Universidade de Aveiro
Departamento de Eletrónica, Telecomunicações e Informática
2015

João Henrique Bastos Pinto

Ambiente de Realidade Virtual para visitas imersivas e interação

Virtual Reality Environment for immersive walkthroughs and interaction

Universidade de Aveiro
2015

Departamento de Eletrónica,
Telecomunicações e Informática

João Henrique Bastos Pinto

Ambiente de Realidade Virtual para visitas imersivas e interação

Virtual Reality Environment for immersive walkthroughs and interaction

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Doutor Paulo Dias, Professor auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e da Doutora Beatriz Sousa Santos, Professora associada com agregação do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor Joaquim João Estrela Ribeiro Silvestre Madeira
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Professor Doutor Paulo Jorge Carvalho Menezes
Professor Auxiliar da Universidade de Coimbra

Professor Doutor Paulo Miguel de Jesus Dias
Professor Auxiliar da Universidade de Aveiro

agradecimentos /
acknowledgements

Quero aqui agradecer aos meus pais, ao meu irmão, à minha namorada, aos meus amigos, e a todos os que me apoiaram até ao culminar deste longo percurso académico. Deixo também agradecimentos aos Professores Paulo Dias e Beatriz Sousa Santos pelo acompanhamento e orientação durante a elaboração desta dissertação, e ao Sérgio Eliseu, pela oportunidade de enquadramento artístico que forneceu a este trabalho.

Palavras Chave

Ambiente Virtual, Navegação e Interação, Estudos com utilizadores, Kinect,Realidade Virtual

Resumo

Como solução para visitas virtuais imersivas a museus, propomos uma extensão à plataforma previamente desenvolvida para efectuar a configuração de ambientes virtuais imersivos (pSIVE), mantendo todas as funcionalidades de criação de ambientes virtuais e de associação de conteúdos (PDF, videos, texto), mas também permitindo interações baseadas em gestos e navegação. Para isso, propomos navegação um para um usando rastreamento do esqueleto com uma Kinect que é calibrada no espaço do mundo real em que o utilizador se situa, assim como métodos de interação por gestos. Para validar os métodos propostos de navegação e interação, foi efetuado um estudo comparativo entre a interação e navegação à base de gestos e em botões. Com os resultados desse estudo em mente, desenvolvemos novos métodos de interação com seleção via direção do olhar. A aplicação desenvolvida foi testada num cenário real, como parte de uma instalação artística no museu da cidade de Aveiro, onde os visitantes podiam navegar uma sala virtual do museu e manipular objetos de maneira a criar a sua própria exposição.

Keywords Virtual Environment, Navigation and Interaction, User Study, Kinect, Virtual Reality

Abstract As a solution to immersive virtual museum visits, we propose an extension upon the platform previously developed for Setting-up Interactive Virtual Environments (pSIVE) that maintains all of the Virtual Environment creation features of the platform as well as content association (PDF, Video, Text), but also allows gesture-based interaction and one to one navigational input using skeleton tracking with a Kinect that is calibrated in the real world space where the user is located in. In order to validate our new navigation and interaction methods, a comparative study was performed, comparing our gesture-based interaction and navigation with controller-based navigation and interaction methods. With the results of that study in mind, we developed new interaction methods with gaze-directed selection. The developed application was tested in a real scenario, as part of an art installation at the museum of the city of Aveiro, where visitors could navigate in a virtual room of the museum and manipulate objects in order to create their own exposition.

Contents

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 MOTIVATION

With the rising popularity of virtual reality hardware in the last years, and with the introduction of next-generation head-mounted affordable displays and tracking systems, virtual reality applications are beggining to have wider expression due to cost reduction and software availability. This work comes from a necessity of a flexible and easy to configure platform that supports the latest virtual reality hardware and 3D interaction techniques through gestures or controllers. As a reply to those necessities, this work is a continuation of a previous master thesis, that developed the platform for Setting up Interactive Virtual Environments (pSIVE) (Souza, 2013), which not only allows an easy configuration of a virtual world using a diversity of models, but also interaction with the environment using non-conventional hardware such as trackers and head-mounted displays (HMDs), fitting a lot of our needs.

1.2 OBJECTIVES

This dissertation has the objective of building several new features upon a previously developed framework (Souza, 2013), updating it to support new head-mounted displays such as the Oculus Rift, as well as gesture inputs detected by a Microsoft Kinect sensor, while keeping its flexibility and ease of use, in order to support navigation and

interaction with objects in a configurable immersive virtual reality environment. One of the environments that we aimed to create as a proof of concept of the proposed framework is a virtual museum, in which the goal is the complete immersion of the user in an empty room using several skeleton-tracking kinect sensors and a head mounted display with orientation tracking (Oculus Rift), allowing the user to physically navigate and interact within the virtual world. With those criteria in mind, the developed framework permits the creation and set-up of a custom environment, allowing the users to interact with the virtual objects within the museum, either by browsing content (such as PDF files, images, text information and videos), or inserting and manipulating objects in the museum. Additionally, we propose and evaluate several interaction and navigation methods in relation to conventional controller-based interaction and navigation methods.

1.3 STRUCTURE

This dissertation is divided in six major chapters; The first chapter introduces the motivation driving the work and its objectives. The second chapter presents concepts of Virtual Reality and Interaction, compares some of the frameworks considered for the development of this framework, and some related work. The third chapter describes the developed framework, explaining the architectural decisions and its implementation, as well as the proposed methods of navigation and interaction. The fourth chapter contains the methodologies and results of the usability studies performed, concerning comparisons between input techniques and 3D manipulation methods. The fifth chapter presents the "Imaginary Museum"; and application developed based on the extended platform and showcased as part of an exhibit in a museum. The sixth chapter concludes this dissertation, and contains our appraisal of the work, as well as difficulties and examples of improvements and future work that could be done.

CHAPTER 2

State of the Art and Related Work

2.1 INTRODUCTION

This chapter's objective is to provide an overview of the current state of the art and work developed in areas related to this dissertation. We will explore the subjects of interaction devices and methods as well as enumerate some applications that relate to our fields of interest. On a first approach we will talk about interaction in general and its uses in a virtual environment, describing as well some interaction methods that are relevant to our work. Then we will discuss menus, their types and uses as well as navigation and manipulation methods, and what has proven, or not, to be the most appropriate in virtual environments.

2.2 INTERACTION IN VIRTUAL REALITY

Interaction can be described as the user's ability to alter the Virtual Environment, whether it be in terms of its parameters (Zeltzer, 1992), form and content (Schuemie and Mast, 1999), or movement and manipulation (Slater and Usoh, 1994) in real time. The ways in which the user interacts with the environment are called techniques of interaction, and those techniques are designed to to assist users in performing certain

tasks successfully. Even though the relationship between interaction and the user's sense of presence is not fully clear, research seems to indicate a positive correlation between them (Messinis et al., 2010). There are four universal tasks related to interaction in virtual environments, namely navigation, selection, manipulation and system control (Bowman et al., 2004), which we will succintly describe in the paragraphs that follow.

2.2.1 NAVIGATION

Navigation within the context of Virtual Reality is the act of moving from one place to another in a virtual world and can be divided in two components: wayfinding and travel. Wayfinding is the cognitive process of finding a path through an environment using spatial knowledge about it. Travel is the locomotive process of moving in the world (Lowgren et al., 2014).

In this work we explore three options of navigation in the virtual world, using physical controllers (such as a keyboard or a WiiMote), using head tracking via a Microsoft Kinect depth camera, and using a gesture-based steering wheel.

Direction of motion in virtual environments can be classified in several ways (Mine, 1995):

- Hand directed
- Gaze directed
- Physical controls
- Virtual controls
- Object driven
- Goal driven

Hand directed motion comes in several flavors like pointing mode where the user points in the direction s/he wants to move, cross-hairs mode in which the user's finger determines the spot to move towards and dynamic scaling where the user scales the scene up or down in order to change positions within the scene.

Gaze directed motion can be done in several ways, including gaze directed flying, in which the user goes towards the direction he is looking at, or orbital mode where the user is constantly looking at an object and changing the gaze direction orbits the user around the object (depicted in figure 2.1).

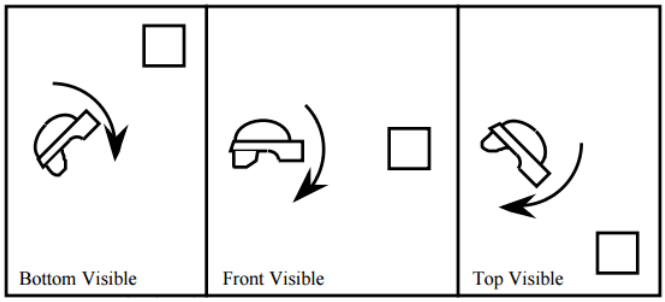


Figure 2.1: Orbital mode (Mine, 1995)

Physical controls consider the input from physical devices, such as a joystick, trackball, buttons, etc. in order to define a direction of motion in the virtual world.

Virtual controls utilize virtual devices (like a steering wheel of a flight stick) in order to control the motion in the world. An example of the steering wheel and handle bar virtual devices was developed by João Cardoso in his dissertation (Cardoso, 2015), and in the case of the steering wheel, also implemented in this work.

Object driven motion is not controlled by the user but by an object present in the virtual world, such as a self driving vehicle, and elevator or anything that moves the user in the virtual space (Mine, 1995).

Goal driven systems give the user a list of "goals" that represent destinations in a virtual world that a user can select to immediately fly to. These can be represented as text or icons within a virtual interactive menu (Mine, 1995).

2.2.2 SELECTION

Selection is the act of picking an option within a virtual world, for instance an object or menu to interact with. There are several methods for option selection within a scene, with various situational applications (Mine, 1995): Gaze-directed selection or pointing with an arm/finger/tracker can be used to select objects at a distance, while things such as virtual reality gloves can be used to select objects within reach.

