

Calculating the Stereo Pairs of a Mirror-Based Augmented Reality System

Filipe Marreiros^a

^aCentro de Computação Gráfica

R. Teixeira de Pascoais, 596, 4800-073 Guimarães
{filipe.marreiros, aderito.marcos}@ccg.pt

Adérito Marcos^{a,b}

^bDSI, Universidade do Minho

Campus de Azurém, 4800-058 Guimarães
marcos@dsi.uminho.pt

Abstract

This paper describes a solution for the calculation of the correct stereo pairs of a Mirror-Based Augmented Reality System (MBARS).

To achieve augmentation half-silvered mirrors are applied due to their dual reflection characteristic that allow simultaneous seeing-through of real objects and the viewing of reflected virtual ones. This way it is possible to merge virtual content reflected by the mirror with real objects that the user can see through it. The virtual content can be either generated by a monitor or a projector. In this paper we will only refer to the monitor-based systems. These systems are view dependent, meaning that depending upon the viewer's position the reflected images have to be recomputed. This is why a Head-Tracking System is needed to supply the current viewer's position. Also we need to consider the main system components and their location, to correctly compute the stereo pairs.

Furthermore we present how the result of the mathematical calculations can be used in combination with commands of the OpenGL API to achieve the desired transformations and to allow interactivity.

Keywords

Augmented Reality, Mirror-Based Display Systems, Stereo Pairs.

1. INTRODUCTION

The Virtual and Augmented Reality technology has been a subject of intensive research and study in the last years. Some results have already been deployed and used in end-user environments.

MBARS, in particular, are used to merge real scenes that are viewed through the mirror, together with the computer generated images, that are projected on it. The virtual content can be either generated by a monitor or a projector. In this paper we will only refer to the monitor-based systems. Using a head-tracking system, the correct images can be computed according to the observer's point of view, this way, allowing the viewer to freely interact with the system.

The systems that we are considering are stereoscopic, so it is necessary to compute the virtual images for the stereo (viewing) pair. The reason for the images to be rendered in stereo is because depth information is required in the final application scenario. When the viewer sees through the mirror he/she will need to perceive the object in its correct location, at a predefined distance from him/her.

This paper is structured as follows: Section 2 introduces the related work, while our conceptual approach is de-

scribed in section 3. In section 4 the transformations performed to the several basic system elements are demonstrated, section 5 presents the OpenGL transformations performed to achieve the correct stereo pairs and section 6 presents our application scenario. Finally section 7 includes a summary and the conclusions of the work.

2. RELATED WORK

The *de-facto's* major work developed for the subject of MBARS is related with the well-known Virtual Showcase device. This work constitutes a seminal contribution to the field, it has been carried out by a international team of researchers leaded by Oliver Bimber whose results have been used for some years now to support further development and research on MBARS (see [Bimber01], [Bimber02a], [Bimber02b], [Bimber03], [Beckhaus03]).

Furthermore the European project intituled Virtual Showcases [VirtualShowcases], also produced a large amount of papers based on the same concepts. We have been working, integrated in this project, to develop our own Showcase – the Augmented Room. For further detail on this device see [Matos03], [Pereira03], [Pereira04], [Marreiros05]. The work of our partners can also be found in [Ledermann03], [Turkish] and in the Virtual Showcases web site [VirtualShowcases].

Regarding the Mirror-Based Systems it is worth mentioning here the work of [Mulder05], which addresses the problems of interaction with the virtual objects in this type of systems. In particular to perform realistic occlusion effects.

In addition to MBARS, there is also an important related work targeting specially the calculation of Stereo Pairs. In this subject we have to refer the brilliant work of Paul Bourke [14] and [15] that approaches some concrete solutions to correctly compute stereo pairs, using OpenGL and GLUT.

Our approach integrates the some of the contributions above-mentioned however adapted for a stereoscopic application scenario where the viewer's point of view requires to be constantly computed to allow a smooth combination of real and virtual artefacts at the eyes of the end-user. Our solution can be defined as a specialization for the common MBARS.

3. OUR APPROACH

The approach used to compute the stereo pairs is based on the following observation: if we have a monitor in a known position over a mirror, then this mirror will reflect an image of the monitor according to the mirror's reflection equations. We call this projected image the "Virtual Monitor" as depicted in Figure 1, because it really does not exist, despite the viewer perceives it as being real. Since the mirror is semi-transparent, computer generated images of the "Virtual Monitor" can be combined with the real objects creating an overall *augmentation* effect.

