Desktop-CAVE for First Person Shooter Games

Fernando Roman, Anderson Maciel and Luciana Nedel
Instituto de Informática - INF
Universidade Federal do Rio Grande do Sul (UFRGS)

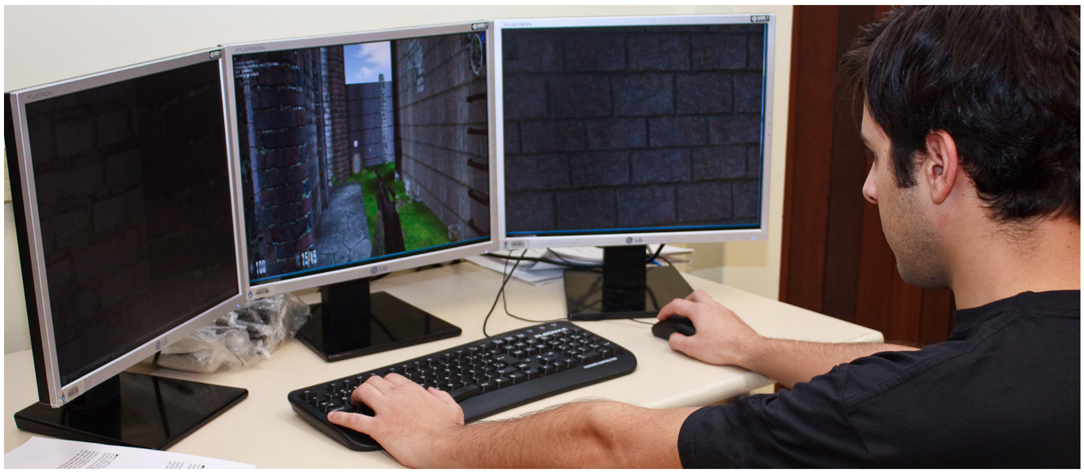


Figure 1: *Desktop-CAVE with three monitors in an open angular configuration*

Abstract

Games are everywhere and, with the sharp improvement of graphics in the later years, a new challenge is to create better interfaces to amplify the sensorial experience of game players. In this context, the present work proposes a desktop based CAVE system using three video monitors with dynamic angles between them. Our hypothesis is that such a system provide an improvement in the players' peripheral vision for 3D first person shooter games. This would even benefit the player performance. In our implementation we used augmented reality and computer vision techniques to calibrate the monitors. Graphics libraries (ARToolkit e OpenGL) have been used to detect and calibrate monitors within the same 3D space and calculate the angles between them. The open source game AssaultCube has been modified to support three monitors and different camera angles. The game have shown to be entertaining and has been used as a use case for user tests. Tests have shown that the desktop CAVE system allows for performance improvement as the players make significatively less look- around movements with the mouse while keeping the average number of kills and deaths favorable in relation to a conventional one monitor setup.

Keywords:: Real-time Strategy, Relief Mapping, Shortest Path Algorithm, First Person Shooter

Author's Contact:

Email: {fjroman, amaciel, nedel}@inf.ufrgs.br

1 Introduction

Video-games and other digital games are becoming more and more present in people's homes and lives. Rich sources of visual content as advertising on TV, YouTube videos and 3D movie theaters have made the games audience more picky, forcing the game industry to offer similar visual quality in games. As a consequence, the entertainment industry is investing huge amounts of money in games which today offer, besides impressive graphics, also very dynamic virtual environments (VE) full of life and adventure. People want to be part of these worlds, and video games are a reachable window to these places. However, for the experience within such worlds to be satisfactory in video-games, the visualization interfaces must be at least as good as the games themselves. Current technology offer TV sets and monitors with very high resolutions. Devices with vertical resolutions of 1080p or Full HD (Full High Definition), which correspond to 1920 by 1080 pixels, are very common today at sizes of over 40 inches.

All these advances contributed to increase the feeling of immersion and presence of a player inside the games' virtual world [Fagerholt and Lorentzon 2009]. However, immersion and presence are concepts borrowed from virtual reality (VR), a domain in which they are the ultimate goal and specific devices are available to provide them. Devices such as head mounter displays (HMD) and cave automatic virtual environments (CAVE) could also be used in games to improve the feeling of immersion and presence to players. Both of them are complex systems which require expensive equipment and large physical spaces, which is acceptable in industrial applications or exhibitions but is inviable for the average player and even for the extreme player.

In this context, the present work proposes the implementation of a display system for desktop games based on three conventional monitors to build up a desktop CAVE. As a standard CAVE configuration with 90 degrees angles between screens might be too much claustrophobic for a desktop configuration, the system also allow arbitrary screen angles with automatic camera calibration. In this work we also propose an evaluation of our desktop CAVE system in the context of a first person shooter game (FPS). An open source game, AssaultCube, has been chosen and adapted to take full advantage of the desktop-CAVE system. We also aim at evaluating how the proposed system can improve immersion, presence and gameplay in such games as FPS where peripheral vision seem to play an important role. We perform tests in which a number of aspects are observed, as shooting accuracy, number of deaths, number of kills, look-around motion, etc. We hypothesize an improvement in player performance based on the fact that the field of view is increased, which amplifies the the peripheral vision, allowing the player to perceive the presence of enemies more quickly than they do through one single monitor. Moreover, by increasing immersion, such approach should also provide a more pleasant gameplay.

The system is implemented using three conventional LCD monitors to build up a CAVE-like multi-display with one frontal and two lateral screens. Varying angles between monitors define which regions of the VE will be shown. A calibration system based on a webcam and computer vision techniques is used to automatically calculate screen angles. Finally, user experiments compare players' performance using the three monitors CAVE with the performance using a traditional one flat screen setup.

In section 2 we present the main VR concepts used in the work as well as related works in games. Section 3 presents the system design, introducing the qualitative parameters which guided the project of the desktop CAVE in contrast with other visualization systems. Section 4 presents the system implementation, including necessary hardware, software libraries and engines used and also how virtual cameras are calibrated from monitor angles. In section 5 we describe and discuss user tests planning and execution as well as an analysis of the results. Conclusions and future works are commented in section 6.

2 Related Work

The development of computer graphics started a few decades ago and has evolved ever since. In the games world, such evolution brought realism to the virtual environments, making the virtual world more and more realistic to users and players. This level of realism is only possible due to more faithful graphics and advanced physics based simulation. However, besides graphics and physics, new interfaces for visualization and interaction have increased the the feeling of immersion and presence in games and other 3D virtual environments today [Kirner and Kirner 2007] [Prabhat et al. 2008].

A brief review of the evolution of games and interfaces is presented in this section to contextualize the reader before we describe our desktop-CAVE interface in sections 3 and 4.