Voice commands are another avenue of performing a selection within a virtual scene, and can be used as a stand-alone method of selection or in combination with pointing or some other method as a gesture to execute the selection action.

Selection from a list is also a valid way of selecting objects, and it has the advantage

of permitting selection of objects that the user cannot see. The drawback is that the user must know the name of the object he wishes to select beforehand, if the object isn't labeled in the scene or isn't in view of the user.

The rapMenu (roll-and-pinch menu) is a technique for freehand menu selection that uses an InterSense IS-900 tracker to detect hand orientation and Fakespace Pinch Gloves to detect pinches in order to select items on a 2D radial menu (Ryan, McMahan, and Bowman, 2008) (illustrated in figure 2.2).

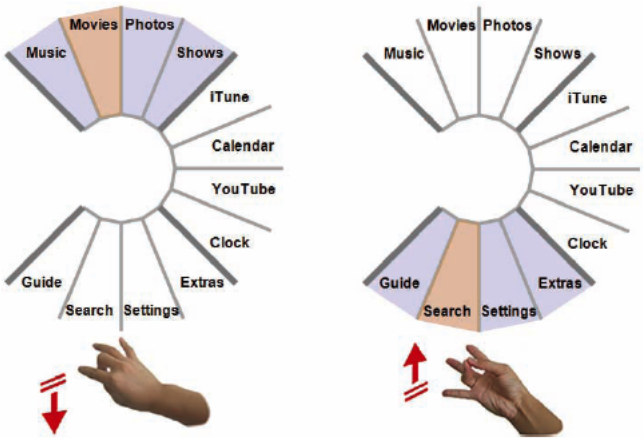


Figure 2.2: rapMenu (Ryan, McMahan, and Bowman, 2008)

2.2.3 MANIPULATION

Manipulation consists in the change of a property within a virtual scene, such as the scaling, position and rotation of an object, among other possible attributes.

One instance of such interaction is manipulating a cube by means of thumb gestures on a touch interface, proposed by (Tiefenbacher, Pflaum, and Rigoll, 2014), as seen on figure 2.3.

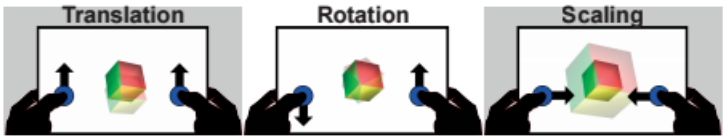


Figure 2.3: Touch based gestural manipulation (Tiefenbacher, Pflaum, and Rigoll, 2014)

Two methods of 3D manipulation are implemented in (Rodrigues et al., 2015), where

a couple of WiiMotes and a Kinect sensor are used (pictured in figure 2.4) to permit manipulation with 6 degrees of freedom.

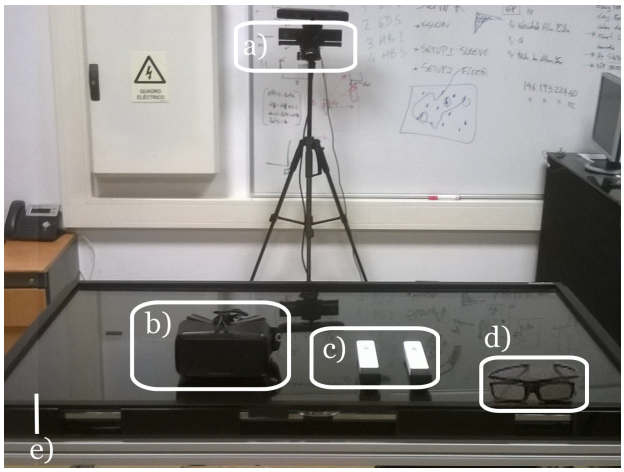


Figure 2.4: Mid air manipulation setup(Rodrigues et al., 2015)

These two methods, named 6DOF Hands and Handle Bar proposed in (Wang, Paris, and Popović, 2011) and (Song et al., 2012) can be seen in practice in figures 2.5 and 2.6.

The 6DOF Hands method uses 6 degrees of freedom tracking to rotate and position the object utilizing the orientation and the position of the hand in space. The Handle

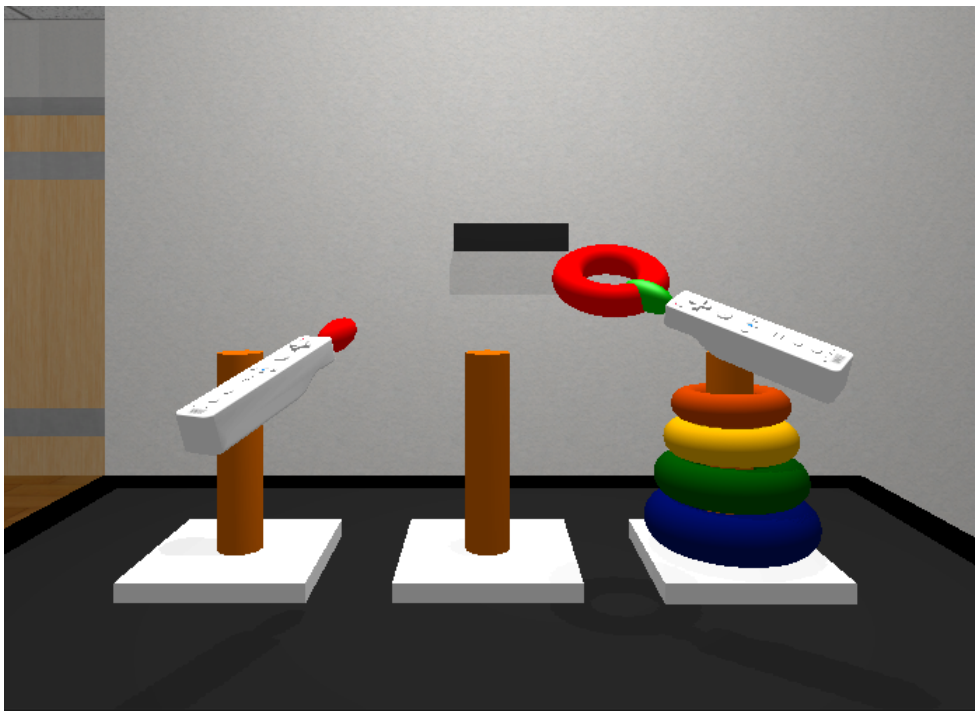


Figure 2.5: 6DOF Hand manipulation (Rodrigues et al., 2015)

Bar method determines the object position by the mid-point between the two hands (or

in this case the tips of the controllers), and the rotation is given by the relative hand position to a given axis.

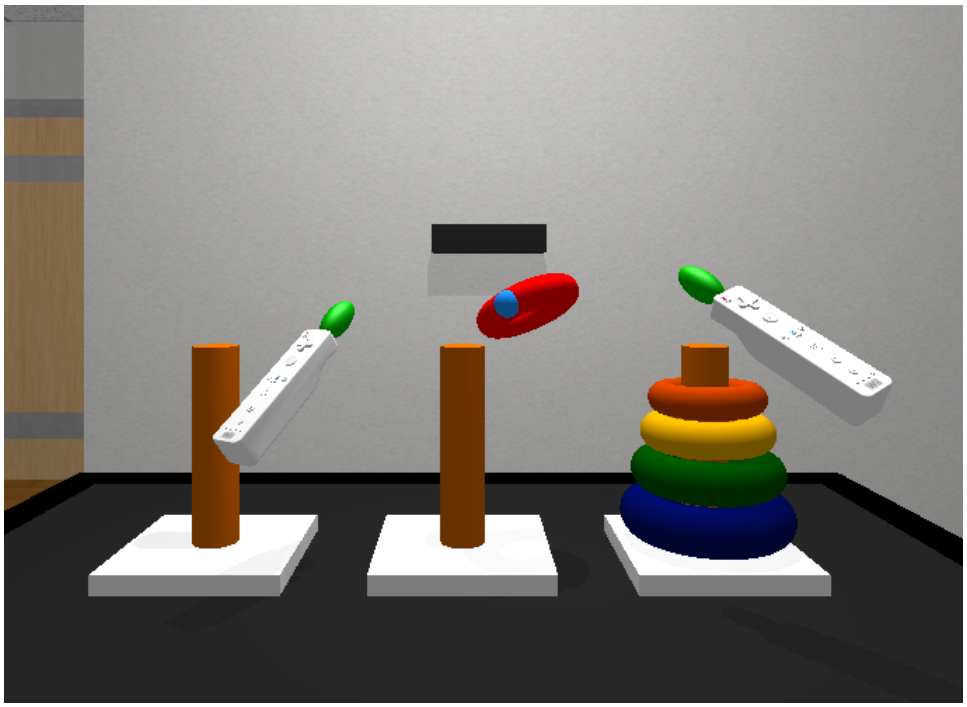


Figure 2.6: Handle Bar manipulation (Rodrigues et al., 2015)

From their experiments, they found that the 6DOF Hands technique was preferred by users, as they felt it was easier to interact and more fun, as well as having higher precision and smaller task completion time when compared to the Handle Bar technique, especially concerning rotations.

2.2.4 SYSTEM CONTROL

System control consists of tasks that alter the system's state or activate some functionality. Some examples of that are voice commands, menus and gestures (Bowman et al., 2004). Many examples of system control interaction are shown in the following section that describes menus.

2.2.5 MENUS

There are a multitude of ways to represent and interact with menus in a virtual environment. In this section we intend to enumerate and describe some of those ways.