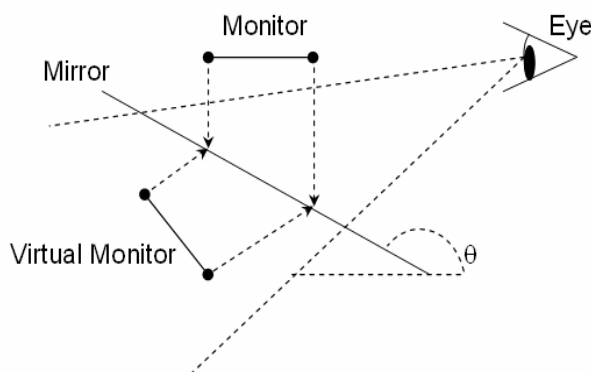


Figure 1: Mirror reflection of the Monitor

The lights play here an important role. If the scene is too much illuminated the viewer observes the virtual object as transparent. If the scene is too dark he/she will not be able to correctly view the real objects. This requires a mid-term solution where the scene is neither too much illuminated nor dark. Due to the fact that semi-transparent mirrors are being used to combine real together with virtual objects, one should be careful where to place the virtual objects. In fact, if a virtual object is placed over a real one that is too much illuminated or has a bright colour, the user will still see the real object behind the virtual one. To overcome this problem, it has

been developed a so-called "Occlusion shadows" [Bimber03], where the lights are explicitly controlled. In the spots where virtual objects are placed, the area (shadow) is simply not illuminate. However, this solution requires an extra projector only to control the lights. Since we deal here with monitor-based this approach does not satisfactorily fulfils our needs. This means that extra attention is necessary when placing the virtual objects.

To allow the viewer to perceive the virtual objects in their correct locations, the stereo images displayed in the Virtual Monitor have to be computed. However, the Virtual Monitor does not really exist; it is just an abstraction tool used to aid the understanding of the overall transformation processes. This means that the image displayed in the real monitor has to be transformed for the Virtual one. There are several steps involved in this procedure that will be described in the following sections.

4. TRANSFORMATIONS OF THE SYSTEM ELEMENTS

The system is composed by basic elements, which are: the Monitor, the Mirror and the Viewer. To be able to compute the images, the positions of these elements have to be known.

The monitor is defined by its edges points and the mirror by its plane that can be obtained using points directly measured over its surface.

To compute the virtual monitor position, the mirror reflection matrix R [Bimber01] is used. Equation 1, exemplifies how to compute the position of the virtual monitor.

$$P_{vm} = R \times P_m \quad (1)$$

To obtain the eyes position (viewer), a head tracking system is used which tracks a point in the middle of both eyes. This will introduce some error, especially if the user rotates his head to certain angles. It is also necessary to take into account the position of the tracking system, since the positions of the eyes have to be converted from the tracking system coordinate system, centred at the device, to the environment coordinate system. Also, note that the positions of the monitor and mirror have as reference the environment coordinate system. The origin of the coordinate system should be at a point viewed through the mirror, for example the at a table top where are placed real and virtual objects.

The second step involved in the process is to simulate that we are using a conventional Monitor to view the stereo images. This way we can use all the general formulation for calculating stereo pairs found in [Bourke99a] and [Bourke99b]. But this involves a series of transformations that will be discussed further ahead.

4.1 Tracking System transformations

As mentioned earlier we have to transform the eyes position that is acquired by the tracking system in its own coordinate system to the system's coordinate system. In

Figure 2 are depicted the elements that take part in this process.

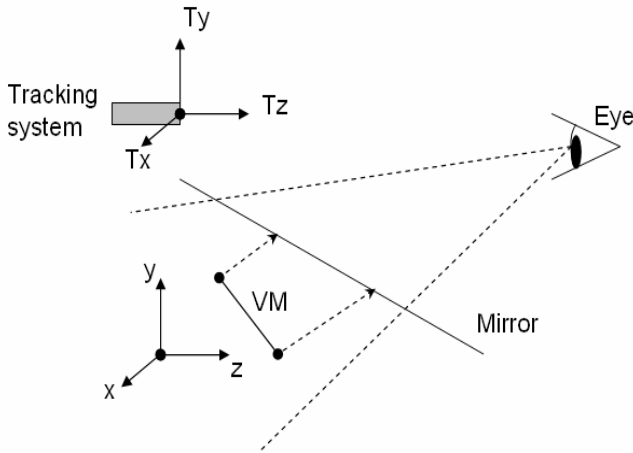


Figure 2: System's and Tracking coordinate systems

To convert a point taken in (Tx, Ty, Tz) to the (x, y, z) coordinate system we first have to translate the point in (Tx, Ty, Tz) to (x, y, z) .