2.1 Games

The games market moves billions of dollars today and has already overcome the movie industry. Both PC games and console-based video games attract more and more people and companies interested on a share of this profitable market.

The first video game ever was created less than 50 years ago by a group of students in the MIT. The Spacewar! ran on a DEC PDP-1 in 1961. In the game, two human players fight against each other controlling a spaceship which could fire missiles. The game has later been distributed with DEC computers and was the first to ever be played by people not directly involved in the project.

In the 70's, the golden age of the arcade games, Nolan Bushnell and Ted Dabney created the Computer Space. 1,500 arcade machines were sold that year. In 1972 the game Pong became the first popular video game. It was released by Atari which sold 19 thousand machines. The same year, the Magnavox Odyssey, represented the first generation of game consoles. After the Space Invaders, in 1978 by Taito, the Atari Asteroids in 1979 and Pac-Man in 1980, the second generation consoles already used ROM-based cartridges.

The 1980's were the time for the games to diversify. Many new genres appeared as adventure, fight, labyrinth, platform, race, RPG and so on. The machines also evolved. In that decade computers as the Commodore Vic20, Commodore 64 and the Apple II took the place of the second generation consoles. Then came the third generation consoles with the 8 bits Nintendo NES. It was an immediate success due to the game Super Mario Bros. Others, like Sega's Master System with the game Sonic appeared at that time. It was the beginning of famous series as The Legend of Zelda and Final Fantasy. In the 1990's the fourth generation 16 bits consoles determined the victory of the consoles over the arcade.

It was also in the 1990's that the introduction of the 3D graphics into video games caused an important revolution. The First Person Shooter (FPS) games and the Real-Time Strategy (RTS) games become very much appreciated. Many games as SimCity, Ultima Online, Mortal Kombat and the precursors of the FPS: Wolfstein 3D (1992) by idSoftware, which used techniques as texture mapping; and Doom (1993) also by idSoftware. Then it was the time for the fifth generation consoles, e.g., Sony Playstation and Nintendo 64, which revolutionized the graphics again. With their huge computing power they offered much better graphics and innovations for FPS. Mission based games with rich stories and small in-game movies appeared.

In the end of the 1990's the sixth generation consoles included support for online gaming, as the Dreamcast. In 2000, Sony's Playstation 2 become the best seller of all times. The Microsoft's XBox and the Nintendo's GameCube also appeared which increased the number of available games to hundreds. With the widespread of the Internet, online games gave a new impulse to many games which allow playing with friends and players from around the world.

The latest game consoles, XBox 360, Playstation 3 and Nintendo Wii, with very high general processing power and also very high graphics processing power due to dedicated chips, allowed game designers to create games with incredibly realistic and dynamic graphics, and with involving richer stories in which every little action of the player influences the game sequence.

Initiatives to move the player out of the desktop to a more natural contact with the game environment are becoming more and more common. Some are directly associated with body motion [Silva and Bowman 2009]. Tangible and physical interaction [Zhou et al. 2007] have also been introduced with the Nintendo Wiimote. Experiences are also being made with mobile devices to adapt well known game styles as FPS to smartphones and similar devices [Wei et al. 2008].

2.2 The CAVE Automatic Virtual Environment

Sutherland created the Head-Mounted Display (HMD) in 1966. It is a device to be wore as a helmet, with a small screen adapted to each of the eyes (Figure 2). With this device he created the idea of a window to the virtual world, increasing the immersion of the user in the virtual environment [Cakmakci and Roland 2006]. However, the use of an HMD involves complex equipment for rendering, tracking, and interaction. It is also cumbersome, with wires connecting the HMD, gloves and other peripherals.

Figure 2: *Binocular Head-Mounted Display (HMD)*

Motivated by the limitations of the HMD, the first CAVE (Cave Automatic Virtual Display) was presented in the SIGGRAPH 1992, by the Electronic Visualization Lab of the University of Illinois at Chicago [Cruz-Neira et al. 1992]. A basic CAVE is a rectangular room where three of the walls and the floor are screens upon which high definition images are displayed (Figure 3). A user inside the room is surrounded and can interact with the virtual environment.

A CAVE is actually a virtual reality interface which promotes immersion. This provides a feeling of presence for a user as they are surrounded by the virtual world. The CAVE design eliminates many problems of other environments, as the interaction limited to one user and low resolution images. Another interesting aspect is that the objects can be seen from both sides and different perspectives depending on the user posture and position.

Simplified CAVE systems have been proposed. The HybridDesk (Figure 4) aimed at creating a workspace aggregating elements of traditional WIMP interfaces (Windows, Icons, Menus and Pointer) with other resources (hardware and software) to allow interactive 3D tasks [Carvalho et al. 2010].

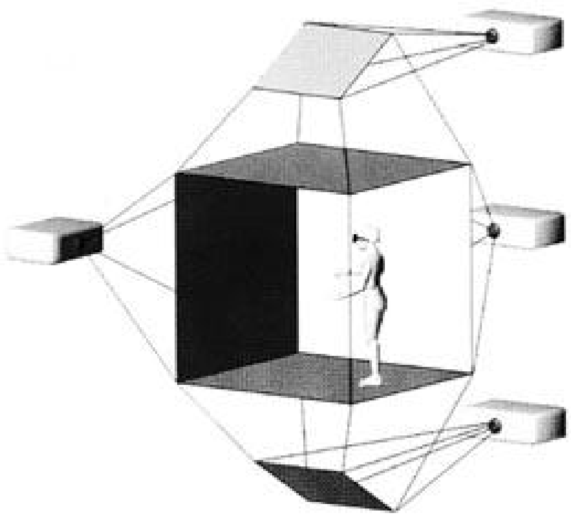


Figure 3: *A typical CAVE system [Cruz-Neira et al. 1992]*

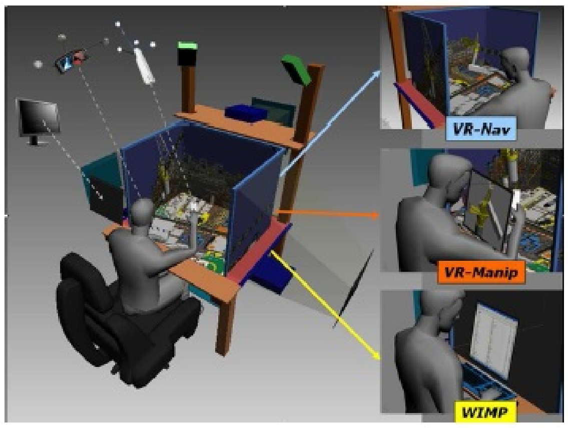


Figure 4: *A schematic of the HybridDesk [Carvalho et al. 2010]*

2.2.1 Characteristics of a CAVE

Each wall is a rear-projection screen and the user can wear polarized or shutter-glasses with stereoscopic projected images. For stereoscopy it is required that two images are rendered for each frame, one for the left and one for the right eye. Shutter-glasses, for example, are synchronized with the projection frequency to send the images for the respective eye, blocking the other one. This produces the illusion of a three-dimensional image, i.e., the objects seem to be floating in the air and can be seen from different points of view.