ADAPTED 2D

Adapted 2D menus are a representation of standard 2 dimensional menus rendered on 3D geometry within the scene, and are the more prolific group of 3D system control techniques (Bowman et al., 2004), and can be present in multiple forms (linear, radial, etc.). Their placement can be classified as surround-fixed windows, which are placed in a fixed position in the world, display-fixed windows placed in a static location in the display, regardless of head rotation and world-fixed windows that are located in the world or on objects and are evocable by the user (Feiner et al., 1993).

Pull-down menus rendered as thin rectangular objects were used in Teleop , a virtual environment telerobotic application (Richard H. Jacoby, 1992), and interaction was based on gestures (pulling down the menu) and pointing at desired the menu option with an electronic glove and tracker (shown in figure 2.7).

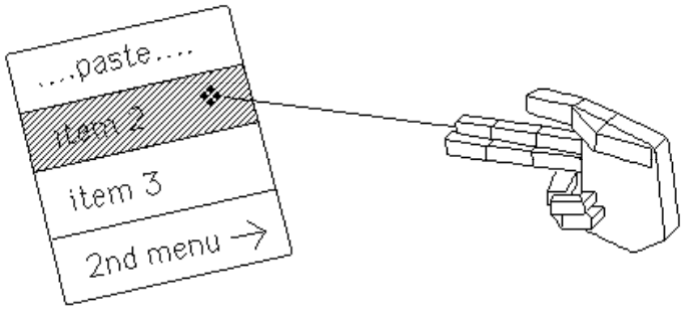


Figure 2.7: Pointing based selection (Richard H. Jacoby, 1992)

PEN AND TABLET METAPHOR

In an attempt to fix the problem of the lack of haptic feedback that a lot of input methods for virtual environments have, a physical tool (also known as prop) can be utilized. This physical tool has a duplicated representation within the virtual environment, and can take on multiple forms, from medical tools to a pen and tablet.

One example of the pen and tablet metaphor in use is the Personal Interaction Panel (PIP) (Zsolt Szalavari, 1997), shown in figures 2.8 and 2.9.

Figure 2.8: Physical Pen and Tablet (Zsolt Szalavari, 1997)

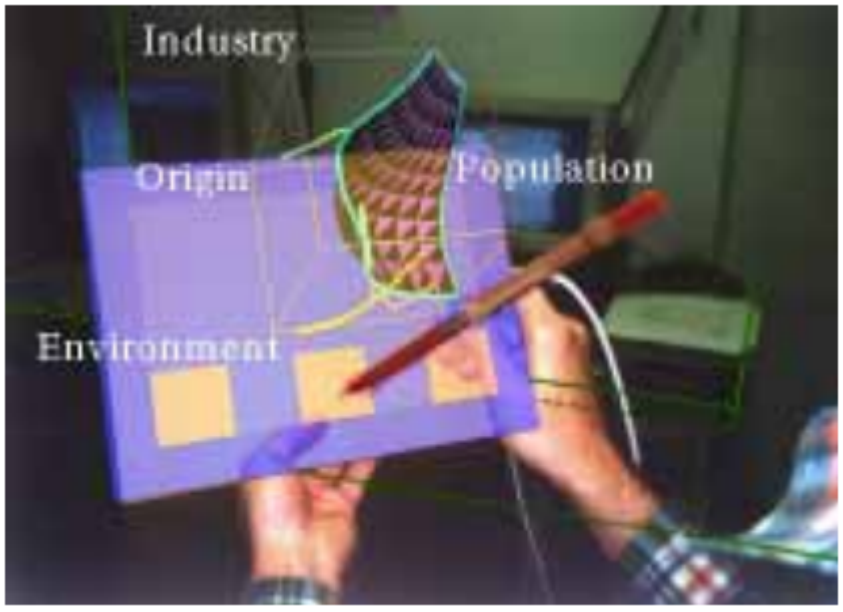


Figure 2.9: Augmented Reality tablet interface (Zsolt Szalavari, 1997)

CIRCULAR MENUS

Circular menus are a deviation from the typical rectangular shaped menus, displaying options radially around a point instead of listing options in a column, and providing

faster selection times and higher accuracy than their linear alternatives (Callahan et al., 1988) (Chertoff, Byers, and LaViola, 2009).

There are several examples of the use of circular menus in virtual environments, like the rapMenu (Ryan, McMahan, and Bowman, 2008) (shown previously in figure 2.2), the holosketch (Deering, 1995) and the extended pie menus (Gebhardt et al., 2013) (shown in figures 2.10 and 2.11, respectively).

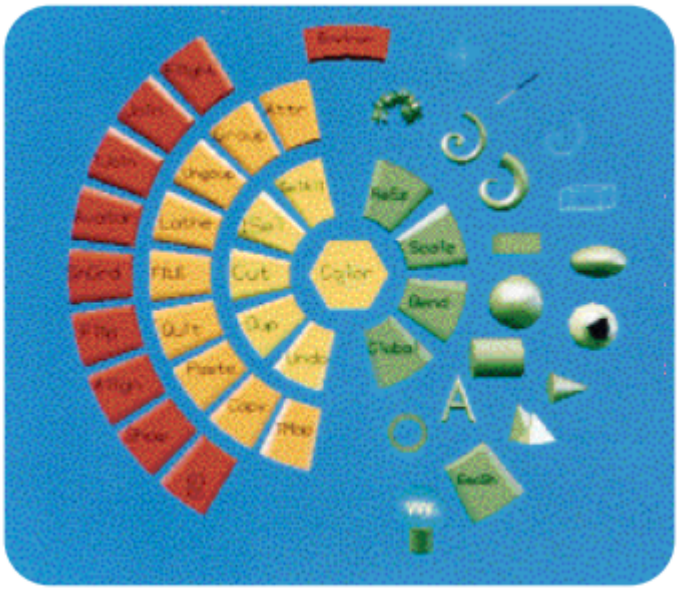


Figure 2.10: HoloSketch menu (Deering, 1995)

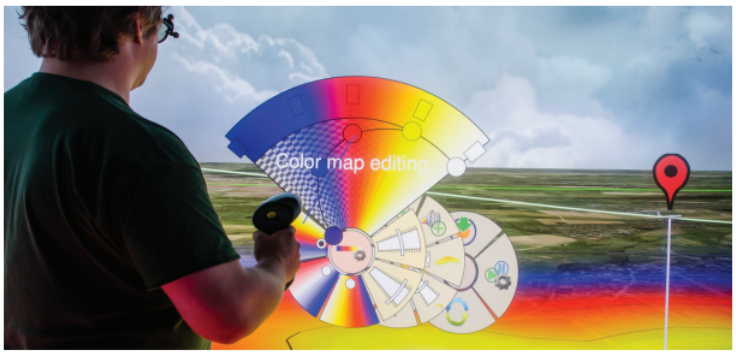


Figure 2.11: Extended Pie Menus (Gebhardt et al., 2013)

RING MENUS

Ring menus consist of 3D objects representing commands, distributed on a ring and facing the user's viewpoint. The Spin Menu (Gerber and Bechmann, 2005) is an

example of a ring menu that features hierarchy, with sub-menus being opened as a ring stacked on top of the initial ring (shown in figure 2.12).

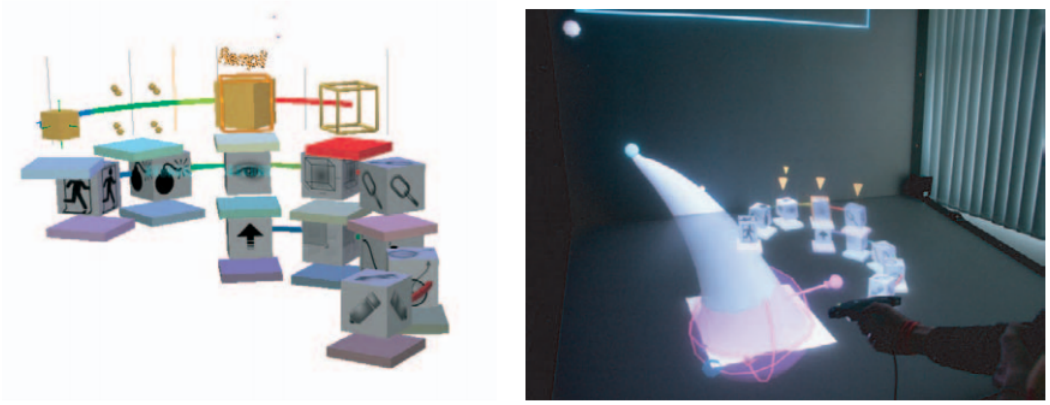


Figure 2.12: The Spin Menu (Gerber and Bechmann, 2005)

2.3 FRAMEWORKS

For the development of this project, we needed to choose the appropriate framework. Several factors weighed in on the decision such as versatility, ease of use, documentation, community activity and tracker/head mounted display compatibility.

2.3.1 GRAPHICAL ENGINES

OPENSCENEGRAPH

OpenSceneGraph (Wang and Qian, 2010) is an open source cross-platform toolkit written in C++ for graphics rendering, used in applications such as simulators, virtual reality and games that provides a higher layer of abstraction over the OpenGL library. It contains several libraries (NodeKits) that fulfill a variety of purposes, such as particle systems (osgParticle), text display (osgText), special effects (osgFX), shadow rendering (osgShadow), 3d interactive controls (osgManipulator), visual simulation (osgSim), terrain rendering (osgTerrain), animations (osgAnimation) and volume rendering (osgVolume). Unofficially, and because OpenSceneGraph is an open source project, there are useful libraries that interface with newer (Oculus Rift, with osgOculusViewer, Leap

Motion with osgLeap) and older hardware (through VRPN, with osgVRPN) developed by the community.