Equation 2, exemplifies how to compute this transformation.

$$P_{(x,y,z)} = P_{(Tx,Ty,Tz)} + T \quad (2)$$

The T matrix is just the origin point of the tracking system, in the main system's coordinate system. Let's consider an example: the origin of the tracking system is at $(0, 10, 0)$ and the eye position taken in the tracking system coordinate system is $(0, 0, 10)$ so the eye position in the system's coordinate system is $(0, 0, 10) + (0, 10, 0) = (0, 10, 10)$.

The next steps would be to rotate the axis to align the tracking coordinate system to the main system's coordinate system. Normally to simplify this task they are already aligned. This is why we don't give further details about this subject. But one can use, if necessary, the formulation presented in further sections, because the concepts are alike.

4.2 Transformation applied to simulate a conventional monitor

As mentioned earlier we need to simulate that we are using a conventional Monitor to view the stereo images. This involves a series of transformations that will be discussed in this section. Figure 3, presents the result of these transformations.

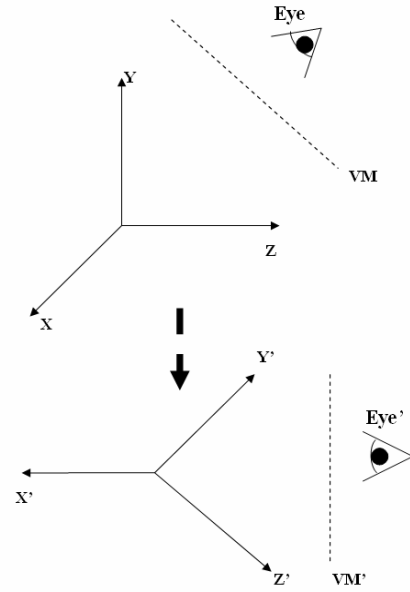


Figure 3: Coordinate System, Virtual Monitor and Eye Transformation

We can observe that several elements of the system are affected by these transformations. We will present now how these transformations are performed.

The first step of this set of operations is to obtain the centre point of the virtual monitor. Considering that we have the four edges points, Pvm_1 , Pvm_2 , Pvm_3 and Pvm_4 , computed using Eq. 1, the centre can be easily obtained using Eq. 3.

$$Vvm_{aux(1,2)} = ((Pvm_{(2,3)x} - Pvm_{1x}) / 2, \\ (Pvm_{(2,3)y} - Pvm_{1y}) / 2, \\ (Pvm_{(2,3)z} - Pvm_{1z}) / 2)$$

$$Pvm_C = Pvm_1 + Vvm_{aux1} + Vvm_{aux2} \quad (3)$$

Then, the translation to the origin is calculated. The translation matrix is given in 4.

$$TM = -Pvm_C \quad (4)$$

The rotations necessary for the virtual monitor plane to be normal to Z axis are obtained performing a set of rotations. The following figures and equations will present step by step these transformations.

The first step is to rotate the virtual monitor that is now centred at the origin, so that the projection of the vector $Vvmx$ on the XZ plane coincides with the X axis. Figure 4 depicts this transformation and the equations are presented in 5.

Y axis rotation:

$$\begin{aligned} \text{If } (Vvmx_z > 0) \quad & \theta = \arccos\left(\frac{Vvmx_x}{D1}\right) \\ \text{If } (Vvmx_z < 0) \quad & \theta = -\arccos\left(\frac{Vvmx_z}{D1}\right) \\ D1 = & \sqrt{(Vvmx_x)^2 + (Vvmx_z)^2} \end{aligned} \quad (5)$$