Some conceptual ideas are linked to the CAVE environment. The willing suspension of disbelief is one of them [Cruz-Neira et al. 1992]. The term has long been used in literature and cinema to define the will of the audience to believe in the images as if they were real, replacing the reality surrounding. The audience enters in a state that they agree to replace the judgement of the truth by the entertainment.

Another concept is the perspective centralized on the user. This idea is based on the camera position along an axis extended orthogonally from the center of the screen. The centralized perspective simulates the view from the viewer position. To keep it, a sensor must constantly send affine tracking information from the simulation.

CAVEs are used today in several areas and are present in many universities. Engineers use CAVEs to improve product development [Buxton et al. 2000]. For example, to design a car part, a CAVE is used to visualize the part in 3D and provide a more accurate mental image of the final part before it is actually produced, improving the quality and reducing the cost of the design. Or to design a new car model and see it from the driver's or the passenger's perspective. Everything before manufacturing any physical element.

3 Desktop-CAVE Design

This section describes how we designed the desktop-CAVE interface, a simplified CAVE system. We first overview the system conception and basic mechanisms. Then we present a qualitative comparison of the desktop-CAVE concept with alternative visualization systems to guide the implementation of the system which is presented in section 4.

3.1 Overview

The system is built upon three identical LCD monitors (Figure 1) and a desktop PC with two graphics cards. The two cards are necessary because each of the cards used allows connecting only two monitors at a time. After the monitors are connected, the operating system is configured to extend the workspace over all monitors creating a single wide desktop area. Then, the game is configured to a higher screen resolution so that the game display spread through the three monitors.

A last and most important task has also been made, which is the monitor calibration. Calibration is accomplished using a webcam. The webcam is placed in front of the monitors in a way that it can *see* them all. Meanwhile, each monitor displays a different fiducial marker. Computer vision techniques are used to calculate the angle between the monitors by comparing how the markers appear in the video captured by the webcam [Claus and Fitzgibbon 2005]. Such angles are then saved and used in the game to define the position and orientation of three virtual cameras, one for each monitor, which are finally used to render the game view to a player (details in Section 4.4.

3.2 Display quality parameters

3.2.1 Immersion parameters

These parameters measure the level of visual simulation provided by a virtual reality interface, also known as suspension of disbelief [Cruz-Neira et al. 1992]. Some parameters involved in creating a willing suspension of disbelief are described below.

Field of view. The field of view (FoV) is the maximum angle a viewer can see without moving their head. Considering a screen with W inches wide and being D the distance from the viewer to the screen, the FoV can be obtained from the expression:

$$FoV = 2 \cdot tan^{-1} \frac{W}{2D} \tag{1}$$

Comparing some of the main visualization interfaces: LCD monitors; head-mounted display (HMD); CAVE; three monitors interconnected – we see distinct scenarios. One LCD monitor provides a variable but limited field of view depending on its size and the viewer distance. For instance, a 19 inches monitor seen at a distance of 18 inches provides a FoV of near 45. With HMDs, as the screens are mounted at a constant distance from each of the eyes, the FoV is constant and angles between 100 and 140 are common. The FoV in a CAVE depends on viewer position, but as the screens surround the viewer, in the ideal case the interface offers a FoV of 360. However, if shutter glasses are used to provide stereo vision, the glasses frame will eventually limit the FoV perceived by the user.

A system with three monitors as we propose, in comparison with a single monitor, provides an increment of the field of view which depends on the monitor angles. In comparison with a traditional CAVE, the three monitors setup present a narrower FoV. However, if a greater number of monitors is used, the FoV could be similar to a CAVE and with a potentially higher resolution.

Panorama. Immersion is also related to the capability of an interface to allow the virtual environment to surround the user. This idea is called panorama and differs from the FoV as the user's head motion is taken into account.

With any monitor-like interface, the idea of panorama is not directly applicable because a monitor is rather a fixed window to the virtual world and does not surround the user whatsoever. With an HMD the panorama is a strong feature as everything the user sees is the simulated world. Whatever side they turn the head or move the eye, they will see some portion of the virtual environment. This also happens within a CAVE, as the users are surrounded by the projections. In CAVEs which are not completely closed by four walls – notice that the typical setup uses three walls, the panorama is interrupted when the user turns towards the empty side. Nevertheless, one advantage to the CAVE in relation to the HMD is that no user tracking is needed to ensure a good level of panorama.

With the three monitors setup, panorama will be higher than one monitor as a user moving their head to the sides will actually see different parts of the VE which are shown on the side monitors. However, due to the size and because the monitors are fixed, HMDs and CAVEs still provide a higher level of panorama.

Intrusion. Intrusion is related to how much the user senses are restricted while using a given interface. The HMD is the most intrusive because it isolates the user from viewing and often listening the real world. A user stepping forward will not know on what they are going to step. Special HMDs for augmented reality can render the real environment mixed with virtual elements, which reduces the intrusion of this interface. Even so, the intrusion persists as the HMD reduces the FoV, causing the user to see a narrow area of the real world. Monitors and CAVEs are less intrusive as they allow the user to move freely, always maintaining knowledge of what are the real and what are the virtual elements of the environment.

The desktop-CAVE with three monitors is not more intrusive than a single monitor. The fact that the user knows where the monitors are placed allows them to clearly separate what is virtual and what is real, interacting with both worlds with a very low probability of confusion.

3.2.2 Visualization parameters

These parameters measure how effective a visualization interface is. Such measure is important because an interface must offer a satisfying visual experience. Some of these parameters are described below.

Visual acuity. The quality of a display is often measured by its resolution, i.e., the number of pixels or points composing the image. The quality of a virtual reality interface is better measured using a combination of resolution and field of view. This measure is called visual acuity of a display.

A typical method to calculate visual acuity uses the resolution and the distance from the viewer to the center of the screen. A screen with H pixels of horizontal resolution and W inches wide presents a pixel pitch of $P = W/H$ inches per pixel. Considering a distance D, the angle represented by one pixel on the retina is $tan^{-1}P/D$ and is measured in minutes. Thus, in visual acuity, the portion of a pixel occupying one minute on the retina is given by the inverse of this angle [Cruz-Neira et al. 1992]:

$$\frac{1}{tan^{-1}\frac{P}{D}} \tag{2}$$

Another metric used to measure vision is the fraction of Snellen, $20/X$. Such fraction shows that an observer situated at 20 feet can see the same as an observer with normal vision can see at X feet with the naked eye. For example, a person with vision $20/40$ placed at 20 feet from a scene can see the same details a normal person could see at 40 feet from the same scene. Then, this person has a vision considerably poorer than most of the people.