VISUALIZATION TOOLKIT

Unlike OpenSceneGraph, the Visualization Toolkit (VTK) (Schroeder, Martin, and Lorensen, 1998) is a non-hierarchical, open-source framework, maintained by Kitware, that abstracts the usage of OpenGL. Its native language is C++, but wrappers for Tcl, Python, Java, and C# are available. Instead of following a graph-based aproach like OpenSceneGraph, it instead relies on several types of entities, such as actors (that contain the object geometry information), light sources, particle systems, shadows and 2D/3D widgets. VTK has support for several GUI libraries such as Qt, FLTK, wxWindows, Tcl/Tk, Python/Tk, Java, X11, Motif, Windows, Cocoa, and Carbon, as well as various types of image files (png, jpeg, tiff, bmp and ppm), 3D objects (VTK's own xml, 3D Studio Max, obj, etc.) and image processing with a set of filters and effects.

UNITY

Unity [1] is a cross-platform game creation system that includes a game engine and an integrated development environment. It provides a graphic working environment that facilitates 3D scene creation and layout, and operates with a basis on assets, scenes and scripts. Unity utilizes both Direct3D an OpenGL, and as such can produce scenes with dynamic shadows, reflections and textures. It also has a physics engine that provides collision detection, forces, torque and physic materials with different properties. There are two licencing options for Unity, namely Unity and Unity Pro, the latter being a paid license with a higher number of features.

[1] https://unity3d.com/unity

UNREAL ENGINE 4

Unreal Engine 4 [2] is a multiplatform game development toolkit that supports state of the art graphical fidelity, as well as support for hardware such as the oculus rift, kinect and leap motion. Developers have full access to the source code of the engine, and development can be done either in C++ or using their Blueprint Visual Scripting system. As of 1/03/2015 Unreal Engine 4 abandoned its subscription based model and was released for free usage with a 5% commission on commercial products.

OPENSPACE3D

OpenSpace3D [3] is an open source platform for 3D virtual/augmented reality application development, designed with ease-of-use in mind. It provides a function-based visual "programming" system (PlugITs), multimedia features like video playing, sound and speech recognition, and web deployment. Supported hardware includes the Oculus Rift DK2, Leap Motion and Kinect.

2.3.2 SENSOR INTERFACES

VIRTUAL REALITY PERIPHERY NETWORK

Virtual Reality Periphery Network (VRPN) (Ii et al., 2001) is a system that provides a network-transparent interface between applications and physical devices used in VR systems using a client-server approach (the client being the application, and the server being the interpreter of input from the physical devices). It provides a layer of abstraction that classifies inputs into several categories (Analog, Button, Dial, ForceDevice, Sound, Text, and Tracker), which allows us to receive generic input from different devices.

FLEXIBLE ACTION AND ARTICULATED SKELETON TOOLKIT

The Flexible Action and Articulated Skeleton Toolkit (FAAST) (Suma et al., 2011) is a middleware that aims to facilitate integration of full-body tracking with games and

[2] https://www.unrealengine.com/unreal-engine-4
[3] http://www.openspace3d.com/

VR applications using OpenNI [4] or the Microsoft Kinect skeleton tracking software. FAAST utilizes a custom VRPN server to stream up to four user skeletons over a network, allowing VR applications to read the skeletal joints as trackers using any VRPN client. Additionally, the toolkit permits keyboard input emulation triggered by user defined body-based control mechanisms (body posture and specific gestures), allowing for the control of existing off-the-shelf applications that do not have native support for depth sensors.

VR JUGGLER

VR Juggler (VRJ) (Bierbaum et al., 2001) is a free and open source application framwork that provides C++ classes for VR Application creation, with support for several graphic APIs, such as OpenGL, OpenSG and OpenSceneGraph. It provides an abstraction layer between the hardware of a VR system (such as an HMD or tracker), and the virtual world created in software. VRJ supports several modules, designed to provide flexibility to be used in a variety of VR systems. Among those modules, we have the VR Juggler Portable Runtime (VPR), designed to provide an abstract layer between VRJ and the operating system, Tweek, that allows communication between a Java user interface and a C++ application, the Juggler Configuration and Control Library (JCCL), that provides an XML-based configuration system for the components, and finally the Gadgeteer module that handles the hardware input from VR devices, as well as interfacing with VRPN to extend the list of supported devices.

2.3.3 CONCLUSION

From the several frameworks that we have analysed, a few fit our requirements. OpenSceneGraph was the prime candidate, as the previous platform used it, and it now features support for all of the hardware and software that we require to develop our application (osgOculusViewer and osgVRPN, namely). The Unity3D platform was another candidate that fit our requirements, but certain core features that we needed (such as animated textures for video playback) were locked behind a paywall, and as

[4] http://structure.io/openni

such this option was discarded. Unreal Engine 4 also fit our requirements, however it was released as a free license too late into our development cycle, after we had already begun development using OpenSceneGraph. An attempt of utilizing OpenSpace3D to develop a prototype was made, but the platform was unstable, and as such was discarded as an option. The Visualization Toolkit was another option that we had worked on previously, and although it was not used in the main application as the other options had better support for the hardware that we intend to use (Oculus Rift and virtual reality peripherals), it was still utilized in some of our auxiliary programs that utilize its Iterative Closest Point to calibrate our kinect cameras (described in chapter 3.2.1). While FAAST provides a VRPN solution to the Kinect camera, it does not support gestures that were deemed crucial to our application, such as the detection of the user's grip. As such, we required another solution to interface the Kinect with VRPN. We describe that solution in chapter 3.2.2.

2.4 VIRTUAL REALITY HARDWARE

In this section we will enumerate and categorize some of the harware that makes virtual reality possible, namely displays and trackers.

2.4.1 DISPLAYS

Displays are output devices that present information in a visual form. They come in a wide variety of shapes and sizes, but in this subsection will will describe the two most common types of display in virtual reality; Head-Mounted Displays and CAVEs.

HEAD-MOUNTED DISPLAYS

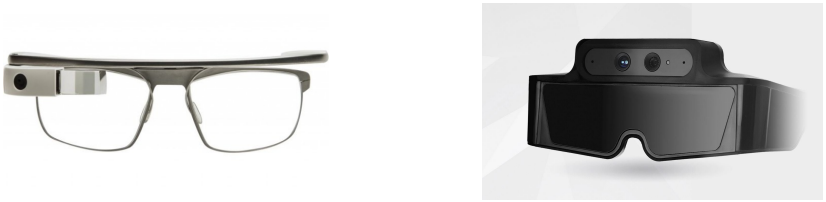
Head-mounted displays are one of the most commonly used type of display in Virtual Reality applications (Burdea and Coiffet, 2003) A head-mounted display, or HMD, is a display device worn on the head, or mounted on a helmet, that contains one or several screens in front of one (monocular HMD) or both eyes (binocular HMD). Several

new high-tech HMDs have been announced in the last couple of years, such as Sony's Morpheus [5], Oculus' Oculus Rift [6] and HTC's Vive [7] (depticted in figure 2.13).

(a) Morpheus (b) Oculus Rift (c) Vive

Figure 2.13: Head-mounted displays

HMDs that are see-through and utilize reflected projections are called Optical head-mounted displays (OHMD). Two examples of OHMD's can be found in figure 2.14, which depicts the Google Glasses [8], by Google, and the MetaGlasses [9], by Meta.



(a) Google Glasses (b) MetaGlasses

Figure 2.14: Optical head-mounted displays

CAVE AUTOMATIC VIRTUAL ENVIRONMENTS

The other more commonly used type of display in Virtual Reality applications is the Cave Automatic Virtual Environments (CAVE) (Burdea and Coiffet, 2003), shown in in figure 2.15. It consists of a room where the entire surface area is covered with high resolution projections that generate rapidly alternating images for each eye, allowing any number of users in the room to perceive the visual data. Stereoscopic shutter glasses are synchronized with the projectors in order for the user to only see the images for the correct eye, which enables depth perception. The projectors are placed outside of the room, and are controlled by one or more computers.

[5] https://www.playstation.com/en-us/explore/project-morpheus/
[6] https://www.oculus.com/en-us/
[7] http://www.htcvr.com/
[8] https://www.google.com/glass/start/
[9] https://www.getameta.com/

Figure 2.15: Multiple users in a CAVE (Craig, Sherman, and Will, 2009)

2.4.2 TRACKERS

Tracking devices are crucial in a virtual reality system. They provide input data into the system in the form of positioning (relative or absolute) and orientation that can be used in a variety of ways to immerse the user in a virtual world or provide interaction. In this subsection we will enumerate and describe several different types of trackers, as well as their advantages and disadvantages.

OPTICAL TRACKERS

This kind of tracking uses light, usually by recognizing the location and orientation of markers, but not necessarily. There are three types of markers, passive, active or visible markers with predefined patterns; Passive markers are made of a retro-reflective material, reflecting light sent by the camera (an example of one such camera is used in the ARTTRACK 5 [10] system by Advanced Realtime Tracking (ART), depicted in figure 2.16), and active markers emit beams of light, usually infra-red, directly into the camera, usually in periodic flashes synchronized with the camera in order to reduce interference from other infra-red light sources. The LaserBIRD 2 system [11], shown in

[10] http://www.ar-tracking.com/products/tracking-systems/arttrack-system/arttrack5/
[11] https://www.vrealities.com/products/motion-trackers/laserbird-2

figure 2.17, by the Ascension Technology Corporation, is an example of the use of active markers.

Figure 2.16: Camera with IR lights used in a passive tracking system

Figure 2.17: LaserBIRD 2 system for head/object tracking

Visible markers with predefined patterns can be used for optical tracking, and can vary in shape and size as long as they are able to depict a large number of distinct patterns that can be easily identified by cameras in a computationally efficient manner.

Marker-less tracking can be done by recognizing features of an object, like a face of a body, or by effectuating a comparison with a known 3D model. A real-time depth map obtained by a camera can also be utilized in order to recognise and extract the object that is being tracked, and analysing it to obtain the positional data.