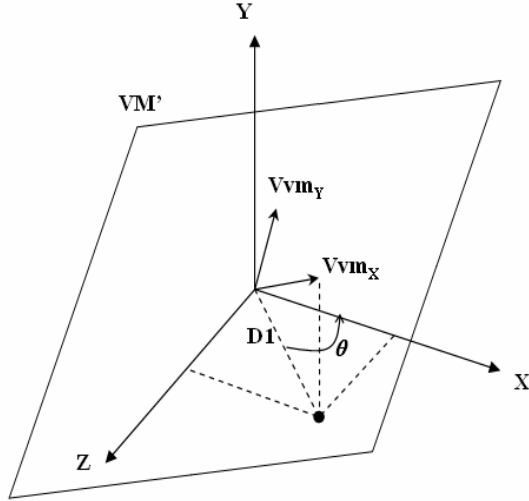


Figure 4: Initial Rotation for the virtual monitor to be normal to the Z axis

The second rotation will then transform vector $V'vmx$ so that it coincides with the X axis, like presented in figure 5 and using Eq. 6.

Z axis rotation:

$$\begin{aligned} \text{If } (V'vmx_y > 0) \quad & \Phi = \arccos\left(\frac{V'vmx_x}{D2}\right) \\ \text{If } (V'vmx_y < 0) \quad & \Phi = -\arccos\left(\frac{V'vmx_z}{D2}\right) \\ D2 = & \sqrt{(V'vmx_x)^2 + (V'vmx_z)^2} \end{aligned} \quad (6)$$

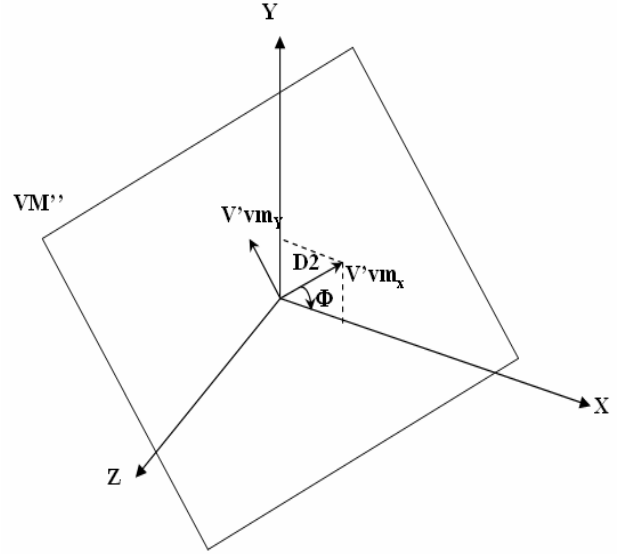


Figure 5: Second Rotation for the virtual monitor to be normal to the Z axis

Finally the last rotation will place $V''vmx$ and $V''vmy$ in the respective axis the X and Y, using Eq. 7. Then we have the transformed virtual monitor normal to the Z axis, like depicted in figure 6.

X axis rotation:

$\text{If } (V''vmy_z \geq 0) \quad \alpha = -\arccos\left(\frac{V''vmy_y}{D3}\right)$	(7)
$\text{If } (V''vmy_z < 0) \quad \alpha = \arccos\left(\frac{V''vmy_y}{D3}\right)$	
$D3 = \sqrt{(V''vmy_z)^2 + (V''vmy_y)^2}$	

One has to take in attention that these transformations are applied not only to the virtual monitor but also to the main system's coordinate system and to the eye. This way we guarantee that the image viewed in the cases presented in figure 3 are always the same.