If we consider, for example, a 19 inches monitor with $1280x1024$ pixels of resolution and a distance of around 18 inches from the screen we obtain a visual acuity of $20/45$. Comparing with the international standards for issuing driver licenses, this is an unsatisfactory visual acuity to drive light vehicles, category B. The minimum recommended acuity is $20/30$. With a CAVE, considering that each projector has a horizontal resolution of 1280 pixels for a 7 feet screen, the visual acuity is around $20/110$. If it is possible to increase the resolution of the projectors, the visual acuity will increase proportionally. Current HMDs, in turn, often worsen drastically the visual acuity to near $20/425$. This is due to the screen size and the very low resolution. In terms of acuity, a CAVE is not so bad as an HMD, but is also not as good as monitor.

With three similar 19 inches monitors the visual acuity is equal to the one of a single monitor, as the size of the screen increases together with the total number of pixels.

Look-around. This parameter represents the possibility for a viewer to move around an object and see it from different perspectives [Cruz-Neira et al. 1992]. Visualizations with this property can be used in many applications. It makes possible to model a new product in 3D and inspect it from different angles much before the physical product is manufactured or built.

The look-around property does not work with common LCD monitors. When one moves to the side they begin to see a smaller area of the screen. Moving even further, the screen can no longer be seen. With an HMD it does not happen. Anytime the user moves to the side, or look around, the system recalculates their position in relation to the virtual world and displays the right portion of the VE to the user. Look-around is also plenty valid within a CAVE. As they move, users displace their field of view to different areas of the projections causing them to see elements of the VE from a perspective spatially associated with their real position. Wearing shutter-glasses for stereoscopic vision the feeling is even more intense. Closer objects will be seen as if they were floating in the air in front of the user who is able to move and see them from any of the sides.

When using three monitors configured as a desktop-CAVE, it is possible for a user to look-around and see different parts of the VE. However, due to the size of the monitors, the freedom to move is restricted, and looking-around is only possible at a limited range.

Collaboration. Visualization interfaces can also be classified according to their potential to allow collaboration. It means that they are more collaborative if they allow more than one user to see and/or interact with a VE at the same time.

This is possible with a monitor-like interface, but the perspective will be that of only one of the users. The same happens with a CAVE and a desktop-CAVE, but in these cases the surrounding screens and the use of shutter-glasses allow individual user perspectives to be perceived. With HMDs it is only possible to have collaboration if every user wears an individual HMD and the system is able to render every individual perspective of the same VE in time.

4 System Implementation

This section describes the implementation of the desktop-CAVE interface.

4.1 Third party software

We begin by presenting the existing software frameworks, toolkits and libraries chosen to integrate our implementation.

4.1.1 ARToolkit

ARToolkit is a very popular toolkit for fast development of augmented reality (AR) applications. It is widely used in part because it is open source, inviting the users to run, study and modify the available examples at will.

The toolkit is implemented in C e C++, offers support to the development of AR applications with low computational cost. It implements computer vision algorithms which are essentially used for optical tracking. It is able to estimate in real time the position and

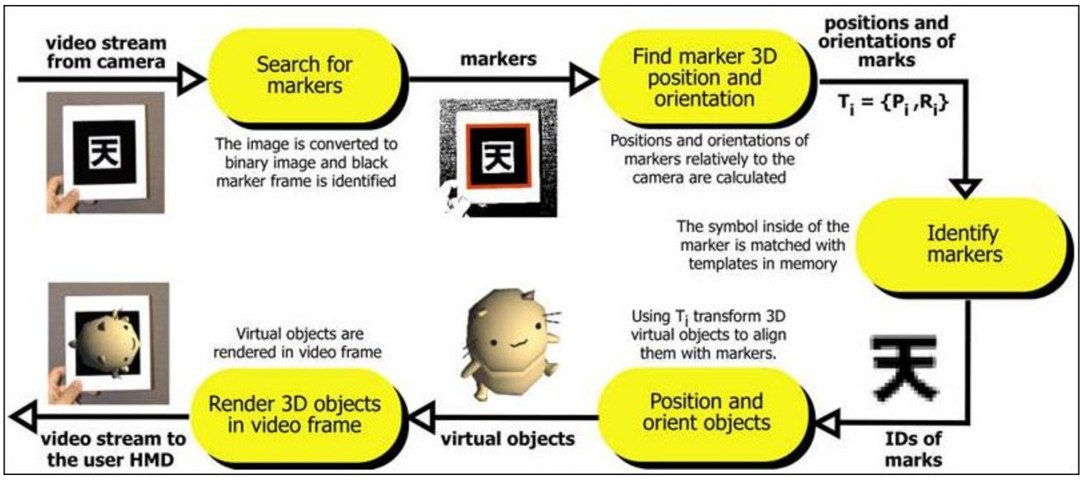

video stream from camera
Search for markers
markers
Find marker 3D position and orientation
positions and orientations of marks
$T_i = \{P_i, R_i\}$
The image is converted to binary image and black marker frame is identified
Positions and orientations of markers relatively to the camera are calculated
The symbol inside of the marker is matched with templates in memory
Identify markers
Virtual objects are rendered in video frame
Using T_i transform 3D virtual objects to align them with markers.
Render 3D objects in video frame
Position and orient objects
video stream to the user HMD
virtual objects
IDs of marks

Figure 5: *ARToolkit pipeline [Kato]*

orientation of markers in relation to the video capture device, usually a webcam. In the AR applications this is used to place and orient virtual elements inside a real scene captured by the camera, building a mixed reality scenario.

For AR applications, the use of the toolkit can be summarized by the following steps:

1. set video parameters; initialize camera; read marker files;
2. capture a video frame;
3. detect and identify markers on the video frame;
4. calculate the relative transformation between marker and camera reference frames;
5. render virtual object on the marker reference frame.

These steps are illustrated in Figure 5, extracted from [Kato].

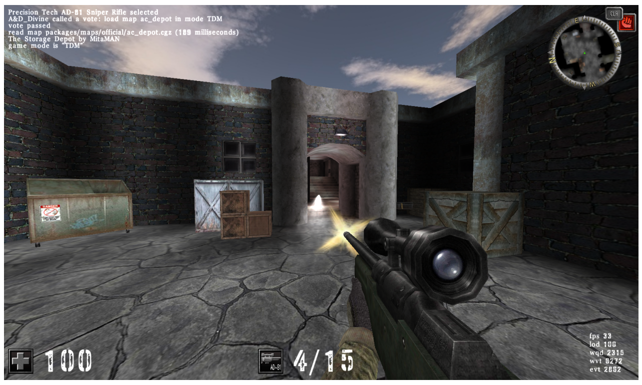
One limitation of the ARtoolkit is the range of distance between camera and marker. If they are too far, the marker can be too small to be identified; if they are too close, parts of the marker may stay out of the video frame, being impossible to identify it. Marker size can be adjusted to minimize this problem.