MAGNETIC TRACKERS

Magnetic trackers have a stationary transmitter that produces a magnetic field, and a receiver. Positional data is determined by the strength of the magnetic field, as it decreases the further away the receiver is from the transmitter. Rotational information is gathered from the changes in the distribution of the magnetic field across all axes caused by the rotation of the receiver. An example of a magnetic tracker is the Razer Hydra [12], by Razer, depicted in figure 2.18.

Figure 2.18: Razer Hydra magnetic tracker

As it uses magnetic fields in order to gather the positional and rotational data, metallic objects or any kind of device that produces a magnetic field can cause interference. As such, in order to improve accuracy, special care is needed when it comes to the placement of the receiver. HMDs or electronic shutter glasses, which are often paired with tracking devices for virtual or augmented reality applications, can cause interference, and that should be taken into consideration for the placement of the receiver (Craig, Sherman, and Will, 2009).

MECHANICAL TRACKERS

The first type of tracker used in Virtual Reality (Burdea and Coiffet, 2003), mechanical tracking systems are still used today due to being fast, with accurate position tracking and no need for calibration. They consist of a serial or parallel kinematic structure composed of links interconnected using sensorized joints, attached to a fixed point in space. They track one end of the mechanical tracker relative to the other using the real-time information given by the sensor joints, which means that attaching one

[12]http://www.razerzone.com/gaming-controllers/razer-hydra-portal-2-bundle

end of the tracker to a chair or a table and the other to an object allows the computer to calculate and track the object's 3D position. The BOOM display (figure 2.19), manufactured by Fake Space Labs, is an HMD connected to an articulated mechanical arm (the mechanical tracker)

Figure 2.19: BOOM Display (Craig, Sherman, and Will, 2009)

Mechanical trackers do not suffer from the problems of interference that electromagnetic trackers can suffer from, and do not have a problem with visual occlusion of the tracked object, unlike optical trackers.

ULTRASONIC TRACKERS

Ultrasonic Trackers work by measuring the time that an ultrasonic signal emitted from a transmitter (a speaker) takes to arrive at a the receiver (a microphone), and using that information to calculate the distance between the two. By utilizing several pairs of transmitters and receivers, it is possible to calculate the orientation and position of the object. The use of sound is inherently problematic when it comes to interference from obstacles in the path of the projected sound waves, as well as any sounds emitted by other sources. One such tracker, along with a mechanical tracker, was used in what is considered to be the first virtual reality head-mounted display, Ivan Sutherland's Sword of Damocles (Sutherland, 1968), and can be seen in figure 2.20.

Figure 2.20: Ultrasonic position sensor (Sutherland, 1968)

INERTIAL TRACKERS

This type of trackers utilize accelerometers and gyroscopes to calculate position and orientation, respectively. The InertiaCube 4 [13], by Thales, shown in figure 2.21, is an example of a 3DOF (pitch, yaw, roll) inertial tracker.

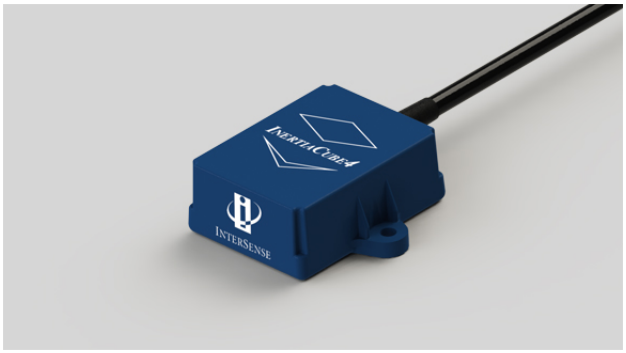


Figure 2.21: InertiaCube 4 intertial tracker

Some of the problems with this type of trackers are the relative nature of the measured movement, the compounding error of the gyroscopes which cause drift, and the sensitivity of accelerometers to sudden variations in acceleration, which cause accumulated errors in the position readings, lowering the accuracy of the measurements.

[13] http://www.intersense.com/categories/18/

These problems mean that these trackers lose accuracy over time, and must be re-calibrated often. On the flip side, they do not suffer from any interference related problems, nor do they require transmitters.

2.5 RELATED WORK

2.5.1 PLATFORM FOR SETTING UP VIRTUAL ENVIRONMENTS

pSIVE is a platform that was developed in the scope of a Master's Thesis (Souza, 2013), and can be used to set up a virtual scene using a diversity of models, and those models can have a variety of content (PDF, Video, Text) attached to them. It uses OpenSceneGraph (Wang and Qian, 2010) as a graphical engine and VRJuggler (Bierbaum et al., 2001) as a middleware to interpret input from trackers and controllers, which are used to define orientation within the scene and interact/navigate, respectively.

Figure 2.22: PDF display in pSIVE (Souza, 2013)

Content can be accessed through a 2D linear menu that pops up when the user presses a button on his controller while looking at an object that has been configured with content.

Figure 2.23: pSIVE linear menu (Souza, 2013)

This framework, however, lacked certain features that made truly immersive virtual visits difficult (such as the necessity of a physical controller for movement in the environment), and lacked support for gestural ways of manipulating objects in the virtual environment, as well some of the newer HMDs.

2.5.2 THE EMPTY MUSEUM

The Empty Museum (Hernandez et al., 2003) is perhaps the application in literature that most closely resembles the application we have used as proof of concept, the "Imaginary Museum" presented in chapter 5, thematically and practically. It features a multi-user immersive walkable and wireless system where users can navigate a virtual environment with content such as audio, text, images and video. The system architecture consists on a laptop computer in a backpack that renders the environment according to the user's perspective, connected to a system that captures wireless movement (Foxlin, Harrington, and Pfeifer, 1998) and using virtual reality goggles (setup shown in figure 2.24).

Figure 2.24: Empty Museum Hardware Setup (Hernandez et al., 2003)

When compared to our system, The Empty Museum has the advantage of supporting multiple users at the same time, with 3D avatars within the virtual environment. Its interaction, however, is solely determined by the user's position (seen in figure 2.25), while we support button input for interaction, as well as gestures. Their tracking system must also be mounted on the ceiling, while ours is easy to mount and easily expandable by setting up more kinects.

Figure 2.25: Interaction with virtual objects in the empty museums (Hernandez et al., 2003)

2.5.3 VENLAB

Brown University's VENLab (Tarr and Warren, 2002) is an immersive virtual reality space, with a 12m^2 walkable area that uses an IS-900 head tracker to measure the user's position in real time, and an 80 degree field of view HMD to fully immerse the user in the virtual environment (figure 2.26).

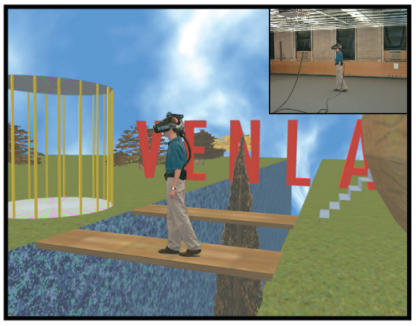


Figure 2.26: User walking in the VENLab space (Tarr and Warren, 2002)

2.5.4 DCNS TRAINING SCHOOL

DCNS Training School is a virtual training environment, developed in OpenSpace3D, linked to a Learning Management System (LMS) developed by STUDEC (*STUDEC*). This LMS is a web solution that provides access to training courses and pedagogical progress of students, as well as managing the content of the courses. Users can navigate in the environment, visualise training courses, media and products, check messages and chat with other users (students and professors) (Bastien, 2013).

This application can be used in a desktop environment, in a web browser using the mouse and keyboard, or in an immersive environment, with stereoscopic video projection on a wall, where the user navigates using a WiiMote and interacts with content utilizing a Kinect camera.

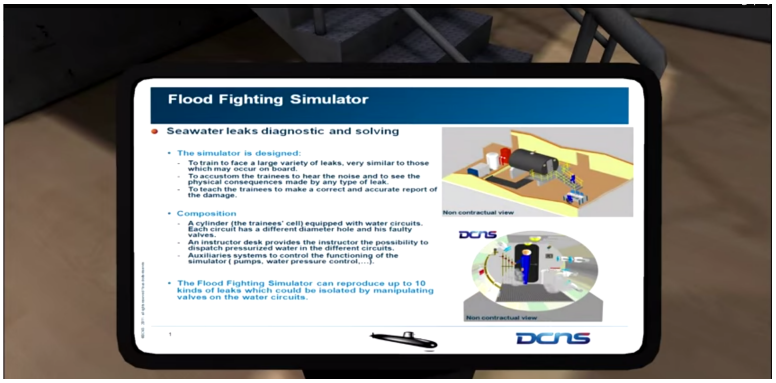


Figure 2.27: DCNS environment (Bastien, 2013)

2.5.5 THE KIDSROOM

The KidsRoom's (Bobick et al., 1999) aim is to explore computer vision technologies in an augmented reality interactive playspace for children. To do so, it recreates a child's bedroom composed of two real walls and two large video projection screens where images are projected from outside of the room, as well as ceiling mounted colored lights, speakers, cameras and a microphone, all controlled by a six-computer cluster. There

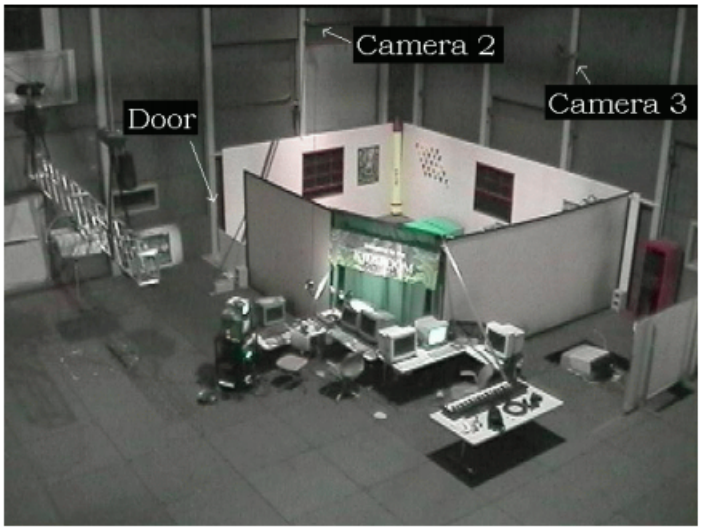


Figure 2.28: KidsRoom bedroom setup (Bobick et al., 1999)

are 4 cameras pointing at the room, one for tracking people, two for action recognition of people in the room, and one to provide a view of the room for spectators (Bobick et al., 1999).

