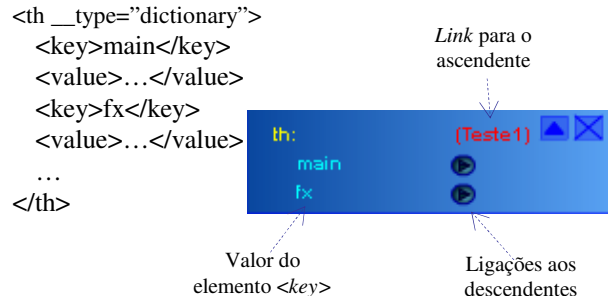


Figura 7. *DefaultObject* para um dicionário.

O atributo `__type` serve para modificar a forma como a informação é processada no *DefaultObject*. Colocando `__type` com o valor *"lsuper"*, significa que deve ser colocado um *link*, à frente da *label* do objecto, com ligação ao elemento ascendente. Isto é importante para associar o objecto ao seu ascendente e para distinguir objectos do mesmo tipo.

Para elementos cujos descendentes são todos do mesmo tipo, o `__type` deve ser definido com o valor *"list"*, o que permite representar esses descendentes, não pela própria designação (que é igual para todos os descendentes), mas pelo valor do atributo `__label`. A Figura 6 mostra um destes casos.

O atributo `__type` pode ainda assumir o valor de *"dictionary"*. Neste caso os descendentes são processados dois a dois, em que o valor do primeiro é utilizado como *label* e o segundo, como objecto que surge do lado direito

(label, botão *pop-up* ou link). A Figura 7 representa uma destas situações.

2.2.2 Componentes específicos

Os componentes específicos visam dar uma interpretação mais intuitiva da informação gerada pelo DOLPHIN. Existem cinco componentes específicos: o *Node* e o *CNode*, que representam vértices do GFC; o *Expression*, que representa nodos das árvores de expressões; e o *DTlist* e *DT* que representam, respectivamente, listas de *DataTransfer* e *DataTransfer's* (elementos da RI que visam caracterizar as trocas de dados). A Figura 8 mostra um exemplo que inclui vários destes objectos.

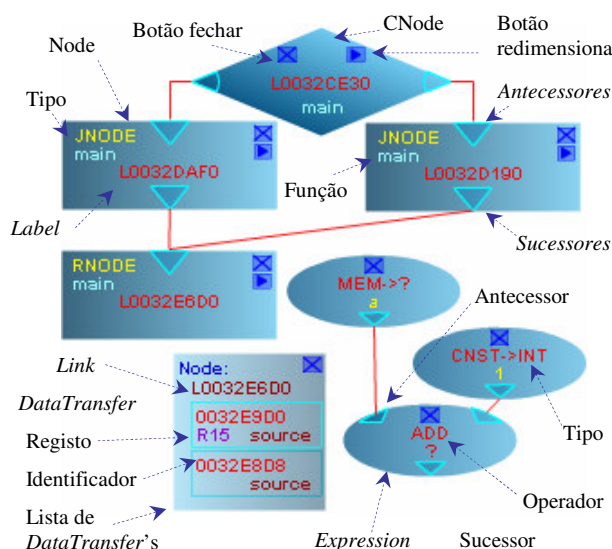


Figura 8. Visualização do GFC, dos *DataTransfer's* e das árvores de expressões.

O *Node* é representado por um rectângulo que contém: duas *label's*, uma com a identificação do vértice e a outra com o tipo (RNODE/JNODE); um *link* para a função à qual o vértice pertence; um botão para fechar o vértice e outro para aceder à respectiva lista de *DataTransfer's*. Existem ainda áreas sensíveis que se activam à passagem do rato. No caso do *Node* permitem aceder aos vértices antecessores e sucessores do GFC. O comportamento das zonas sensíveis é muito semelhante ao do *link* e do botão *pop-up*, a diferença está no facto de desenharem as ligações entre vértices. O componente *CNode* apenas difere do *Node* na representação gráfica (losango). Ambos podem ser deslocados pelo ecrã.