In the present work we use the tracking capabilities of ARToolkit to calibrate the relative angles between monitors in the desktop-CAVE configuration. See details in section 4.4.

4.1.2 AssaultCube

AssaultCube (Figure 6) is an open source first person shooter game based on the Cube engine and game. It was started in 2004 with the name *ActionCube*, by member of the Cube community. Official release date was in November 2006, and in May 2007 the name was changed for AssaultCube to avoid ambiguity with the name of another game, *Action Quake*. The game is a more realistic version of the original Cube game which keeps the simplicity and velocity of the original game.

In the game, the characters, including players, are divided in two factions: the Cubers Liberations Army (CLA) and the Rabid Viper Special Forces (RVSF). A player chooses which of the factions they want to join before starting the game. The game can be played online with and against human players or with virtual enemies (bots). There are twelve different game modes: Capture the Flag, Keep the Flag, Team Keep the Flag, Deathmatch, Team Deathmatch, One Shot One Kill, Team One Shot One Kill, Last Swiss Standing, Survivor, Team Survivor, Pistol Frenzy and Hunt the Flag. Among them, only the modes Deathmatch, Team Deathmatch and One Shot One Kill are available to play against bots.



```
Precision Tech AD-81 Sniper Rifle selected
A&D_Divine called a vote: load map ac_depot in mode TDM
vote passed
read map packages/maps/official/ac_depot.cgz (189 milliseconds)
The Storage Depot by MitaMAN
game mode is "TDM"
```

```
fps 33
lod 100
wqd 2315
wvt 8272
evt 2862
```

Figure 6: *Scene of the game AssaultCube*

4.2 Hardware

Three LCD monitors have been used in this work. All having the same size, 19 inches, and the same resolution, $1280x1024$ pixels.

A desktop PC with two graphics cards (GPUs) has also been used. The GPUs are both from Nvidia. One is a GeForce 9600 and the other is a GeForce 8600. Two identical GPUs are recommended for safe compatibility, but this is not a hard requirement.

For monitor angles calibration, an ordinary 1.3 megapixels webcam has been used. This resolution is recommended as lower resolution cameras may have difficulties to detect the markers appropriately.

4.3 What changes in the game code

The original game source code has been modified to support the rendering of three independent viewports (one for each monitor) and to allow angle calibration. This is done by first replicating the virtual camera twice to render one frame for each of the viewports. This causes the three monitors to render the same view. Then two of the cameras are rotated laterally to visualize the sides of the scene in such a way that a continuous horizontal field of view is provided sewing the three viewports at the vertical edges of neighboring monitors (see figure 7).

Minimaps, life gauges and ammunition information are displayed only on the central/frontal screen, which has the focus.

Source-code is presented in Algorithm 1. It shows how each of the viewports are defined, how the cameras are rotated and that a frame is rendered by calling the function `gldrawframe`.

As the angles between monitors can vary, one last step is necessary to obtain a consistent visualization for every angles. If the angle between them is equal to the field of view, we have the ideal case in

Algorithm 1 Definition of the three viewports

```
//FIRST VIEWPORT
glViewport(0, 0, screen->w/3, screen->h);
viewportNum = 1;
player1->yaw = player1->yaw-dynfov();
computeraytable(camera1->o.x, camera1->o.y, dynfov());
gl_drawframe(screen->w, screen->h,
             fps<lowfps ? fps/lowfps : (fps>highfps ? fps/highfps : 1.0f), fps, viewportNum);
if(frames>4) SDL_GL_SwapBuffers();
player1->yaw = player1->yaw+dynfov();

//THIRD VIEWPORT
glViewport(2*screen->w/3, 0, screen->w/3, screen->h);
viewportNum = 3;
player1->yaw = player1->yaw+dynfov();
computeraytable(camera1->o.x, camera1->o.y, dynfov());
gl_drawframe(screen->w, screen->h,
             fps<lowfps ? fps/lowfps : (fps>highfps ? fps/highfps : 1.0f), fps, viewportNum);
if(frames>4) SDL_GL_SwapBuffers();
player1->yaw = player1->yaw-dynfov();

//SECOND VIEWPORT
glViewport(screen->w/3, 0, screen->w/3, screen->h);
viewportNum = 2;
computeraytable(camera1->o.x, camera1->o.y, dynfov());
gl_drawframe(screen->w, screen->h,
             fps<lowfps ? fps/lowfps : (fps>highfps ? fps/highfps : 1.0f), fps, viewportNum);
if(frames>4) SDL_GL_SwapBuffers();
```

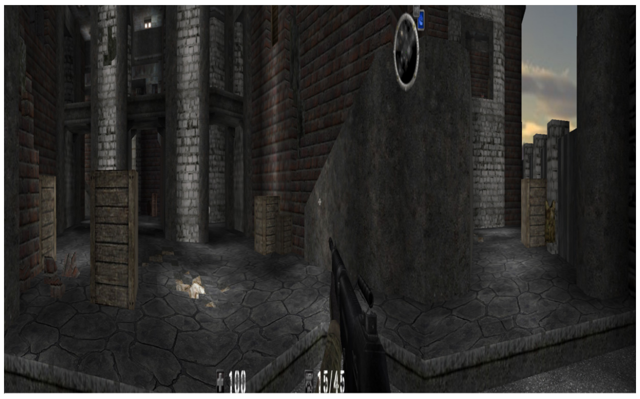


Figure 7: *Three viewports with a continuous image*

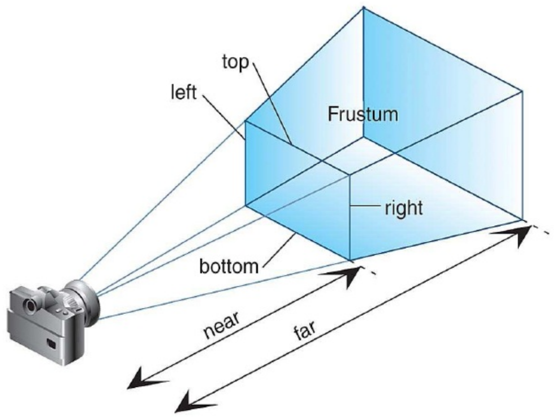


Figure 8: *Frustum definition*

which the default camera projection parameters are correct. However, as the angles may be changed, we also had to proceed with modifications in the function `setperspective` of the Cube engine. Such changes aim at configuring the view frustum according to the angles calibration (section 4.4). As shown in figure 8 the frustum defines the visualization volume within which are placed the objects that will be rendered.

