Chapter 3

Mathematical description to the display of Virtual Environments

3. Mathematical description to the display of Virtual Environments

Currently there are mayor developments in visualization techniques. This has allowed us to experience different virtual environment on multiple screens. For instance, visualization observatories (Lucet, 2006), Panoramic Display Systems (Simon, et al., 2002), different CAVEs (Cruz-Neira, et al., 1993), (Ihrén, et al., 1999), (Jalkanen, 2000), etc. Therefore, some papers have shown different methods for displaying a virtual world using several screens (Cruz-Neira, et al., 1993), (Jalkanen, 2000), (Granados, 2006), (Pape, et al., 1978.). However, we have found the following:

- There are a few details about the characteristics involved in this task.
- We have found no references with general information about how to build virtual environments.
- We have found no details that help us to understand the work behind of the displaying on multiple screens, specifically the work of the view frustum, to be described in this chapter.

In this chapter, there is a description about frustum and the required formulae for building a multi-screen environment. The result of this research is published in the 1st International Conference on Computer Science -CICOMP 2006 (Mora, et al., 2006).

This chapter is organized in the following way:

Section3.1 contains a description about frustum

Section 3.2 describes the required formulae for building a virtual environment.

Section 3.3 presents how to compute the frustum parameters

Section 3.4 contains a system, which is programmed using the methodology presented in this chapter.

Finally, in section 3.5 there is a chapter summary.

3.1 Frustum

A *view volume* created by a perspective projection is called *frustum*. This view volume is also called *view frustum* or *the viewing frustum*, see Figure 3.1. *View frustum* determines the region of a virtual world that is going to be displayed on a screen (Jones, 2005). It can be defined by the *near, far, left, right, top, and bottom* distances between the screen and the viewer position. *Near* and *far* are the planes that cut the frustum and are perpendicular to the *Z* axis. The *projection center* is the pyramid apex and a projection plane is parallel plane to the pyramid base. The dotted line represents the *vision line*, also called *frustum axis*, if this line is perpendicular to the screen center and is spread on *Z* axis the frustum is symmetric, in any other case is asymmetric.

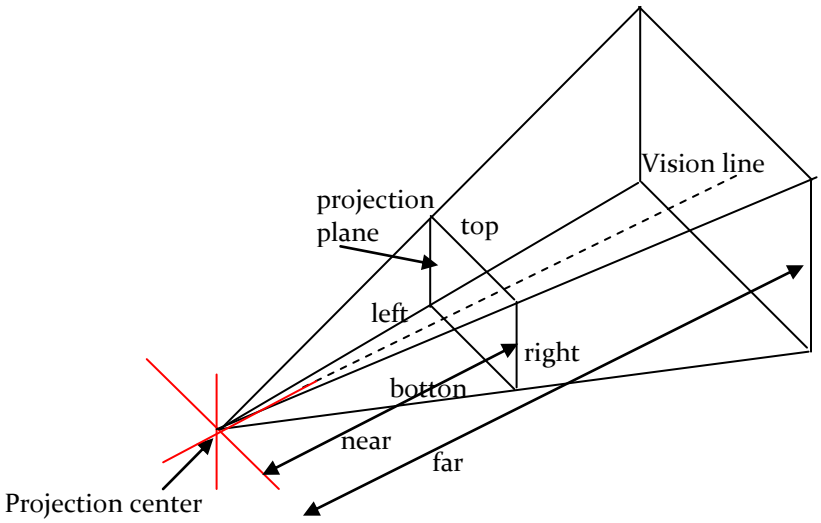


Figure 3.1 Frustum

In a virtual environment, it is necessary to determine the viewer's and the screens's positions. The viewer position is located in the pyramid apex. A screen is a projection plane and it can be seen as an analogy of a glass window that can be seen through. The viewer position is captured with a head-tracker device and the volume of the frustum constantly changes according to the viewer position. A virtual environment which uses this technology is dynamic. When the viewer position is taken into consideration, only one viewer can be tracked at the same time.

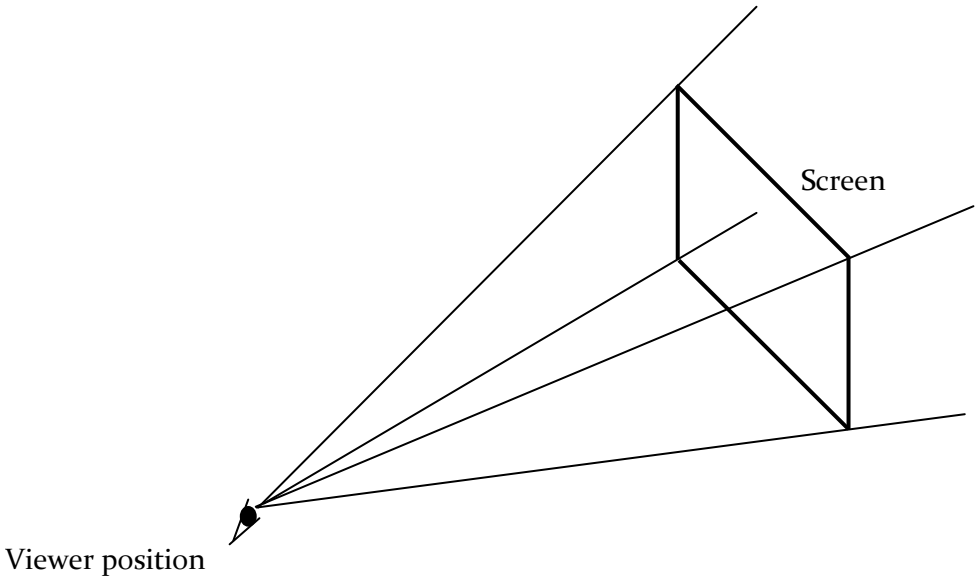


Figure 3.2 Viewing Frustum

If three men watching a virtual world through one screen could be seen, it would look like they are seeing the same, but this is a mistake, because when the viewer position is considered the viewing frustum is computed using this position, therefore only one man would be seeing the world correctly, while the others are witnesses to see a deformed world. Figure 3.3 shows a man placed in three different positions at three different moments the

man would be seeing the image correctly in each position. The viewer would be watching a simple world which contains just two spheres. Although it seems the user is watching the spheres in the same way, in each position the worlds look different.