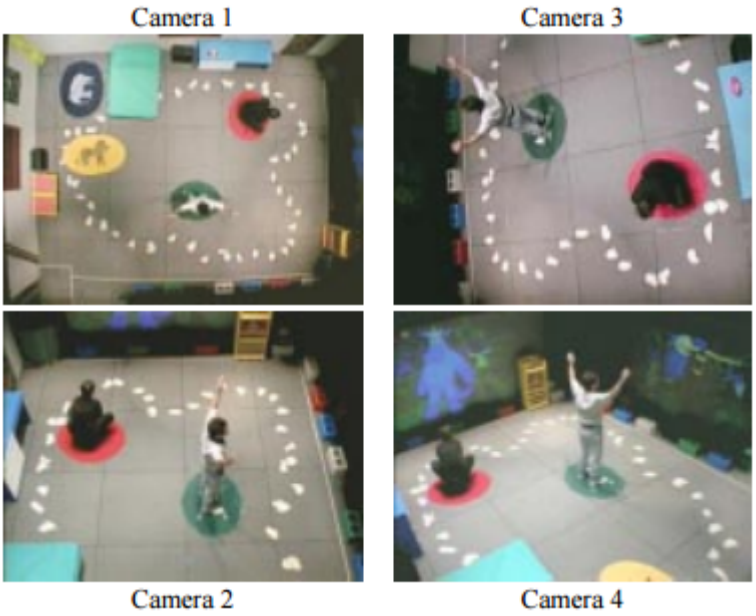


Figure 2.29: KidsRoom camera setup (Bobick et al., 1999)

CHAPTER 3

Framework Developed

3.1 architectural decisions

In this section we will enumerate the options we considered and the decisions taken for the design and architecture of the developed framework.

3.1.1 framework choice

During the early development of the project, two main frameworks were considered: OpenSceneGraph and Unity.

OpenSceneGraph (in conjuntion with VR Juggler) had already been used in pSIVE, and its development and community are still active to this day. Libraries such as osgVRPN and osgOculusViewer further increased the inclination towards this framework, as they provided all of the necessary utilities to meet all of the project's requirements, and as such this is the graphical framework we decided to use.

We have chosen to not use VR Juggler as we found it to be complex to use and configure, and the project seems to not have been updated since late 2013. Instead we use the osgVRPN library as a bridge between the application and the VRPN Server that listens to the hardware.

This choice gives us the freedom to use any hardware natively supported by VRPN. For our internal testing, we used a Microsoft Kinect and a WiiMote.

The osgOculusViewer library was also chosen due to its direct integration with the Oculus Rift, its SDK and OpenSceneGraph.

Unity provided a lot of easy to use features and VR hardware integration, but the commercial license was needed in order to implement some of the required features, such as video playback and VRPN integration, and in the end proved not to be a viable alternative.

3.1.2 MODELING TOOL CHOICE

As this project is a continuation of work developed in pSIVE, any format supported by OpenSceneGraph can be used (listed in table 3.1[1]), as well as any modeling tools that can export in those formats. For the purposes of modeling the testing environment, Sketchup2015 was used to create a scene and acquire the models that decorate the scene.

Extension	Description
.3dc .asc	3DC point cloud reader
.3ds	3D Studio
.ac	AC3D modeler
.bsp	Quake3 BSP
.dae	COLLADA 1.4.x
.dw	Design Workshop Database
.dxf	Autodesk DXF Reader
.fbx	Autodesk FBX
.gem .geo	Geo
.iv .wrl	Inventor
.ive	Native osg binary
.logo	Logo database
.lwa .lw .geo	Lightwave Object
.lws	Lightwave Scene
.md2	Quake MD2
.obj	Alias Wavefront
.ogr	
.flt	Multigen Openflight

Table 3.1: Supported OpenSceneGraph model formats

[1]http://trac.openscenegraph.org/projects/osg//wiki/Support/UserGuides/Plugins

3.1.3 SYSTEM DESIGN

In order to provide a fully immersive VR experience with skeleton tracked navigation in mind, the framework needed to be designed to be easily expandable in terms of area covered by the depth sensor(s). As such, we have decided to use a PC based client-server architecture in which the client is responsible for all of the rendering of the virtual world and handling of the Head Mounted Display (HMD) orientation tracking, and the VRPN servers are used to communicate the hardware input, when using a physical controller, or the user's skeleton information (positional data of head, hands and gripping gestures) to the client using one or several Microsoft Kinect devices. Skeleton data is collected with the Kinect SDK 1.8 [2], in the case of gesture based interaction or skeleton tracked navigation (illustrated in figure 3.1).

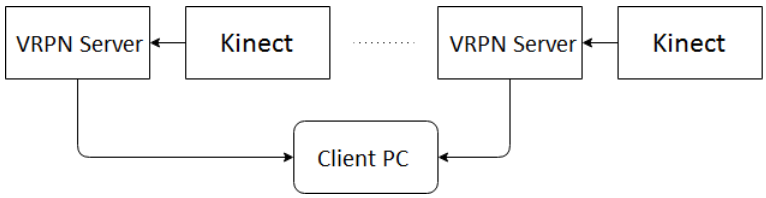


Figure 3.1: Client-server setup for multiple Kinects

The framework's general architecture and workflow is detailed in figure 3.2. It is configured with several XML files (detailed in section 3.1.4) and receives input information from one or more VRPN Servers, which is then interpreted by the osgVRPN library. That information is then handled in one of two ways: interaction with menus or content (Menu Handler), or navigation (Camera Handler). The scene is then rendered for use with the Oculus Rift using the osgOculusViewer library, which uses the Oculus SDK.

[2]https://www.microsoft.com/en-us/download/details.aspx?id=40278

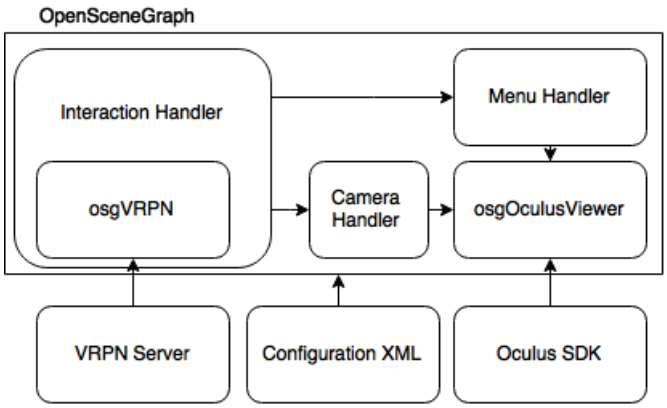


Figure 3.2: Application Architecture

3.1.4 SYSTEM CONFIGURATION

The platform, composed by the client which renders the scene and tracks the user's head orientation, and the server(s) which track(s) the user's body or read input from a physical controller, are configured through 3 XML files.

- Config.xml configures the scene by loading a list of objects and their content with configurable scale, position and rotation. Its general structure was maintained from pSIVE, and as such previously configured scenes are fully compatible.
- Kinect.xml configures the custom VRPN Kinect servers to listen to, using their server name and IP address.
- Controls.xml configures the navigation input buttons, allowing configuration of directional input and menu activation input.

SCENE CONFIGURATION

The Config.xml file consists of a list of models, their physical attributes (size, rotation, location), and their available content.

- Filename: Path to the model.
- Label: Model description that is shown when an object is interactive.
- Rotate/Translate: User defined object rotations and translations in all axis.
- Context: Whether the object is interactive or not.

In that same file, each model has a list of available content with the following properties:

- Type: From 0 to 3, Text, PDF, Video and Image data types, respectively.
- Label: Menu entry label.
- userContent: In case of the Text data type, the text to display. In all other data types, it is the path to the content to be shown.

```
<layout>
  <data>
    <model filename="C:\Users\J\Desktop\IEETA\IEETA\Modelos\Teste.osg"
        label="sala" xRotate="0.0" yRotate="0.0" zRotate="0.0"
        xTranslate="0.0" yTranslate="0.0" zTranslate="0.0" context="false">
    </model>
    <model filename="C:\Users\J\Documents\plant_test.obj" label="House␣Plant"
        xRotate="0.0" yRotate="0.0" zRotate="0.0" xTranslate="0.0"
        yTranslate="0.0" zTranslate="0.0" context="true">
      <userDatas>
        <userData>
          <type>0</type>
          <label>Name</label>
          <userContent>Abelia</userContent>
        </userData>
        <userData>
          <type>3</type>
          <label>Flower</label>
          <userContent>C:\Users\Public\Pictures\Sample
              Pictures\Chrysanthemum.jpg</userContent>
        </userData>
      </userDatas>
    </model>
  </data>
</layout>
```

Listing 1: Scene configuration

SERVER CONFIGURATION

The client can be set up to listen to multiple servers which read data from multiple Kinects, in one or more computers in order to achieve coverage of a larger area. These servers must be listed in the kinect.xml file with the following format:

```
<layout>
 <data>
  <kinect>
   <tracker>
    Kinect1@192.168.0.5
   </tracker>
  </kinect>
  <kinect>
   <tracker>
  Kinect2@localhost
   </tracker>
  </kinect>
 </data>
</layout>
```

Listing 2: Kinect server configuration file

In this example, Kinect1@192.168.0.5 and Kinect2@localhost are two different servers running on two different machines, one being the local host (127.0.0.1), and the other being a remote machine with the IP address 192.168.0.5.