With our method we use three cameras, and each of them has its own frustum and field of view. Thus, to obtain the widest possible visualization, the angles between monitors should be equal to the field of view of each camera, see figure 9. When the angles between monitors are different from the FoV, the viewer sees only part of the original total FoV, see figure 10.

The precalculated monitor angles (see section 4.4) are read from a text file and stored as global variables in the game code. In the function `setperspective` it is necessary to check if the angles between monitors are the same as the FoV angles. If they are, the original frustum parameters are used with the function `glFrustum`. Otherwise, a correction must be made to obtain a coherent visualization.

Such correction is made by increasing the frustum parameter `left` of the leftmost viewport and decreasing the frustum parameter `right` of the rightmost viewport. To calculate the amount of increase and decrease we first have to compute how much a degree represents in relation to the size of the viewport. For example, if the angles between monitors are $30°$, the frustum `left` is equal to -50, the frustum `right` is equal to 50 e the FoV angle is $60°$, each degree corresponds to $100/60 = 1.667$. This value can be used to calculate the frustum size with a different angle. We multiply, then, the angle between monitors ($30°$) by the value corresponding to $30°$ (1.667). We obtain 50 which is the new size of the visible frustum for this monitor. Finally, to obtain the frustum `left` for the left viewport, we subtract the new size from the frustum `right`. In the example the new `left` will be 0.

If such example represented the right viewport, we would have to calculate the frustum `right`, which would be the `left` plus the size, which is equal to 0. As the whole screen is occupied by this narrower visualization area, the final image being displayed looks stretched. Some objects will appear horizontally larger on these viewports, even if the user placed in a central position will not notice that.

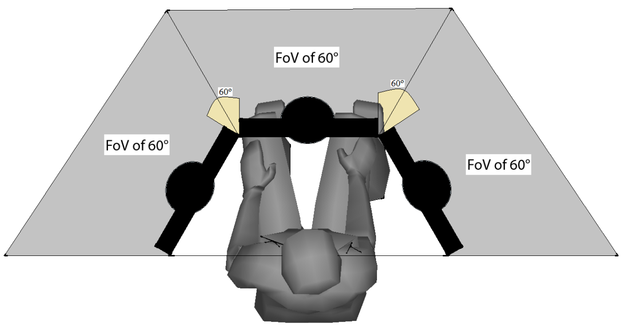


Figure 9: *Angle between monitors equal to the FoV angle (60°)*

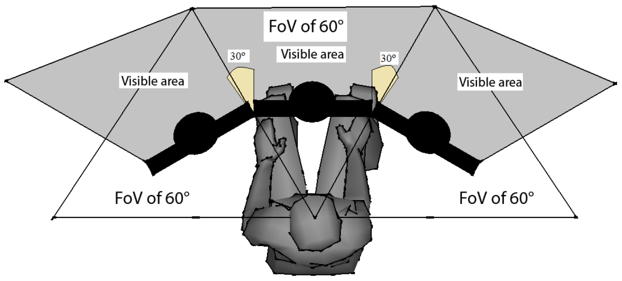


Figure 10: *Angle between monitors (30°) different form the FoV angle (60°)*

4.4 CAVE angles calibration

To calibrate the angles between monitors we created an application based on the ARToolkit. With the application, a webcam is used to film the the monitors while these latter present fiducial markers on the screen (Figure 11). The application, using computer vision algorithms from ARToolkit, identifies the markers and calculates the angles between the webcam and each marker. As a result, the relative angles between the markers and consequently between the monitors can be obtained.

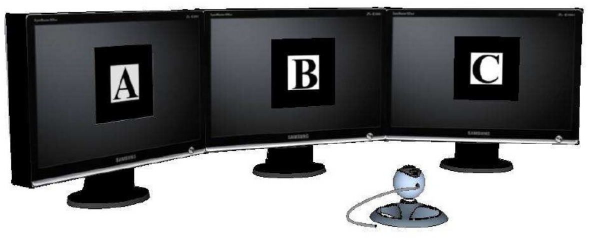


Figure 11: *Schematic of the monitors being tracked by the webcam*

4.4.1 Tracking and identification of fiducial markers

Fiducial markers are images containing visual features that are easy to be extracted. Often, they are black and white square figures which contain identifiable symbols (Figure 12). The full marker must be visible for the marker to be successfully tracked by a camera and identified by vision algorithms.

The tracking procedure processes the image, extracts image information like vertices for detection or identification features, and estimates marker position and orientation. Pattern recognition is performed by identifying the four vertices of square regions contained on a video image, which is then converted to a binary image (black and white). The symbol inside the vertices is compared to templates input by the user or developer [Claus and Fitzgibbon 2005]. Whenever the information contained in the extracted square is similar to any of the registered markers, the system identifies the marker and determines its relative pose to the camera.

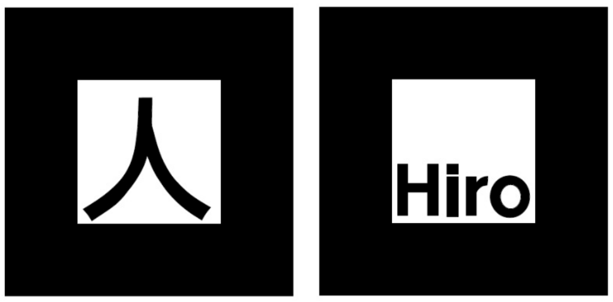


Figure 12: *Examples of ARToolkit markers*

Marker pose (position and orientation) is determined by relating marker and camera coordinates as in Figure 13.

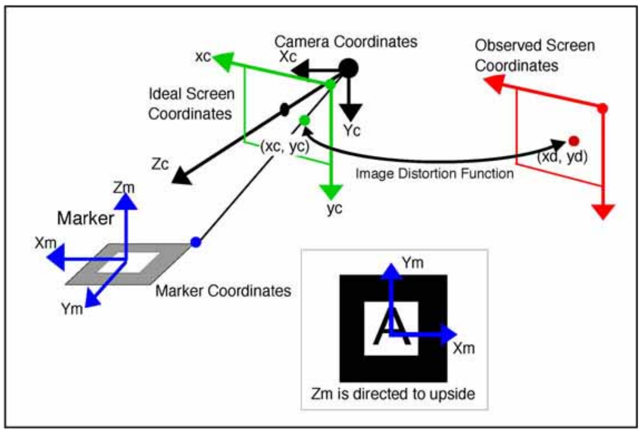


Figure 13: *Relation between marker and camera coordinates [Kato]*

A $4x4$ homogeneous transformation matrix T containing the relation between marker and camera coordinates is obtained from ARToolkit for every video frame (Figure 14). The multiplication of T by a 3D point in the marker (Xm, Ym, Zm) allows obtaining the corresponding point in the camera coordinates system (Xc, Yc, Zc).

$$\begin{bmatrix} Xc \\ Yc \\ Zc \\ 1 \end{bmatrix} = \begin{bmatrix} R_{11} & R_{12} & R_{13} & T_1 \\ R_{21} & R_{22} & R_{23} & T_2 \\ R_{31} & R_{32} & R_{33} & T_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xm \\ Ym \\ Zm \\ 1 \end{bmatrix} = \mathbf{T} \begin{bmatrix} Xm \\ Ym \\ Zm \\ 1 \end{bmatrix}$$

Figure 14: *Transformation matrix*

This matrix allows extracting the angle between a marker and the camera. As a marker is shown for each monitor, after storing the camera-marker angles for each marker/monitor, to calculate the angle between the three monitors is straightforward.