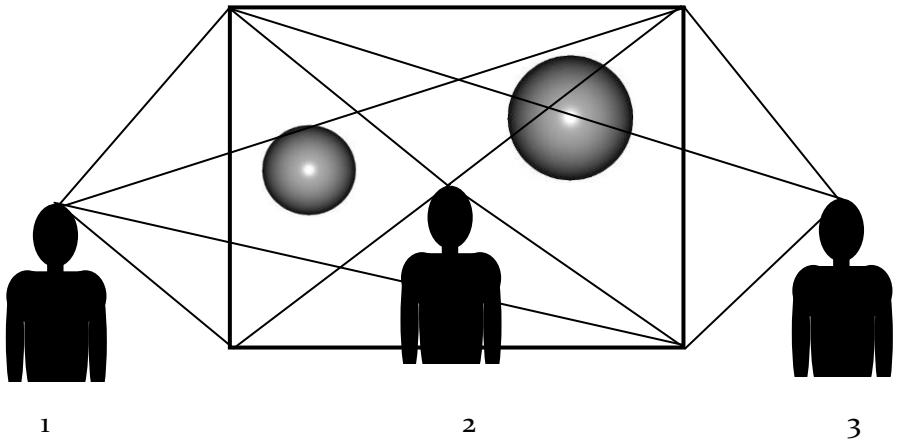


Figure 3.3 Three men watching two spheres

Figure 3.4 shows the same example shown in Figure 3.3, but this one seen from above. When the man is placed to the left (1), he can see the small sphere incomplete, and the other one can see it without any problem. If the man is placed in the middle (2) can see both complete spheres. Finally, when the man is placed to the right (3) can see the small sphere complete and the other one truncated.

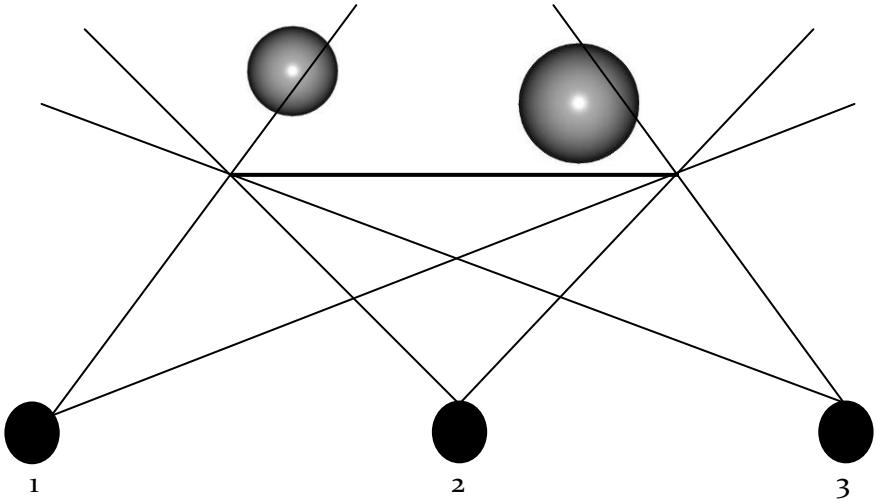


Figure 3.4 Three different viewer position seen since above

Although the distances among the viewer positions are minimal, the view volume is different. Figure 3.5 shows a world seen from twelve viewer positions, even if the distances among the nearby viewers change a little, the viewers will not see the same. This can be seen in Figure 3.6.

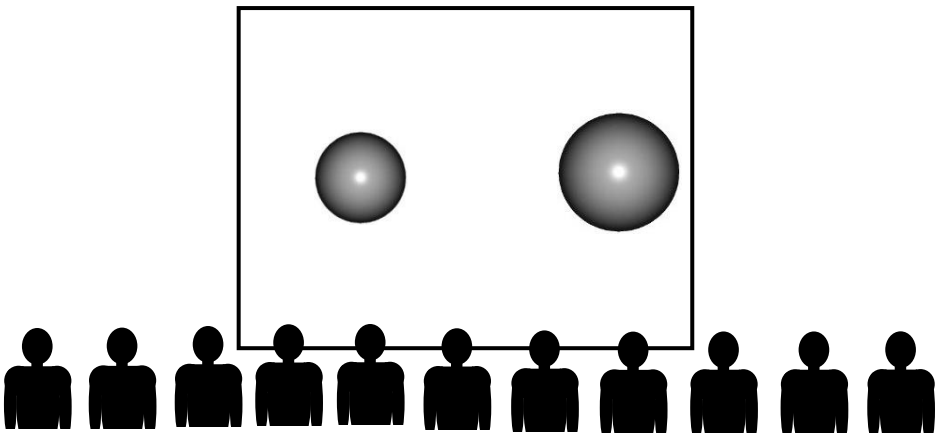


Figure 3.5 A virtual environment seen from eleven viewer positions

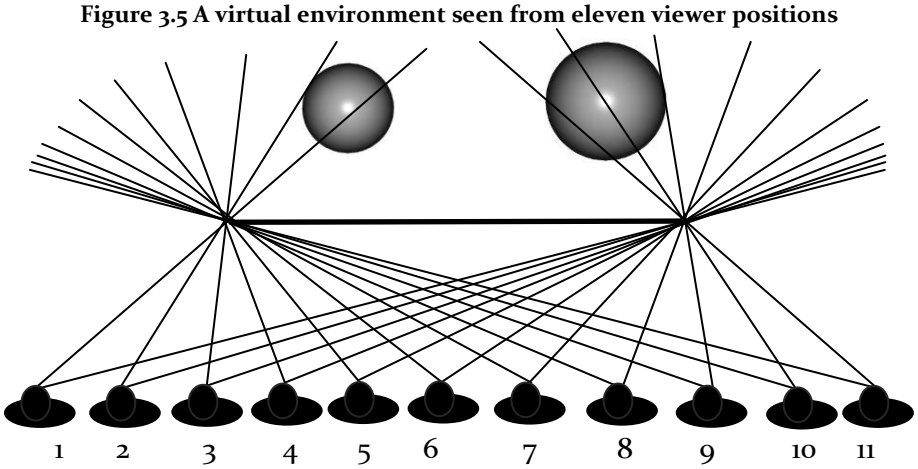


Figure 3.6 Show the eleven frustum generated by eleven viewer positions seen from above

Therefore, when a specific viewer position is linked with a different *viewing frustum*, the viewer sees the incorrect information, although the *frustum* only differs some millimeters. In Figure 3.7 there are two viewer positions and two *viewing frustu*ms, if the viewer positions were exchanged, the information shown would be wrong. Therefore, when a virtual environment changes constantly and the frustum is incorrect, the environment looks unreal.