CONTROLLER CONFIGURATION

Controls for the use of a physical controller are defined in the Controls.xml file. This file uses the ID's of the hardware device inputs, as well as the VRPN server to read the inputs from.

```
<layout>
  <data>
    <controls>
      <tracker>Tracker0@localhost::3884</tracker>
      <forward>31</forward>
      <back>17</back>
      <left>30</left>
      <right>32</right>
      <activate>18</activate>
      <switchmodeleft>33</switchmodeleft>
      <switchmoderight>34</switchmoderight>
    </controls>
```

```
  </data>
</layout>
```

Listing 3: Controller configuration

In this sample configuration file, we have configured our application to work with the keyboard; Tracker0@localhost::3884 indicates the name of the VRPN server (Tracker0), its IP address (in this case, localhost represents 127.0.0.1), and its port, 3884. If no port is indicated, the default port is used (3883). This option is necessary in order to run multiple VRPN servers in a single machine. The controls themselves are divided in several categories representing movement directions (forward, back, left and right), as well as the activating button for menus and two buttons for switching the manipulation mode within the application. The IDs that the configuration file uses are obtained using a vrpn client that prints the device information (number of buttons, button IDs, number of channels, etc.) from any given server.

3.2 IMPLEMENTATION

In this section we delve into the technical details of how our different applications function, the challenges we encountered and the solutions we found. First, we give an overview of how we calibrate and position the user in our virtual scene when we use the Kinect sensor. Then, we talk about how that information is conveyed from our server to our client application.

Finally, we discuss technical aspects of the implementation of our client application, in terms of navigation, interaction, and display of information.

3.2.1 KINECT CALIBRATION TOOLS

In order to transform the Kinect's coordinate system into our world's coordinate system, two things are needed: A frame from the Kinect depth camera (as seen on figure 3.3), and a fairly accurate 3D model of the physical space that is being captured

by the Kinect.

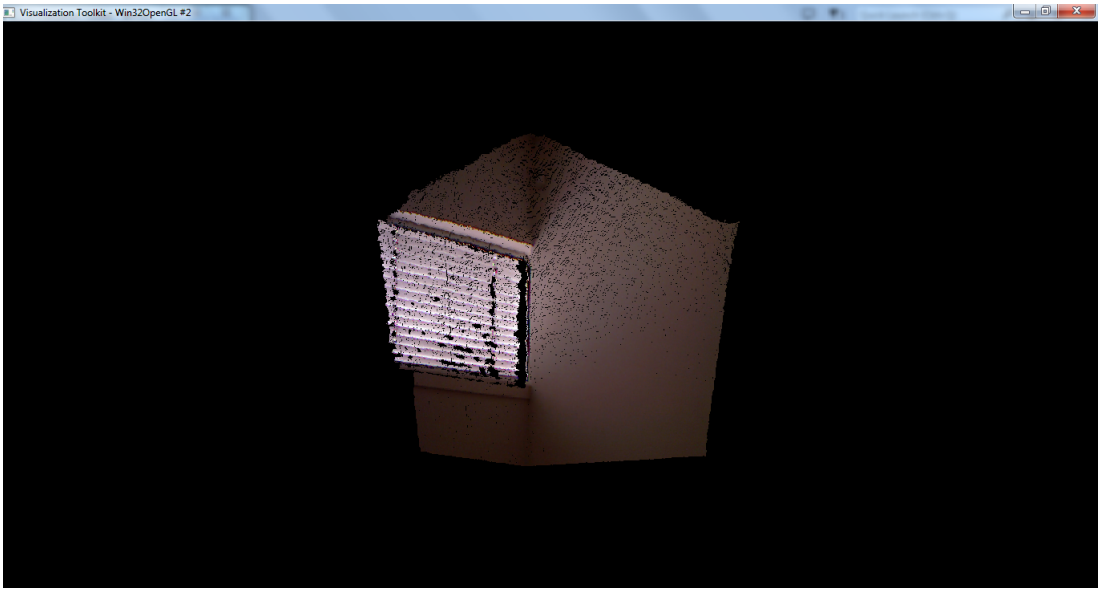


Figure 3.3: Frame taken from a Kinect Sensor

In order to form a 4x4 transform matrix that represents the transform from Kinect coordinates to our world coordinates, two programs were developed using the VTK library. Our choice to use VTK comes from a combination of having some previous experience with the library as well as needing an algorithm to match the point cloud captured from the Kinect sensor with the 3D model of the room. For that purpose, we use an Iterative Closest Point algorithm (Besl and McKay, 1992), that is already implemented in the VTK library.

The first program captures a depth frame from the Kinect, converts it into a VTK PolyData object (each point in the cloud is converted into a vertex with an associated color array) and exports it in VTK's VTP format (which is the corresponding format for storing vtkPolyData data types) for later use.

The second program reads that depth frame and the 3D model of the room and allows the user to roughly fit the depth frame within the 3D model, using the keyboard to position and rotate the depth frame. After the frame is positioned manually (shown on figure 3.4), the user can perform the calibration, either manually using the position the user chose, or automatically by pressing a key the frame's position is adjusted within the model using VTK's ICP function (the result of which is shown in figure 3.5).

When the frame and the model overlap, a file containing the 4x4 transformation matrix is exported, ready to be used as input in our custom VRPN server.

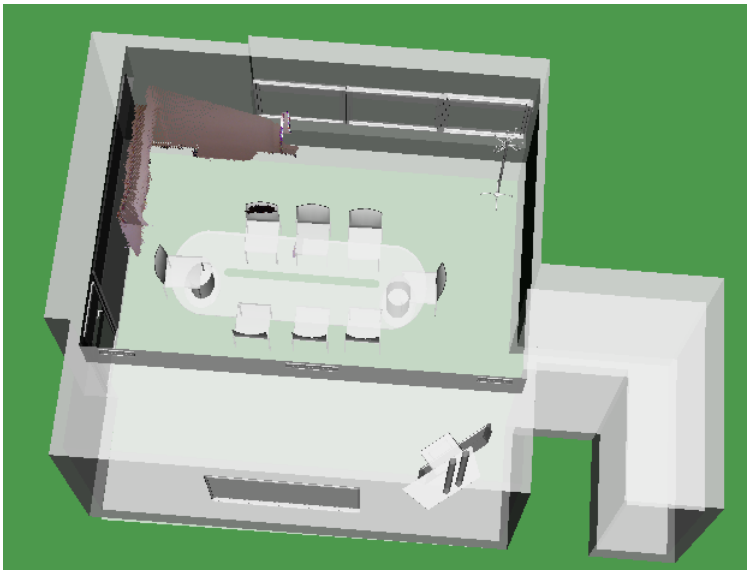


Figure 3.4: Manual placement of the frame within the model

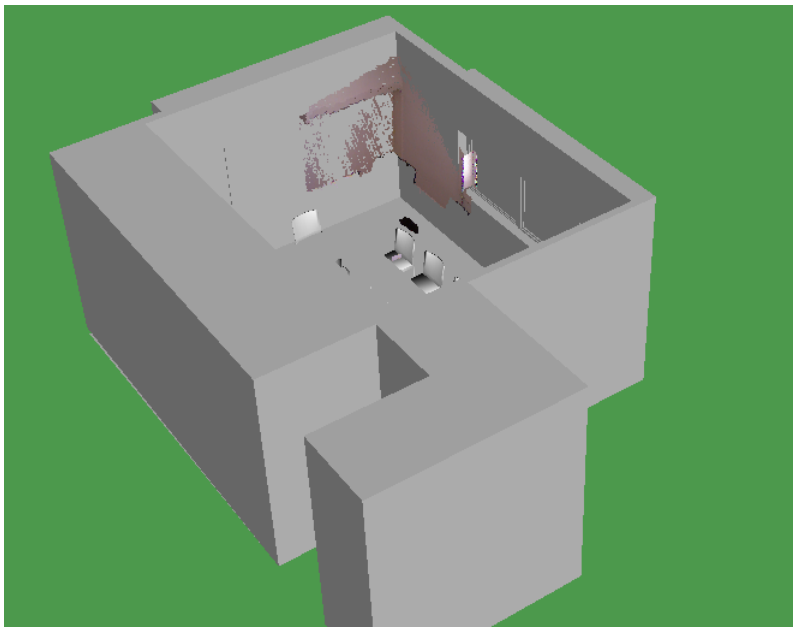


Figure 3.5: Position after ICP

3.2.2 CUSTOM VRPN SERVER

In order to integrate the Kinect with our system, we developed a custom VRPN Server using the VRPN library in conjunction with the Microsoft Kinect library.

The server uses a Kinect, and the Kinect 1.8 SDK, to track the user's skeleton and the grip gesture of both his/her hands, and transforms the coordinates obtained using the transformation matrix mentioned in the last section. All positional information is conveyed after being transformed into virtual world coordinates using the calibration method described in section 3.2.1. That information is then relayed to the client with the following format:

- Analog channels 0, 1, 2 send the user's head position
- Analog channels 3, 4, 5 send the user's left hand position
- Analog channels 6, 7, 8 send the user's right hand position
- Analog channels 9, 10, 11 send the Kinect's position
- Button 0 sends the state of user's left hand grip
- Button 1 sends the state of user's right hand grip
- Button 2 sends the state of the skeleton tracking

The head position is necessary in order for the client to position the camera in the right place within the scene, and the right hand and left hand positions are used to represent the hands in the scene as a visual aid. The buttons are used to convey three binary inputs, representing the left and right hand grip, and whether the users skeleton is detected. Buttons 0 and 1 are used for several interactions in the client, and button 2 is particularly important when multiple Kinects are in use, so as to know which one is detecting the user and sending the correct coordinates. The left and right hand grip inputs are detected using the Kinect 1.8 SDK, and are used in most of the hands-free interactions.