5 User experiments

5.1 Goals and hypotheses

The proposed system has been assessed through user experiments. With the tests we aimed at responding the question: does a desktop-CAVE using three monitors help in improving gameplay and immersion for players of first person shooter games? The approach was to log parameters like shooting precision, number of deaths, number of kills, number of injuries and horizontal movement, and propose a questionnaire to the players after the test for subjective evaluation. We assume that the increase of peripheral vision obtained with the desktop-CAVE should improve performance.

5.2 Tests planning and execution

Most of the tests have been performed during an Open Day event at our university. Random players from the academic community and outside have been invited to participate.

A short text explaining the test was presented to all volunteers before the test. The text introduces the game and present instructions to play. A questionnaire was responded by the volunteers after the test. The questionnaire should answer if the players subjectively felt more immerse in the VE of the game.

The test itself consists of playing the first person shooter game AssaultCube. In the test, the player has one minute to get used with the controls and the game velocity. Just after, two tests of 2 minutes are performed, one with one monitor and the other with three monitors. The order of the tests (one or three monitors) is randomized to avoid the learning curve to interfere in the results. During these 2 minutes, performance parameters are logged. Two maps of the game have been chosen: the map `acgothic` and the map `accomplex`, and they were also randomly selected.

5.3 Results and discussion

25 people have been submitted to the test. With the questionnaire we noticed that most of the people has some experience with digital games, and that more than half of them has at least 8 years of experience (figure 15). According to their own judgement, they have a varied skill level for FPS games: 32% is weak; 24% average skills; 24% consider they are fairly good; 20% affirm they are very good. None believe they are among the best players around (Figure 16).

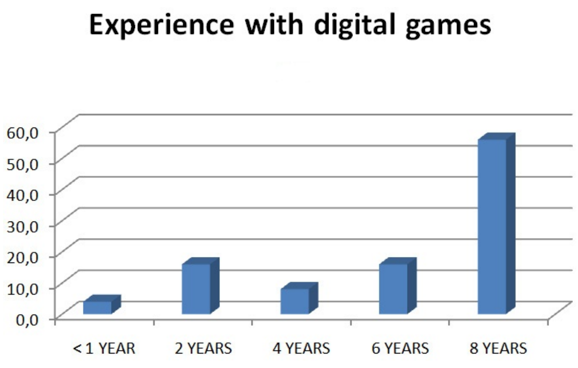


Figure 15: *Subjective level of experience in games of the volunteer users*

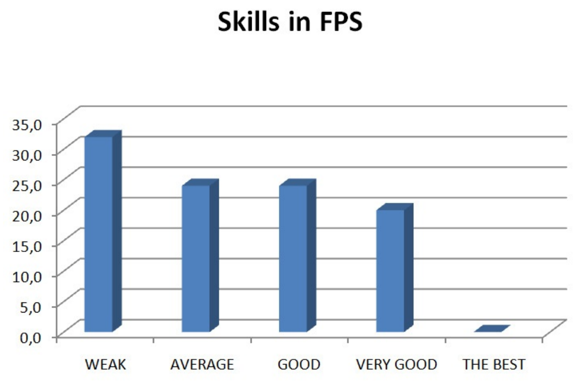


Figure 16: *Subjective skill level in FPS games of the volunteer users*

Even if most people have experience with digital games, very few have experience with other types of 3D applications (Figure 17). Examples of 3D applications are CAD tools and authoring tools like Blender and 3DStudioMax.

To evaluate the influence of the desktop-CAVE in the performance 5 variables have been analyzed: precision, number of deaths, number of kills, number of injuries and horizontal look-around motion.

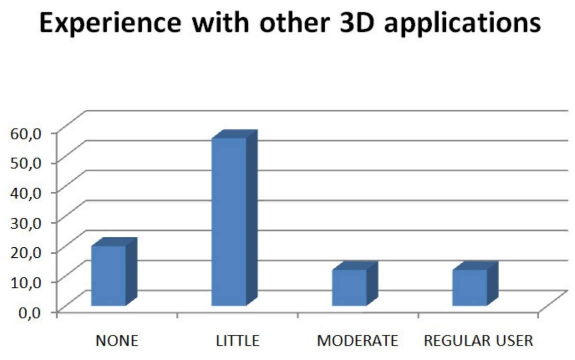


Figure 17: *Subjective level of experience in other 3D applications than games*

All of them gave better averages when the player was using the desktop-CAVE with three monitors than the conventional one monitor display. However, better averages not always are enough to prove the hypotheses as random elements can be more influential then an independent variable, especially with a small number of samples. Then, we performed Student's t-tests to evaluate the statistical significance of the higher averages.

5.3.1 Precision

Precision is the number of hits divided by the total number of shootings. For precision, with one single monitor we computed an average of 21.75% shots that hit an enemy. With three monitors the average was 24.36%. Even with the higher average precision for the desktop-CAVE, the t-test with $\alpha = 5\%$ resulted an F value of 1.04 which reveals a probability of 68.8% that our hypothesis hold. This is not enough to provide statistical significance as at least 95% is usually required to overcome the random effects.

5.3.2 Number of deaths

This is the number of times the player died, shot by enemies, during the 2 minutes test. The average of deaths was 4.6 with one single monitor. With the three monitor CAVE, the average was 3.36, meaning that players died less with the proposed interface. however, the t-teste with $\alpha = 5\%$ resulted in an F of 3.25, which reveals a probability of 92.24% that the hypothesis hold. This is a good probability. Even so, the required threshold of 95% was not reached and we cannot state positively that the desktop-CAVE helps the player to die less than with an ordinary monitor.

5.3.3 Number of kills

This is the number of enemies killed by the player. The average number of kills with one monitor was 3.04 per test, and with three monitors it was 3.76 kills per test. Analogously to the previous parameters, the t-test with $\alpha = 5\%$ resulted in an F value of 1.03 revealing a probability of 68.52% only that these average resulted from the types of visualization interface tested. Thus, despite the better average, we cannot affirm that this difference is caused by the interface.