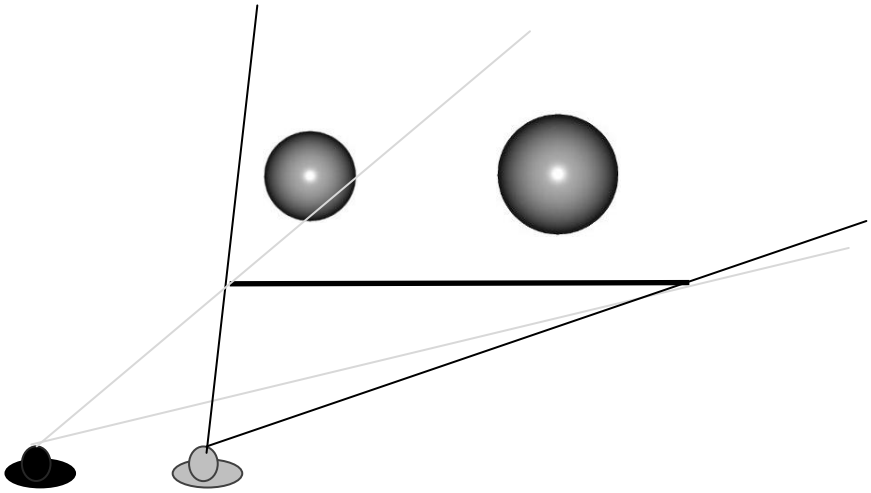


Figure 3.7 Two viewer position and two frustums

When a virtual environment is built with *n* screens, it will have *n* viewing *frustums*. Then the virtual environment is going to be divided into regions, which are displayed on individual screens. The projection center for all screens is the viewer position. Figure 3.8 shows an environment built with four screens, this environment is seen by a user, and his position is the projection center for every involved *viewing frustum*.

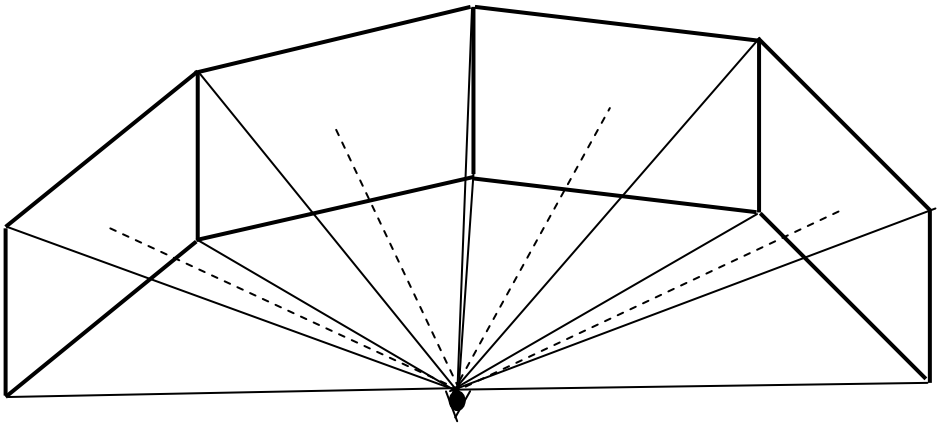


Figure 3.8 A virtual environment built with four screens

In the following figures there are three designs of virtual environments. These designs are very simple, but it is possible to distinguish the frustums from each screen.

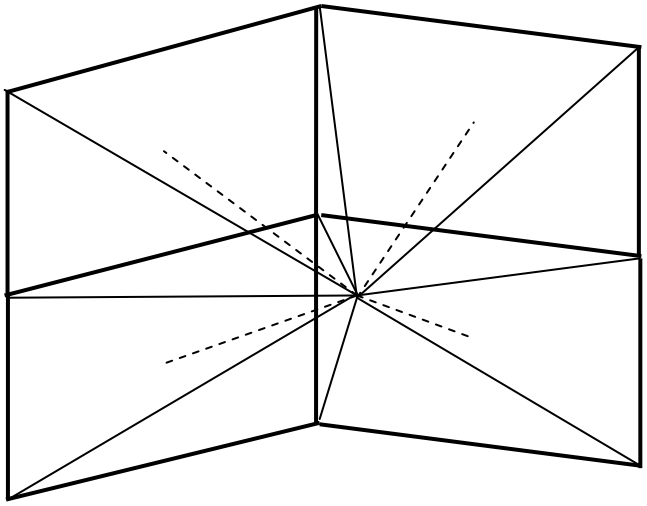


Figure 3.9 An environment built with four screens

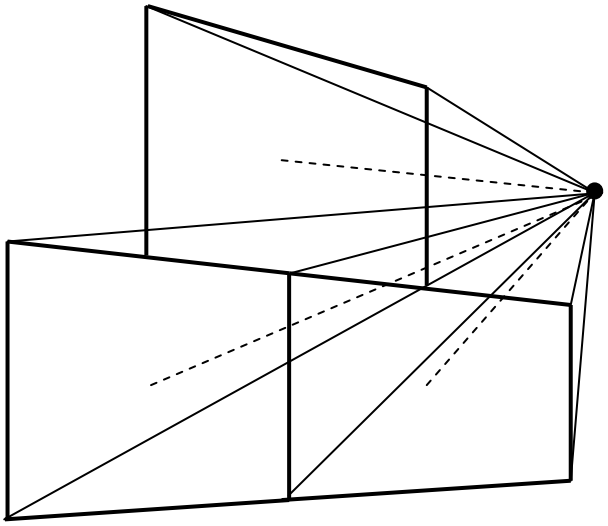


Figure 3.10 An environment built with three screens

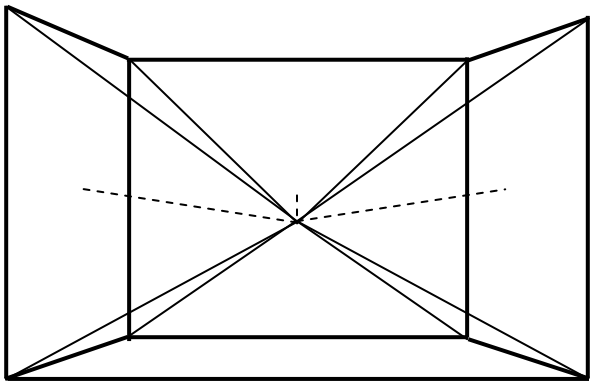
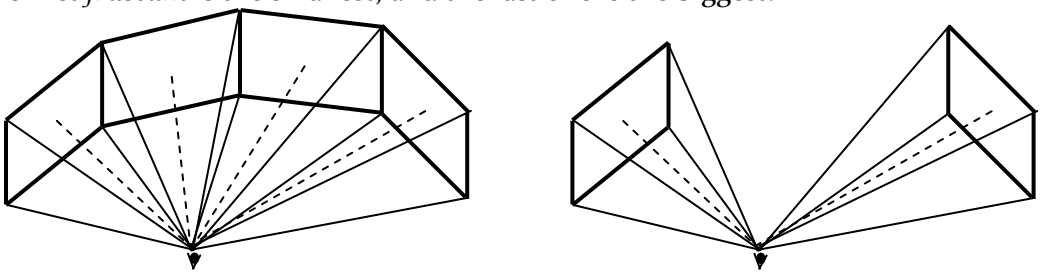


Figure 3.11 An environment built with four projections

For preserving the coherence among screens it is necessary that every *frustum* is correctly computed. Figure 3.12.1 show the same virtual environment shown in Figure 3.8, but the viewer is placed to the left with respect to the environment center. The *frustum sizes* are different. In Figure 3.12.2 two *frustums* of the environment of Figure 3.12.1 are show, the first and the fourth. Due to the fact that the viewer is nearer to the first screen than the others, the first *frustum* is the smallest, and the last one is the biggest.