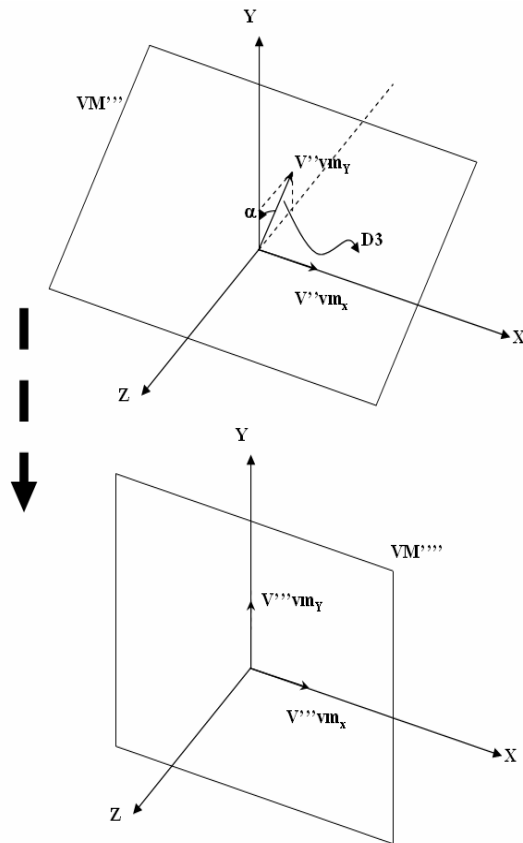


Figure 6: Final Rotation for the virtual monitor to be normal to the Z axis

Having the angles of the rotations and the translation displacement, we can apply them to the several elements. The virtual monitor only needs to be transformed once because its position is a fixed one. The eyes positions need to be computed each time because they can change their position. The operations that must be performed are exemplified on the following pseudo-code:

```
//performed only at the initialization
Pvm[1..4] = Pvm[1..4] - Pvm_c;
Pvm[1..4] = RotationY (Pvm[1..4],  $\theta$ );
Pvm[1..4] = RotationZ (Pvm[1..4],  $-\Phi$ );
Pvm[1..4] = RotationX (Pvm[1..4],  $\alpha$ );
```

```
//performed every time
//Left Eye
EyeLeft = EyeLeft - Pvm_c;
EyeLeft = RotationY (EyeLeft,  $\theta$ );
EyeLeft = RotationZ (EyeLeft,  $-\Phi$ );
EyeLeft = RotationX (EyeLeft,  $\alpha$ );
```

```
//Right Eye
EyeRight = EyeRight - Pvm_c;
EyeRight = RotationY (EyeRight,  $\theta$ );
EyeRight = RotationZ (EyeRight,  $-\Phi$ );
EyeRight = RotationX (EyeRight,  $\alpha$ );
```

Since the coordinate system is also changed the virtual objects will also be affected by these transformations.

```
//performed only at the initialization
//Left Eye
VO = VO - Pvm_c;
VO = RotationY (VO,  $\theta$ );
VO = RotationZ (VO,  $-\Phi$ );
VO = RotationX (VO,  $\alpha$ );
```

Because we didn't want to apply these transformations to the object every time, we just transform them at the beginning and use OpenGL commands for updating the positions if they change.

5. OPENGL TRANSFORMATIONS

Now that we have all the necessary information to create the stereo pairs, we just need to apply the OpenGL commands.

The correct way to create stereo pairs is to use an Off-axis projection. It requires a non symmetric camera frustum; fortunately this is supported by OpenGL.

In [Bourke99a] the author demonstrates that this is effectively the correct way to compute the stereo pairs by giving picture examples that prove it. In [Bourke99b] the same author presents how this can be achieved with OpenGL and GLUT. Our work is based on the referred author's work, but applied to our particular case.

5.1 Calculating the OpenGL Model View Transformations

The OpenGL model view matrix, is easily created knowing the transformations presented in section 4. For each eye we will have the following pseudo-code:

```
//Right Eye
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt (EyeRight.x, EyeRight.y,
           EyeRight.z, EyeRight.x,
           EyeRight.y, 0, 0, 1.0, 0);
glRotated( $\theta$ , 0, 1, 0);
glRotated( $-\Phi$ , 0, 0, 1);
glRotated( $\alpha$ , 1, 0, 0);
glTranslated(-Pvm_c.x, -Pvm_c.y, -Pvm_c.z);
```

```
//Left Eye
glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
gluLookAt (EyeLeft.x, EyeLeft.y,
           EyeLeft.z, EyeLeft.x,
           EyeLeft.y, 0, 0, 1.0, 0);

glRotated( $\theta$ , 0, 1, 0);
glRotated( $-\Phi$ , 0, 0, 1);
glRotated( $\alpha$ , 1, 0, 0);
glTranslated(-PvmC.x, -PvmC.y, -PvmC.z);
```

One has to be careful with the order of the transformations; these should be supplied as they are.

5.2 Calculating the OpenGL Projection Transformations

The final step of this process is to obtain the OpenGL Projection Matrix. The pseudo-code used in this case is:

```
//Right Eye
glDrawBuffer(GL_BACK_RIGHT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(Pvm1.x-EyeRight.x,
          Pvm2.x-EyeRight.x,
          Pvm3.y-EyeRight.y,
          Pvm1.y-EyeRight.y,
          EyeRight.z, 100000.0);

//Left Eye
glDrawBuffer(GL_BACK_LEFT);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
glFrustum(Pvm1.x-EyeLeft.x,
          Pvm2.x-EyeLeft.x,
          Pvm3.y-EyeLeft.y,
          Pvm1.y-EyeLeft.y,
          EyeLeft.z, 100000.0);
```

5.3 The resulting camera Frustum

The last operations will produce the camera frustums for both eyes (the stereo pairs). Figure 7 will be used to better exemplify the result of these operations and to verify that we really have a non symmetric camera frustum.