5.3.4 Number of injuries

This is the number of times the player was hit by an opponent's shot. The average with one monitor was 21.88 injuries per test, and with three monitors it was 17.40 injuries per test. Again, the desktop-CAVE obtained a considerably better average. However, the t-test with $\alpha = 5\%$ resulted in $F = 1.51$ which indicates a probability of around 77.5% that the hypothesis hold. Again, there is not enough statistical significance to prove the hypothesis despite the good results.

5.3.5 Horizontal motion

This parameter represents the total horizontal angular motion executed by a player during the test. This motion is applied moving the mouse to either sides and causes the player vision to turn to the corresponding sides as if they were turning the head. The angles turned at every move-and-stop of the mouse, in degrees, were summed up to compose this parameter. With one monitor, the average angle was 5676° per test, and with three monitors it was 3732° per test. The t-test revealed that this test is statistically significant, with a probability of at least 99.99% that the hypothesis hold. The t-test with $\alpha = 5\%$ resulted in $F = 28.88$. This result supports the hypothesis that the system with three monitors improves the player's peripheral vision, as they see enemies approaching from the sides and only have to turn to shoot them or run away.

In consequence, the players also turn quickly enough to get less injury, die less and have more time to aim, obtaining a better rate of hits. This statement is supported by the good averages in all parameters tested. Concluding, we can say that the desktop-CAVE effectively participates in improving performance in this game.

Moreover, several volunteers stated that the three monitors system caused discomfort or strange feeling in the beginning. But as they play, they felt better. A few minutes are, then, required for adaptation. Another evidence of is that the players commented that they did not look at the lateral monitors in the beginning, but as an enemy appeared there they could notice, and then they started to look for enemies in the laterals too after sometime.

The questionnaire revealed that 92% of the players like better the three monitors setup. The exact same percentage of testers believed that they played better using the desktop-CAVE, even when it was not supported by the captured data.

6 Conclusion

We presented the design and development of a desktop-CAVE interface for visualization in 3D games. The interface is built using three LCD monitors as screens to compose a simplified CAVE. We modified an open source first person shooter game to analyze the influence of such interface in game performance. Different aspects as shooting precision, number of kills, injuries and deaths were studied along an experiment with 25 randomly selected users. We observed that players have got more involved by the game, presenting better performance when using the desktop-CAVE in comparison with a conventional one monitor interface. We also observed that subjectively the volunteers felt more immersed in the game environment, which allowed a more satisfying experience.

The performance improvement has been attested especially with the measurement of the horizontal angular motion. Users actually do not have to look around all the time because the CAVE provides a considerable increase of the peripheral vision as compared with a single monitor. They identify enemies approaching from the sides more quickly and can promptly react. Even if some of the measured parameters do not present a large difference when comparing the averages, as a whole they always corroborate to the validity of the original hypothesis.

Improvements can be made to the system. One is to compute, besides relative angles, the relative monitor positions. This would allow to place the monitors anywhere around the player, not constraining them to have a common edge as in our current implementation. Another improvement would be to calibrate dynamically during the game, but for this it would be necessary to track the monitors and the player face all the time. This could be made using a number of simple strategies as for example fixing a webcam somewhere on the player's head and markers attached to the monitors. The player would be able to move freely and anytime they look at a monitor the webcam would identify the marker and calculate the appropriate virtual camera position to render the VE on that screen.

Regarding the user tests, we believe that using a higher number of volunteer players, or even by grouping them according to similar profiles, the random effects in the tests would vanish and high statistical significance would be obtained for all parameters. This should happen because most of the analyzed parameters are strongly associated with how quickly the players acquire skills for a game they do not know.

Acknowledgements

The authors would like to thank CNPq-Brazil for the support through projects 309092/2008-6, 481762/2008-6, 580156/2008-7, 481268/2008-1 and 302679/2009-0. Thanks are also due to all the volunteers who kindly tested the system and had fun doing so.

References

BUXTON, W., FITZMAURICE, G., BALAKRISHNAN, R., AND KURTENBACH, G. 2000. Large displays in automotive design. *IEEE Computer Graphics and Applications 20*, 4, 68–75.

CAKMAKCI, O., AND ROLAND, J. 2006. Head-worn displays: a review. *Journal of Display Technology 2*, 199–216.

CARVALHO, F., RAPOSO, A., GATTASS, M., AND TREVISAN, D. 2010. Um sistema hbrido semi-imersivo de baixo custo para interaes 2d-3d. *XII Symposium on Virtual and Augmented Reality 1*, 153–162.

CLAUS, D., AND FITZGIBBON, A. W. 2005. Reliable automatic calibration of a marker-based position tracking system. In *Application of Computer Vision*, IEEE Computer Society, Breckenridge, CO, USA, 300–305.

CRUZ-NEIRA, C., SANDIN, D. J., DEFANTI, T. A., KENYON, R. V., AND HART, J. C. 1992. The cave: audio visual experience automatic virtual environment. *Communications of the ACM 35*, 64–72.

FAGERHOLT, E., AND LORENTZON, M. 2009. *Beyond the HUD - User Interfaces for Increased Player Immersion in FPS Games*. Master's thesis, Chalmers University of Technology.

KATO, H. *ARToolKit Documentation, 2010. www.hitl.washington.edu/artoolkit/documentation*.

KIRNER, C., AND KIRNER, T. G. 2007. Virtual reality and augmented reality applied to simulation visualization. *Simulation and Modeling: Current Technologies and Applications 1*, 391–419.

PRABHAT, FORSBERG, A., KATZOURIN, M., WHARTON, K., AND SLATER, M. 2008. A comparative study of desktop, fishtank, and cave systems for the exploration of volume rendered confocal data sets. *IEEE Transactions on Visualization and Computer Graphics 14*, 3, 551–563.

SILVA, M. G., AND BOWMAN, D. A. 2009. Body-based interaction for desktop games. In *CHI '09: Proceedings of the 27th international conference extended abstracts on Human factors in computing systems*, ACM, New York, NY, USA, 4249–4254.

WEI, C., MARSDEN, G., AND GAIN, J. 2008. Novel interface for first person shooting games on pdas. In *OZCHI '08: Proceedings of the 20th Australasian Conference on Computer-Human Interaction*, ACM, New York, NY, USA, 113–121.

ZHOU, Z., TEDJOKUSUMO, J., WINKLER, S., AND NI, B. 2007. User studies of a multiplayer first person shooting game with tangible and physical interaction. In *ICVR'07: Proceedings of the 2nd international conference on Virtual reality*, Springer-Verlag, Berlin, Heidelberg, 738–747.