O componente *DTlist* possui um *link*, para o respectivo vértice do GFC, e uma lista de instâncias derivadas do componente *DT*. Cada uma contém: a identificação do *DataTransfer*; o registo utilizado na operação; e um *link* para uma instancia do componente *Expression*. O *DTlist* pode ser fechado e deslocado, já os *DT* são apenas elementos estáticos dentro do *DTlist*.

O componente *Expression*, representado por uma forma oval, contém duas *label's*: uma que identifica o tipo de operação; e outra que identifica o tipo de dado (inteiro, real, etc), e se necessário mais alguma informação

relevante. Contém ainda um botão para fechar o objecto e algumas zonas sensíveis: duas para aceder aos descendentes da expressão (esquerdo e direito) e uma terceira para aceder aos sucessores.

3. CONCLUSÕES

A utilização de componentes Flash juntamente com XML é cada vez mais uma fórmula de sucesso para animar e representar informação. Julgamos nós que esta fórmula foi aqui utilizada racionalmente, para resolver um problema real com necessidades específicas. Os ganhos inerentes da implementação desta aplicação ainda estão por avaliar, dado que por um lado ainda está em desenvolvimento e por outro ainda não foi disponibilizada aos potenciais utilizadores. No entanto possui já algumas características importantes, a saber: é simples de utilizar; em nossa opinião está bem desenhada; interactiva bastante bem com o utilizador; permite satisfazer determinadas necessidades que de outra forma seriam inacessíveis; e disponibiliza a informação de uma forma bastante amigável.

O desenvolvimento desta aplicação serviu também para desenhar um *Schema* para a RI do DOLPHIN, o qual prevemos utilizar em outros projectos, alguns dos quais bastante inovadores.

É ainda de salientar que existem planos para dar continuidade ao desenvolvimento desta aplicação, nomeadamente no sentido de: construir novos componentes para representar informação que ficou de fora na actual solução (ex. informação temporária sobre os processos de análise); e construir uma solução para simular o comportamento das rotinas de análise e optimização, a qual passa pela implementação de novos componentes, não para representar mais informação, mas para processar os actuais, permitindo assim simular o comportamento das soluções implementadas ao nível do DOLPHIN.

Não fica de fora a possibilidade de evoluir os componentes genéricos para lidar com outro tipo de dados, como por exemplo: imagens, som, filmes, páginas web, etc.

Para concluir, é de realçar novamente que esta aplicação pode ser utilizada para animar e visualizar outras fontes de informação, sejam capazes de gerar XML.

4. REFERÊNCIAS

- [Matos99] P. Matos. *Estudo e desenvolvimento de sistemas de geração de back-ends para compiladores*. Master Thesis, Universidade do Minho, 1999.
- [Matos02] P. Matos. *The DOLPHIN framework*. Technical Report, Universidade do Minho 2002.
- [Matos03] P. Matos, P. Henriques. DOLPHIN-FEW – An example of a web system to analyze and study compiler behavior. Proceedings of the E-Society Conference, 2003.
- [Sander95] G. Sander. VCG Visualization of Compiler Graphs. Technical Report A01-95, Universität des Saarlandes, 1995.