1 2

Figure 3.12 1) A virtual environment built with four screens. 2) The first and the fourth frustums of the environment shown in Figure 3.8

On dividing a virtual world in regions, some objects of this will be divided in two or more screens. Furthermore, some of them are going to be visible and others are not.

Figure 3.13 shows a set of objects, which, due to specific viewer position can be seen in the following way:

-The small sphere is partially visible on the left screen

-The medium sphere is displayed on the same left screen, but it is completely displayed.

-The big cube is shown on the two screens in the middle.

-The big sphere is displayed on the two right screens.

-Finally the small cube is seen on the right screen.

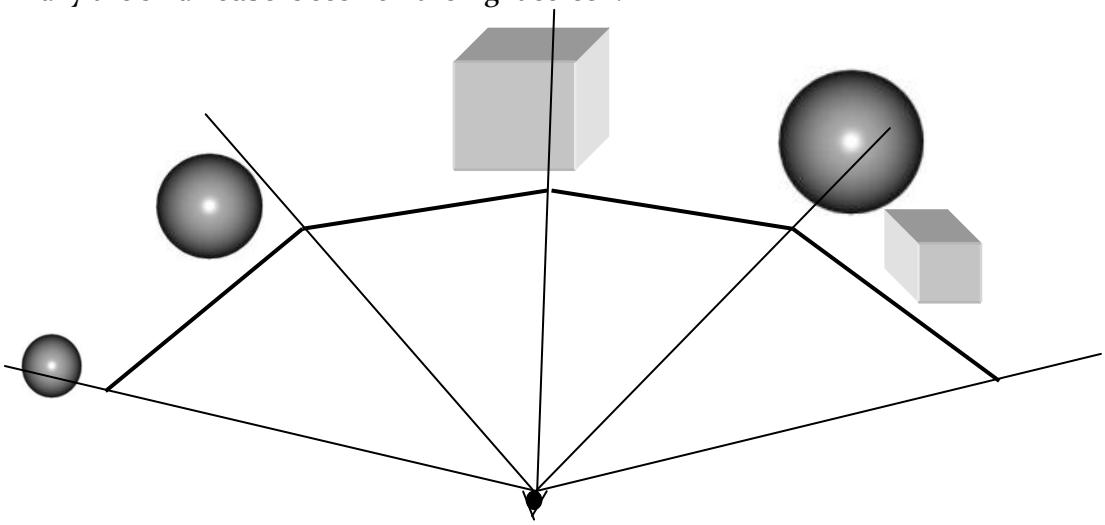


Figure 3.13 Five objects displayed on four screens

On changing the viewer position, the objects will be displayed in different ways. Some of them may be displayed on other screen(s), some of them may not be visible any longer.

If the same virtual environment shown in Figure 3.13 were used, on moving the viewer position towards the left (see Figure 3.14), the objects contained in the environment will be displayed in the following way:

-The small sphere that was visible for the viewer in Figure 3.13, in the new viewer position is not visible.

-The medium sphere is displayed on the same screen than in Figure 3.13.

-The big cube that was shown before in two central screens is now shown in the second screen.

-The big sphere that in Figure 3.13 was displayed on the last two screens, now it is displayed on the third screen.

-Finally, the small cube is displayed on the same screen like in the previous example. See Figure 3.14.

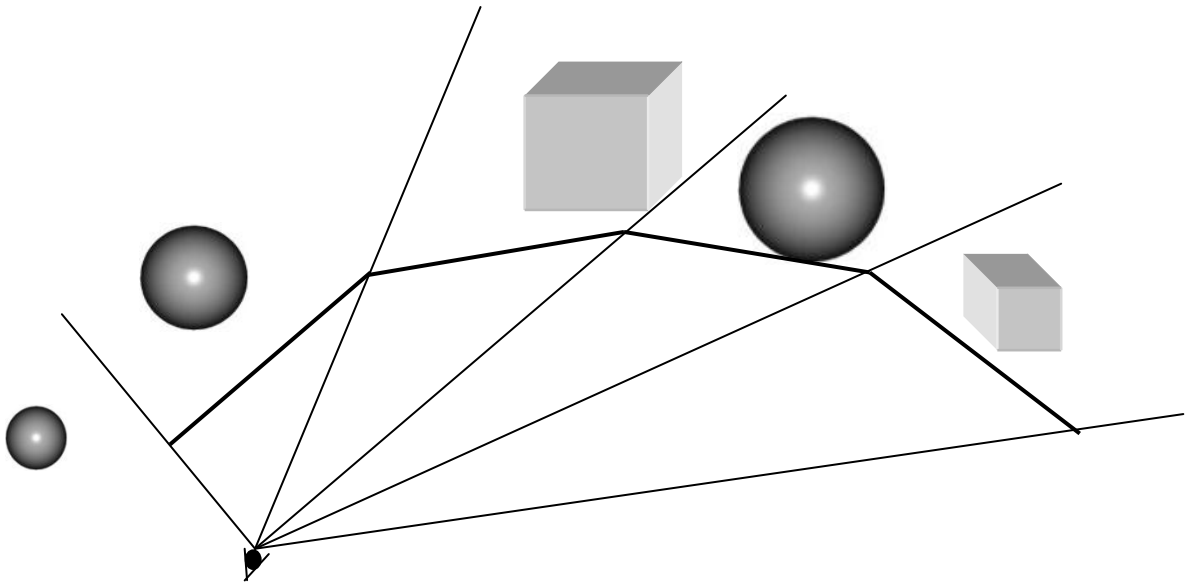


Figure 3.14 Five objects displayed on four screens, the viewer is aligned to the left

When the frustums are incorrectly computed and the virtual environment is built with *n* screens, the information does not look continuous, on some occasions the objects can look wider, thinner or even folded. Therefore, it is necessary to synchronize all measures involved in the environment; take into consideration the screen frames and only one viewer position