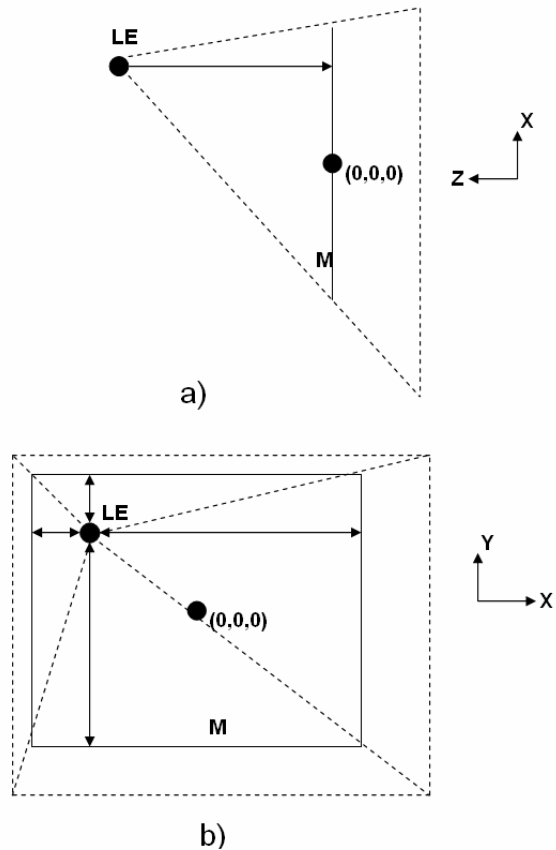


Figure 7: Camera Frustum a) Top View b) Front View

In Figure 7 we only present the camera frustum for the left eye but the same considerations apply to the right eye. The top view clearly shows us that the camera is non symmetric. This is because the `gluLookAt` forces it to be. Let's analyze again the parameters of the `gluLookAt`. The first 2 parameters are the most important they are the Eye and Center. In this case the Eye is (LE_x, LE_y, LE_z) and the Center (LE_x, LE_y, 0). Only the Z changes between them, this forces the camera to point in a straight line towards Z=0, this is showed by the arrow in a). In b) we can see the distances between the transformed eye and the virtual monitor; they are used by the `glFrustum`. The `glFrustum` command requires also the near and far plane. We put the far plane at a position very far away. The nearest plane has to be placed at the transformed virtual monitor; its value will be LE_z because this is the distance between the virtual monitor and the eye.

6. APPLICATION SCENARIO

The concepts presented here were used to develop an application scenario for Museum D. Diogo de Sousa in Braga, Portugal. As mentioned earlier we have been working, integrated in the Virtual Showcases project, and developed our own Showcase – the Augmented Room, which stands at the referred museum. For further details see [Matos03], [Pereira03], [Pereira04], [Marreiros05].

In order to give an overview of the system we present images that help to understand the work developed.

The scenario consist of an Ancient Roman Tomb were virtual archaeological artefacts are placed over it. Figure 8 depicts the real component of the scenario, a 1:2 model of the remainders found of the Tomb.



Figure 8: Real component of the scenario

To exemplify how the augmented environment is displayed, we use the montage present in figure 9.



Figure 9: Augmented scenario

It is important to notice that the user will not view the environment exactly like it is presented in figure 9. Notice that the position of the objects is not accurate. We made this illustration due to the fact that; to capture stereo images special hardware is required, which is not available.

The augmented environment is located inside the structure of the Showcase. Figure 10 presents our showcase- the Augmented Room.



Figure 10: Augmented Room

The user can look through the showcase windows and see the augmented environment. One of these windows is presented in figure 11.



Figure 11: Viewer's window

In figure 11, we can also see the joysticks used for interaction with the virtual content, the shutter glasses and a part of a stereo virtual image used to explain the joysticks usage.

Due to the fact that Museu D. Diogo de Sousa is still closed to the public, further tests like usability tests couldn't be made. But we plan to do so as soon as the showcase is in use by the general public [Marreiros05].