Lista de Artigos

AIF - Uma estrutura de dados B-rep para modelos poligonais	
Frutuoso Silva, Abel Gomes	1
ARK multi-utilizador	
Pedro Pereira, Nuno Matos, Luís Grave, Adérito Marcos	119
BlobMaker: Free-Form Modelling with Variational Implicit Surfaces	
Bruno Araújo, Joaquim Jorge	17
Computação Gráfica e Arqueologia Urbana: o caso de Bracara Augusta	
Paulo Bernardes, Manuela Martins	63
Construção dinâmica de um sistema interactivo para visualização do código intermédio do processo de compilação	
Paulo Matos, Pedro Henriques	173
Corte de Peças de Pedra: Uma Aplicação da Computação Gráfica	
Samuel Silva, Filipe Marques, Manuel Raposo, Joaquim Madeira, Beatriz Sousa Santos	163
Geração de Ambientes Rodoviários Para Simulação de Condução	
Carlos Campos, Vítor Cunha, Miguel Leitão	143
GIDes ++	
Joaquim Jorge, Nelson Silva, Tiago Cardoso	167
Identificação e reconstrução 3D de objectos numa malha urbana através de nuvens de pontos georeferenciados	
Rodrigo Garcia, Mauro Farracha, José Dionísio, Alexandre Bernardino, João Costeira	83
Image Manipulation through Gestures	
José Dias, Pedro Nande, Pedro Santos, Nuno Barata, André Correia	111
Modelação Expedita de Cenas Urbanas	
António Coelho, A. Augusto Sousa, F. Nunes Ferreira	73
Polygon Detection from a Set of Lines	
Alfredo Ferreira Jr., Manuel Fonseca, Joaquim Jorge	159
Representação BSP de Curvas Implícitas 2D	
Francisco Morgado, Abel Gomes	9
Simulação e avaliação de um projecto urbano através dos computadores	
Pedro Leão Neto	137
Sistema de Realidade Virtual para tratamento de Fobias	
Vitor Cunha, Miguel Leitão	131
Task Analysis and Scenario-Based Design of Calligraphic Interfaces	
Filipe Dias, Manuel Fonseca, Joaquim Jorge	153
Técnicas de realidade virtual aplicadas ao ensino na área da construção	
A. Zita Sampaio, Pedro G. Henriques, Pedro S. Ferreira	125
Using Sketches to Edit Electronic Documents	
Pedro Coelho, Gonçalo Faria, Daniel Gonçalves, Joaquim Jorge	149
Variação Dinâmica do Nível de Detalhe na Simulação Estável de Tecidos	
Fernando Birra, Manuel Santos	53
VGLib2D - Biblioteca de classes para a visualização gráfica de grandes volumes de dados em 2D	
Sofia Gameiro, Luís Almeida, Adérito Marcos	35
VIDA: Visualizador Interactivo de Dados (biomecânicos) Aumentados	
José Braz Pereira, João M. Pereira, António Veloso	101
Vídeo Digital para Multimédia: Boas Práticas de Aquisição e Processamento	
Pedro Lopes, Maria Moreira, Noel Santos	43
VRML..... porque não?	
Alexandrino Gonçalves, António Mendes	91
Web-based direct manipulation of feature models	
Rafael Bidarra, Andre van Bunnik, Willem F. Bronsvort	27

Lista de Autores

Almeida, Luís	35	Henriques, Pedro R.	173
Araújo, Bruno.....	17	Henriques, Pedro G.....	125
Barata, Nuno	111	Jorge, Joaquim	17, 149, 153, 159, 167
Bernardes, Paulo.....	63	Leitão, Miguel	131, 143
Bernardino, Alexandre.....	83	Lopes, Pedro	43
Bidarra, Rafael.....	27	Madeira, Joaquim	163
Birra, Fernando.....	53	Marcos, Adérito.....	35, 119
Bronsvoort, Willem F.	27	Marques, Filipe	163
Bunnik, Andre van	27	Martins, Manuela	63
Campos, Carlos.....	143	Matos, Nuno	119
Cardoso, Tiago	167	Matos, Paulo.....	173
Coelho, António.....	73	Mendes, António.....	91
Coelho, Pedro.....	149	Moreira, Maria	43
Correia, André	111	Morgado, Francisco.....	9
Costeira, João	83	Nande, Pedro.....	111
Cunha, Vítor	131, 143	Neto, Pedro Leão.....	137
Dias, Filipe.....	153	Pereira, João M.	101
Dias, José.....	111	Pereira, José Braz	101
Dionisio, José	83	Pereira, Pedro.....	119
Faria, Gonçalo	149	Raposo, Manuel.....	163
Farracha, Mauro	83	Sampaio, A. Zita	125
Ferreira Jr., Alfredo	159	Santos, Beatriz Sousa	163
Ferreira, F. Nunes	73	Santos, Manuel	53
Ferreira, P. Studer.....	125	Santos, Noel	43
Fonseca, Manuel.....	153, 159	Santos, Pedro	111
Gameiro, Sofia	35	Silva, Frutuoso.....	1
Garcia, Rodrigo	83	Silva, Nelson	167
Gomes, Abel.....	1, 9	Silva, Samuel	163
Gonçalves, Alexandrino	91	Sousa, A. Augusto	73
Gonçalves, Daniel	149	Veloso, António.....	101
Grave, Luís	119		