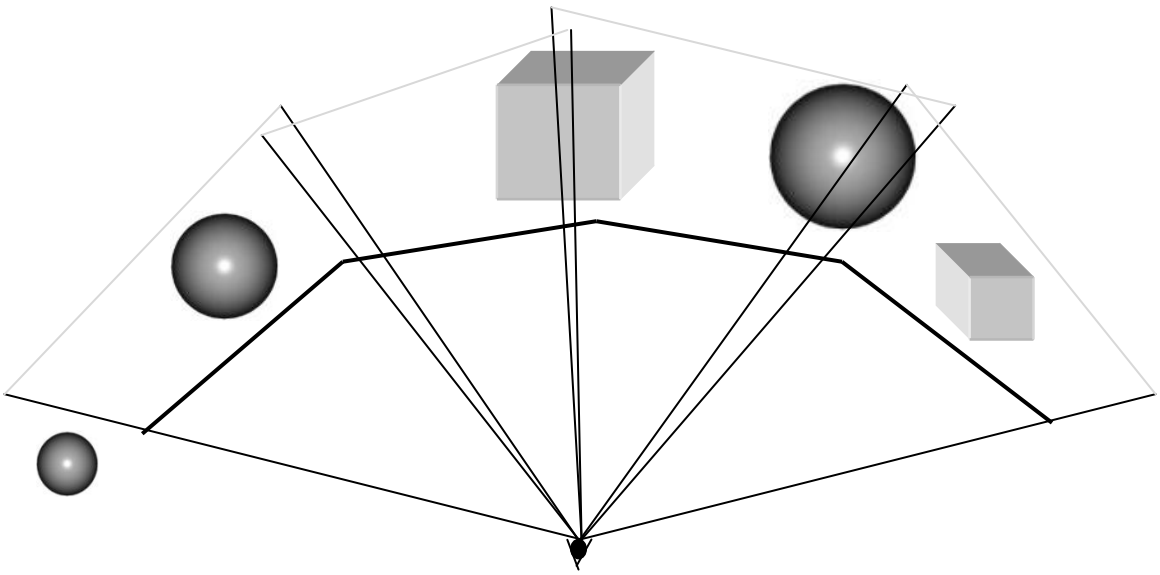


Figure 3.15 incorrect Frustums

3.2 Real-World Screen Position

This document shows some general formulae for building a virtual environment. These formulae allow for the construction of a virtual environment with one or *n* screens and/or projections. The screens used in an environment have to be oriented so that the viewer could see part of a world through them. This section explains how to build environments with screens side by side. Every screen has to be identified when an environment is built.

One way of identifying them is using its corners. The four screen corners are used as means of identification; these corners furthermore allow us to know the size and the orientation of every screen. The corners are labeled as: Upper Left (*UL*), Upper Right (*UR*), Lower Left (*LL*) and Lower Right (*LR*), see Figure 3.16.

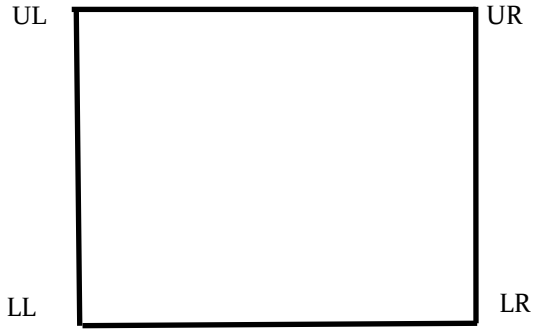


Figure 3.16 Labeling the screens corners

3.2.1 Side by Side Screen

In order to calculate the coordinates of a set of screens placed side by side, in this work the use of the corner *UL* of each screen is proposed. The origin will be in the middle of the environment. The *UL* of the first screen placed in the environment will be on the coordinate (x_1,y_1,z_1). This corner will be used as basis for calculating the others *UL*'s. w_1 represents the width of this screen; and σ represents its angle formed on the x axis. (x_2,y_2,z_2) is corner *UL* of the next screen. See Figure 3.17.

Eq. 3.1 $$x_2 = x_1 + w_1 \cos\sigma_1$$

Eq. 3.2. $$y_2 = y_1$$

Eq. 3.3. $$z_2 = z_1 + w_1 \sin\sigma_1$$

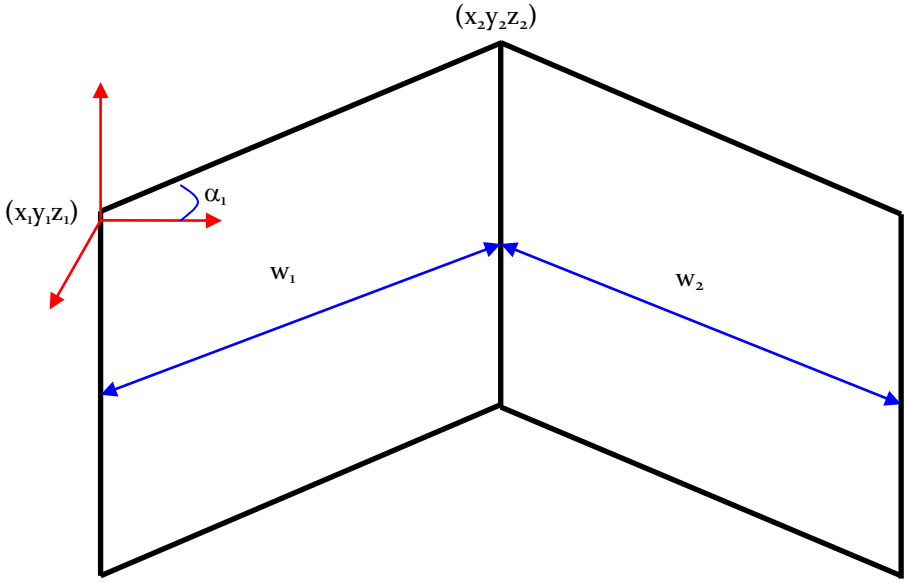


Figure 3.17 A real point of reference in a environment built with two screens

Eq. 3.1, Eq. 3.2, and Eq. 3.3 are generalized to build an environment with *n* screens, in which, each screen have its own width *w*, and its own angle σ.

Eq. 3.4. $$x_{i+1} = x_i + w_i \cos\sigma_i$$

Eq. 3.5. $$y_{i+1} = y_i$$

Eq. 3.6. $$z_{i+1} = z_i + w_i \sin\sigma_i$$

Where: *n* is the number of screens and $1 \leq i < n$

In Figure 3.18 there is an environment built with several screens, although in this figure the width of the screens is the same, each one could have its own width (*w*) and angle σ.