7. SUMMARY AND CONCLUSIONS

We have presented in detail the several steps involved in the calculation of the Stereo Pairs of a Mirror-Based Augmented Reality System.

We started by introducing the specific transformations used in a MBARS, that uses Monitors for image generation. These transformations were used in combination with OpenGL commands to correctly obtain the Stereo Pairs.

The result of this work was tested during our work in the Virtual Showcase project, proving the concepts presented in this paper.

8. ACKNOWLEDGEMENTS

We would like to acknowledge Oliver Bimber and Paul Bourke continuous support during the development of this work.

This work was partially supported by the European Union through the project n. IST-2001-28610.

9. REFERENCES

- [Bimber01] Bimber, O., Fröhlich, B., Schmalstieg, D., and Encarnação, L.M.. The Virtual Showcase, *IEEE Computer Graphics & Applications*, vol. 21, no. 6, pp. 48-55, 2001.
- [Bimber02a] Bimber, O., Fröhlich, B.. Occlusion Shadows: Using Projected Light to Generate Realistic Occlusion Effects for View-Dependent Optical See-Through Displays, *Proceedings of International Symposium on Mixed and Augmented Reality (ISMAR'02)*, 2002.
- [Bimber02b] 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.
- [Bimber03] Bimber, O., Encarnação, L.M. and Schmalstieg, D.. The Virtual Showcase as a new Platform for Augmented Reality Digital Storytelling⁹, *Eurographics Workshop on Virtual Environments*, The Eurographics Association, 2003.
- [Beckhaus03] Beckhaus, S., Ledermann, F., Bimber, O.. Storytelling and Content presentation with Virtual and Augmented Environments in a Museum Context, *CIDOC 2003*, September 2003, Sankt Petersburg, Russia.
- [Ledermann03] Ledermann, F., Schmalstieg, D.. Presenting an Archaeological Site in the Virtual Showcase, *Proc. of the 4th International Symposium on Virtual Reality, Archaeology, and Intelligent Cultural Heritage (VAST 2003)*, pp. 119-126, Brighton, UK, Nov. 2003. Available as technical report TR-188-2-2003-17, Vienna University of Technology.
- [Matos03] Matos, N., Pereira, P., Grave, L., Marcos, A.. ARK: Augmented Reality Kiosk, *In: Human-Computer Interaction – Theory and Practice (PART II)* Edited by Constantine Stephanidis, Julie Jacko, pp. 168-172. (Vol.2 of the Proceedings of HCI International 2003, 22-27 June, Crete, Greece). Lawrence Erlbaum Associates Publishers Mahwah, New Jersey, London, ISBN: 0-8058-4931-9.
- [Pereira03] Pereira, P., Matos, N., Grave, L., Marcos, A., ARK multi-utilizador, *12º EPCG*, Porto, Portugal, October 8-10 2003, pp. 119-124.
- [Pereira04] Pereira, P., Matos, N., Grave, L., Marcos, A.. Augmented room: a device for visualization of museum artefacts, *In: Proc. of 1st International Conference Virtual Design and Automation*, Poznan, 3-4 Jun. 2004, Poland (published in CD-ROM – ISBN: 972-98872-2-5).
- [Turkish] The Demonstrator Scenario “Turkish Chess-Player”.
<<http://www.ims.tuwien.ac.at/~flo/vs/chessplayer.html>>
- [VirtualShowcases] The Virtual Showcases Consortium
<<http://www.virtualshowcases.com>>
- [Marreiros05] Marreiros, F., Figueiredo, P., Pereira, P., Matos, N., Marcos, A.. An augmented reality showcase to support a cultural heritage scenario, *In: Proc. of Training, Education & Simulation International (TESI) 2005*, Maastricht, 22-24 March 2005, Netherlands (published in CD-ROM).
- [Mulder05] Mulder, J.. Realistic Occlusion Effects in Mirror-Based Co-Located Augmented Reality Systems, *In: VR 2005*, Bonn, 12-16 March 2005, Germany.
- [Bourke99a] Calculating Stereo Pairs
<<http://astronomy.swin.edu.au/~pbourke/stereographics/stereorender/>>
- [Bourke99b] 3D StereoRendering Using OpenGL (and GLUT)
<<http://astronomy.swin.edu.au/~pbourke/opengl/stereogl/>>