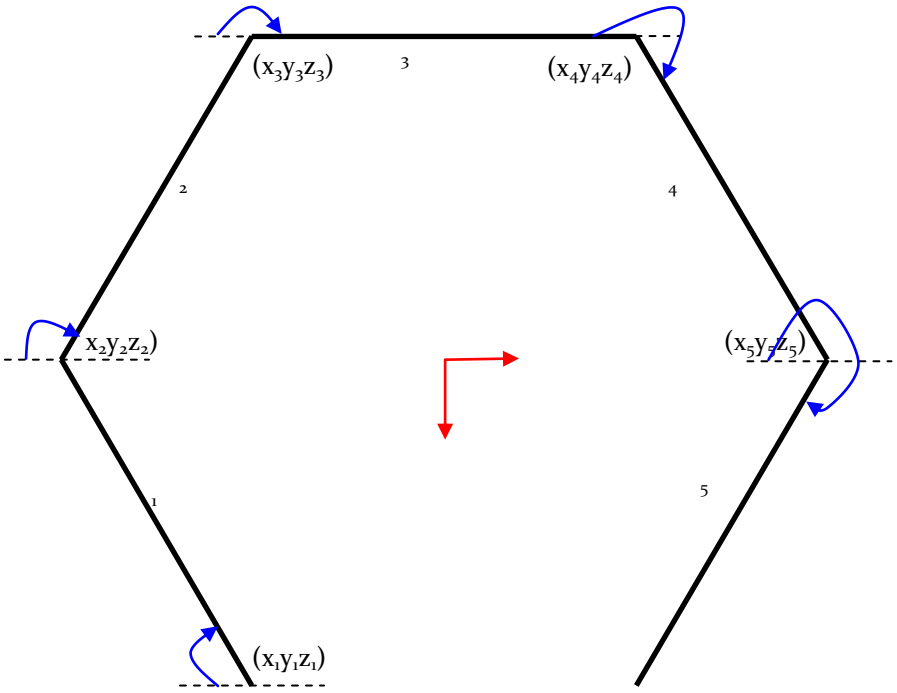


Figure 3.18 Environment built with five screens

Due to the monitors having a frame that takes up some space, it is necessary to considerate it, because the *frustums* will have to be exact. In this way, the frames only appear to be simple obstructions to the environment. Eq. 3.4, Eq. 3.5, and Eq. 3.6 have a modification that is the sum of the width of the frames of two continuous screens S_i.

Eq. 3.7 $$x_{i+1} = x_i + (w_i + s_i)\cos\sigma_i$$

Eq. 3.8. $$y_{i+1} = y_i$$

Eq. 3.9. $$z_{i+1} = z_i + (w_i + s_i)\sin\sigma_i$$

Although the mobile environments are beyond the scope of this work we can propose an interesting idea which could be useful for some future works. We could build sophisticated mobile environments using a kind of robotic arm as support of a set of *n* screens. The idea is that this arm could move in different directions either previously programmed or manually

done. Therefore the formulas of the direct kinematic could be useful in the mobile environments, because they describe the posture of a robotic arm through its joints.

The equations Eq. 3.10, Eq. 3.11, and **Eq. 3.12** allow for the calculation of every joint. Figure 3.19 and Figure 3.20 help to illustrate the idea. The first joint will be placed on the coordinate (x_1,y_1,z_1), which will be the basis to calculate the other joints, s will have to be the sum of the width of the frames of two contiguous screens plus the space generated in the corresponding joint. The number of joints will depend on the number of screens involved in the environment.

Eq. 3.10 $$x_{i+1} = x_i + (w_i + s_i)\cos(\sigma_1 + ... + \sigma_i)$$

Eq. 3.11. $$y_{i+1} = y_i$$

Eq. 3.12. $$z_{i+1} = z_i + (w_i + s_i)\sin(\sigma_1 + ... + \sigma_i)$$

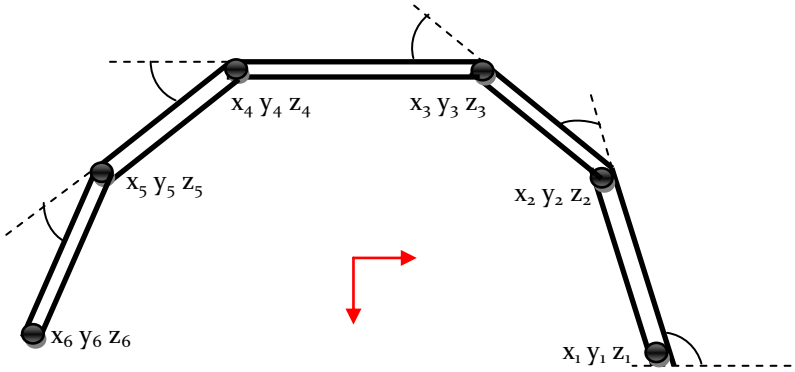


Figure 3.19. Robotic arm as support of virtual environment

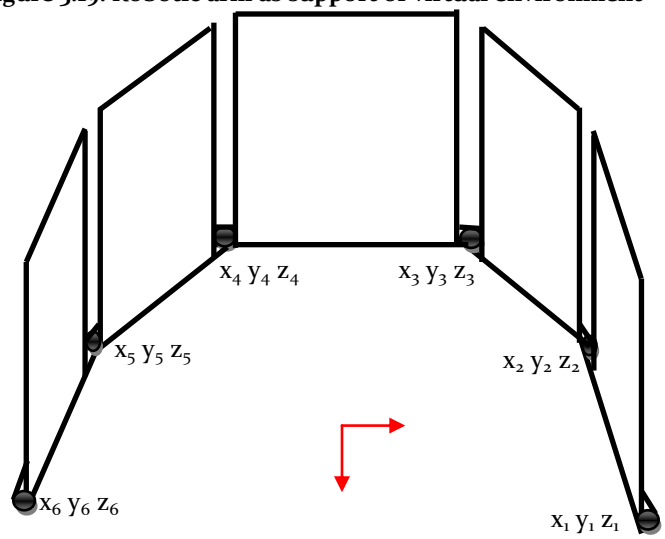


Figure 3.20 A mobile environment

Figure 3.21 shows the simulation of a movement carried out by the robotic arm. The simulated movement was done in the joint $x_4y_4z_4$, all other joints will remain with the same angle.

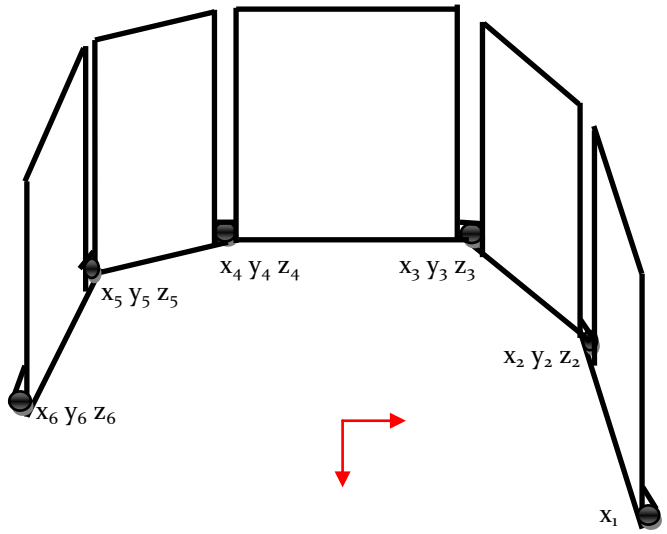


Figure 3.21 Representing a movement in the robotic arm

3.2.2 Multiple non aligned Screens

When the screens are placed without being aligned, a structure that supports a set of screens can be used. Then, this structure could already have the places of the screens established, which allows us to reduce the calculus of the coordinates of every screen. Figure 3.22 shows an environment built with six screens, which are assembled in a specific position.

In this kind of projects, the structure of the environments can change in aspects such as: shape, number of screens, aligning, angles among screens, etc. The monitors allow building sophisticated environment, depending on the resources and the applications.

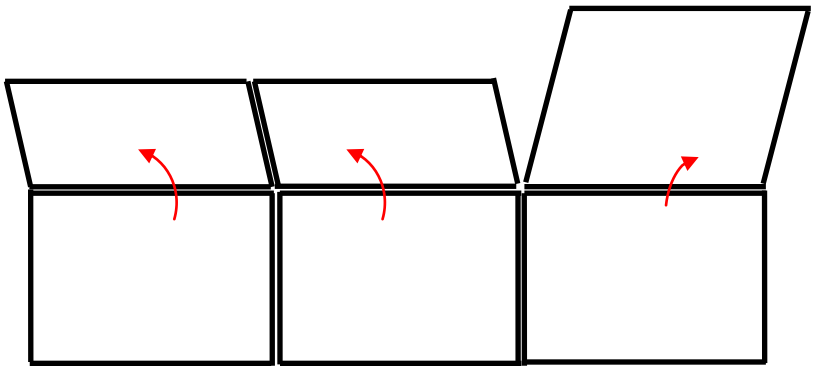


Figure 3.22. An environment built with six screens on a fixed structure

Figure 3.23 shows another environment built with six monitors assembled in a different structure from the one previously mentioned. This example has a three level height. The examples of Figure 3.22 and Figure 3.23 are multi-screen, non immersive, and both are assembled on structures.

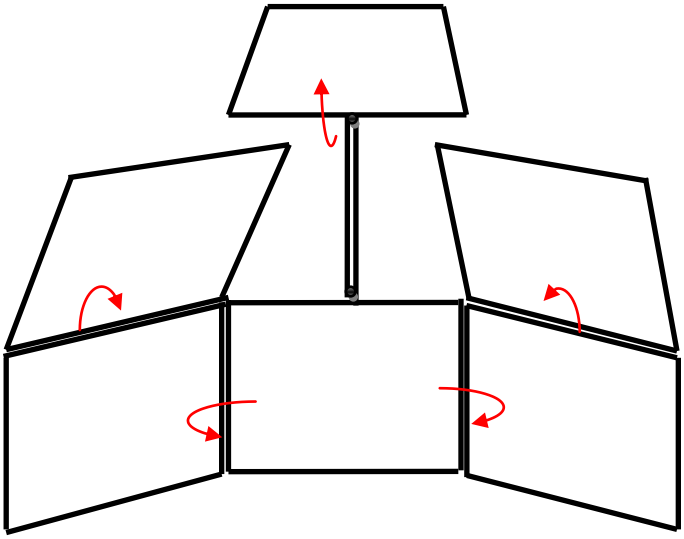


Figure 3.23 An environment built on a fixed structure in three levels

3.3 Computing the Frustum of every Screen

Section 3.1 describes that a frustum volume determines the objects that are displayed on the screen. In this section we will describe how to compute the parameters that OpenGL requires when an asymmetric frustum is used.

Function glFrustum requires the parameters: left, right, top, and bottom. The corners of the screen have been named: *UL, UR, LL, and LR.* These corners are used for defining a new coordinate system by means of vectors basis (Anton, 1992).

It is assumed that the Corner *LL* is the new origin. The vectors basis, which define the new coordinate system are calculated as:

Eq. 3.13 $$X_s = \frac{LR - LL}{\| LR - LL \|},$$

Eq. 3.14 $$Y_s = \frac{UR - LL}{\| UR - LL \|}$$ **and**

Eq. 3.15 $$Z_s = X_s \times Y_s$$

Figure 3.24 shows the directions of the vectors basis in red.

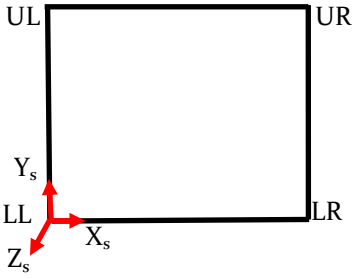


Figure 3.24 Base Vector for the screens

The viewer position regarding the screen (*VPs*) is obtained translating the Viewer Position (*VP*) to the origin of the new screen system:

Eq. 3.16 $$VP_s = VP - LL$$

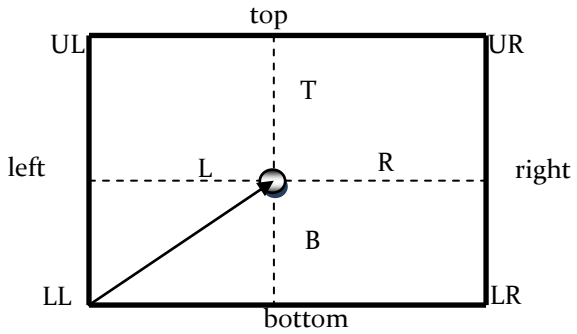


Figure 3.25 Parameters of Frustum

Figure 3.25 shows that the parameter *L* of glFrustum is calculated as the VP_s projection on the axis X_s:

Eq. 3.17 $$L = VP_s \bullet X_s .$$

The parameter *B* is calculated as the projection VP_s on the axis Y_s:

Eq. 3.18 $$B = VP_s \bullet Y_s .$$

With the corners, the height, and the width of the screen are calculated:

Eq. 3.19 $$height = \| UL - LL \|$$

Eq. 3.20 $$width = \| LR - LL \| .$$

Using *L, B, height,* and *width; T*, and *R* are also calculated (Pape, et al., 1978.)

Eq. 3.21 $$T = height - B$$

Eq. 3.22 $$R = width - L$$

The parameters *L, R, B*, and *T* are the distances among the viewer and the four faces of structure of the screen, but OpenGL needs the distance between the viewer and the *near* plane, therefore we have to scale the proportion between both distances.

Eq. 3.23 $$left = -\frac{L\ near}{d}$$

Eq. 3.24 $$right = \frac{R\ near}{d}$$

Eq. 3.25 $$top = \frac{T\ near}{d}$$

Eq. 3.26 $$bottom = -\frac{B\ near}{d}$$

Where $d = VP_s \bullet Z_s$

glFrustum has a default orientation toward –z axis (Neider, et al., 1997), to orient it to its position in the real world it is multiplied by the transpose of the vectors *basis* $[X_s,Y_s,Z_s]^T$ (Stevenson, 2002). M_{view} is obtained translating the viewer position to the origin (*VP*).

Eq. 3.27

$$M = \begin{bmatrix} Xs[0] & Ys[0] & Zs[0] & 0 \\ Xs[1] & Ys[1] & Zs[1] & 0 \\ Xs[2] & Xs[2] & Zs[2] & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Although we have used some formulae of Pape, we did a set of test and found that when we used the transpose of the vectors basis instead of inverse, the result was better.

Eq. 3.28 $M_w \leftarrow_s = (-VP)^T * M$

3.4 Results

This section presents a system, which was programmed using the methodology presented in this chapter. This system considers the screens size, the angle between them, the width of the frames of each screen, and the viewer position.

An important aspect of using monitors is the consideration frames of the screens, because when the frames are not considered the result is an elongated image. Figure 3.26 shows a configuration using two monitors without angle between them. In this configuration the frames of the screens are not considered, producing an incorrect result.

Figure 3.26 Elongated image

The second configuration shows a virtual environment on two coplanar monitors. In this configuration the frames are seen as simple obstructions of the monitors, therefore the virtual world look continuous.

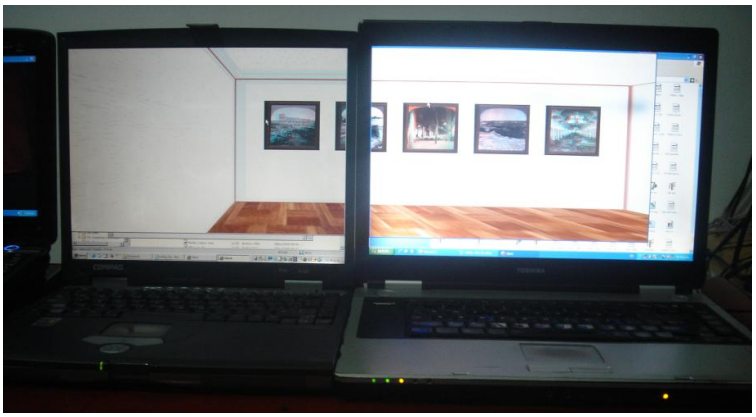


Figure 3.27 Two monitors shows a virtual world

The third configuration is built using two monitors forming an angle of 90 degrees between them, the image displayed is continuous and coherent. Therefore, this photograph has been taken from the correct position, see Figure 3.28

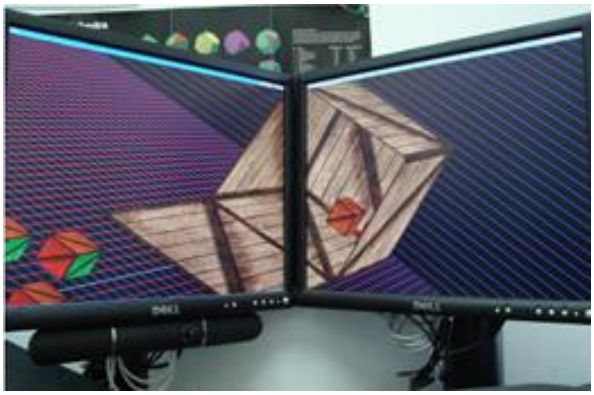


Figure 3.28 Two monitors forming a right angle and displaying a continuous and coherent image.

The fourth configuration is built using three screens, which are aligned. Figure 3.29 shows a virtual world.

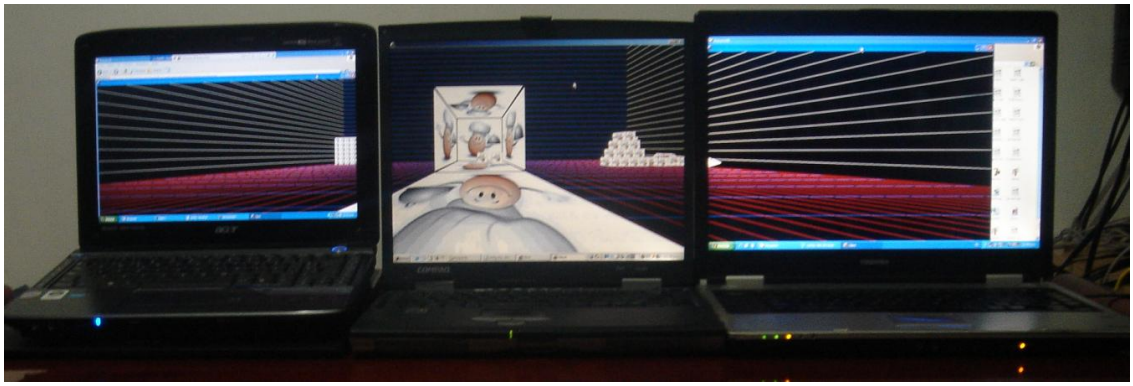


Figure 3.29 Three coplanar monitors displaying a virtual world

Figure 3.30 shows a configuration, which is built with three non-coplanar monitors. Although the frames of the computers are thick and they obstruct somewhat the virtual world, it looks continuous and coherent.

Figure 3.30 Three non-coplanar monitors

3.5 Summary

This chapter presents a study about frustums, which are used to display an image/ virtual environment on multiple screens. Additionally, a set of formulae that allow us to build a physical world with screens side by side was presented. These formulae combined with the formulae to calculate the frustum parameters allow for the building of different virtual environments.

In this chapter the building of dynamic environments with robotic arms has been suggested as future work and a set of formulae were presented to support this proposal. In addition, a system was constructed allowing us to corroborate the methodology presented in this section through different configurations. This system considers the screens size, the angle between them, the width of the frames of each screen, and the viewer position. The configurations presented were built with two or three monitors, which had thick or thin frames. Finally, with the knowledge described in this chapter, a wide variety of virtual environments can be built using different resources and configurations.