Since the server gives us the head position calibrated for our 3D world, as explained in section 3.2.1, but not its orientation, we needed a reference frame in order to calibrate the Oculus Rift orientation to match what the direction the user is looking at in the virtual world with the direction s/he is looking at in the real world.

To do so we require that the user stands straight facing the real world Kinect, establishing a point of reference that is known in the virtual world (which is given by channels 9, 10 and 11 of the custom VRPN server), and grip his/her right hand in order to overlap his/hers view direction of the 3D world with the direction s/he is facing in the real world.

After this calibration process, the user should be facing a white cube that represents the position of the Kinect in the room (shown in figure 3.6), and can then freely walk in the room, with one to one positioning to the virtual room.

Figure 3.6: White cube representing Kinect (pointed by red arrow)

3.2.3 CLIENT APPLICATION

The client application was developed with the requirements of pSIVE in mind, and as such borrowed some of its aspects (namely scene and content configuration), while implementing some of its features in different ways, such as the usage of osgVRPN instead of VRJuggler to receive hardware input, and a rework of the menu system due to incompatibilities in OpenSceneGraph versions. An adaptation of pSIVE to a more recent OpenSceneGraph version was needed in order to use the osgOculusViewer library, which resulted in an entire rework of the previously existing framework. The usage of the Oculus Rift also led to some changes in design due to several factors:

- The lens distortion that the Oculus requires makes it hard to read text closer to the edges of the screen. To alleviate this issue, font size was reduced to accommodate

larger quantities of text, and the menus and content were placed in the center of the screen.

- 2D projections for displaying text were not rendered properly when distorted for use with the Oculus Rift, and were displayed off-center and mismatched when projected on the user's view, as seen in figure 3.7. We fixed this by displaying the text in the 3D world close to the camera, instead of projecting it in the user's 2D view plane.

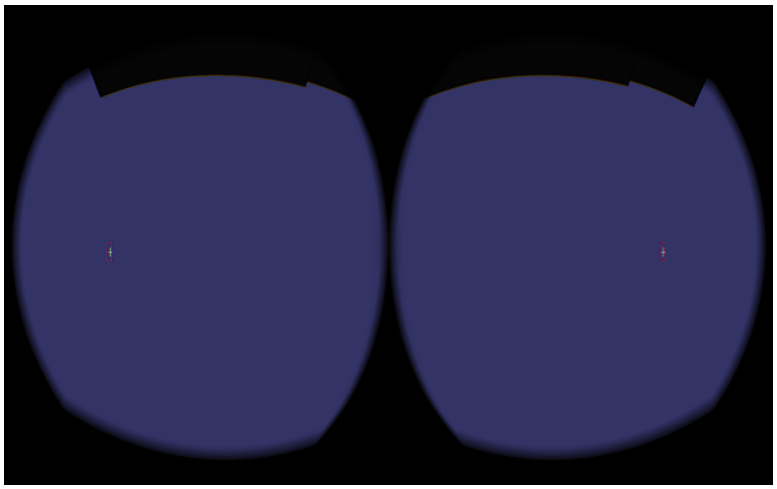


Figure 3.7: Text crosshair projected in a 2D plane

3.3 INTERACTION METHODS

In this section we describe the interaction methods that are present in this framework in terms of menu types, selection, manipulation, and navigation.

3.3.1 MENUS

LINEAR

In order to interact with our application, we have developed several types of menus. On a first approach, we reworked the linear menus that previously existed in pSIVE to work with the new version of OpenSceneGraph (shown in figure 3.8). This was done by listing all the options present in an object, from top to bottom centered on the screen.

The selected option is highlighted in white, while the other options have a transparent black background.

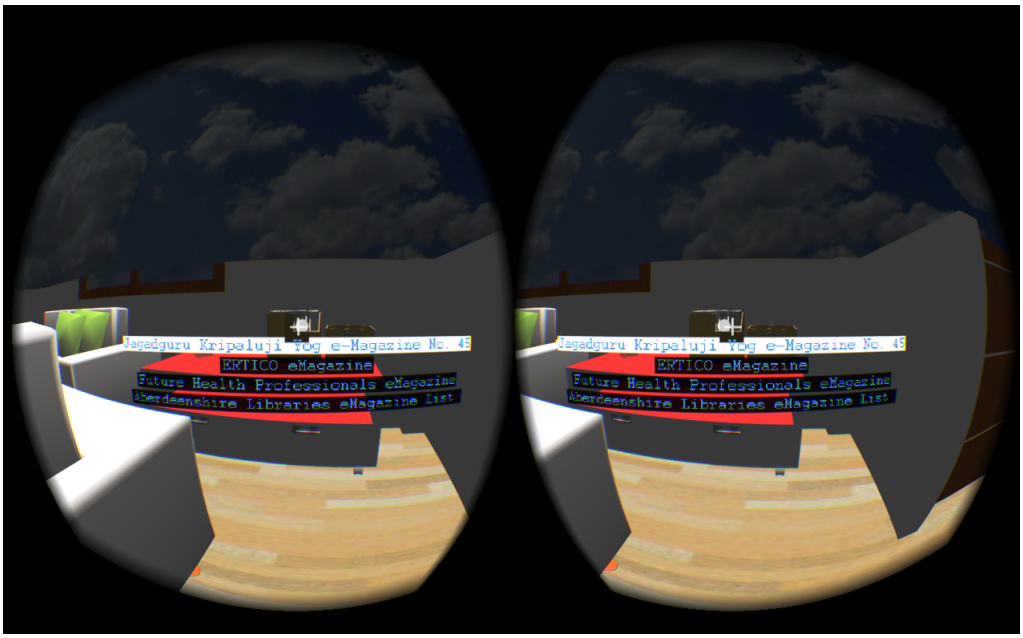


Figure 3.8: Linear menu

RADIAL MENU

On a second approach we implemented a pie menu (shown in figure 3.9) designed with gesture-based interaction in mind, for faster selection and improved accuracy (Callahan et al., 1988) (Chertoff, Byers, and LaViola, 2009). This menu is rendered using hundreds of triangles pointing at the center of the screen to create slices, with lines drawn in between the options. The option's text is then placed at the center of each slice.

Both of these menus (linear and radial) support gesture-based and button-based interaction. In the gesture-based interaction mode, menu options are selected based on the distance of the user's right hand in relation to the menu options; The closest option is always chosen upon the activation gesture (closing right hand), which means that the user doesn't need to hover his/her hand directly over the menu option s/he wishes to choose. This is more relevant in the pie menu, because the user can simply place his hand in the general direction of the slice s/he wishes to select, while in the linear menu the options are generally closer to each other and require more accuracy in

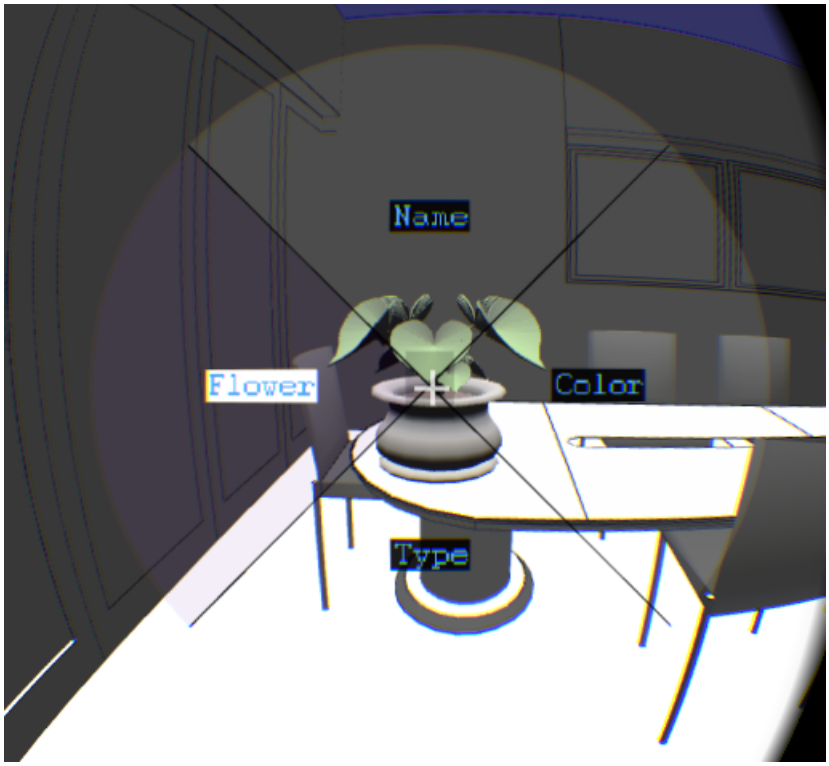


Figure 3.9: Radial menu

the placement of the hand. With a physical controller, buttons are used to navigate between the options and to perform the selection.

GAZE DIRECTED GRID MENU

After our first usability study (detailed in 4.1) comparing button-based selection versus gesture-based selection with the radial menu, and considering its results, we decided to reconsider our gesture-based approach for selecting menu options. Considering the results of an experiment ran with pSIVE in the past that compared laser-pointed selection with head-oriented selection (Souza, Dias, and Sousa Santos, 2014), we decided to implement a head-oriented menu system with the goal of adding models to the scene, for manipulation purposes.

To do so, we present the user with a 3x3 grid of pictures in the 3 dimentional space (configured in a gazemenus.xml file, exemplified in the listing 4 and depicted in figure3.10), with additional virtual buttons that allow the user to navigate more pages of options and close the menu.

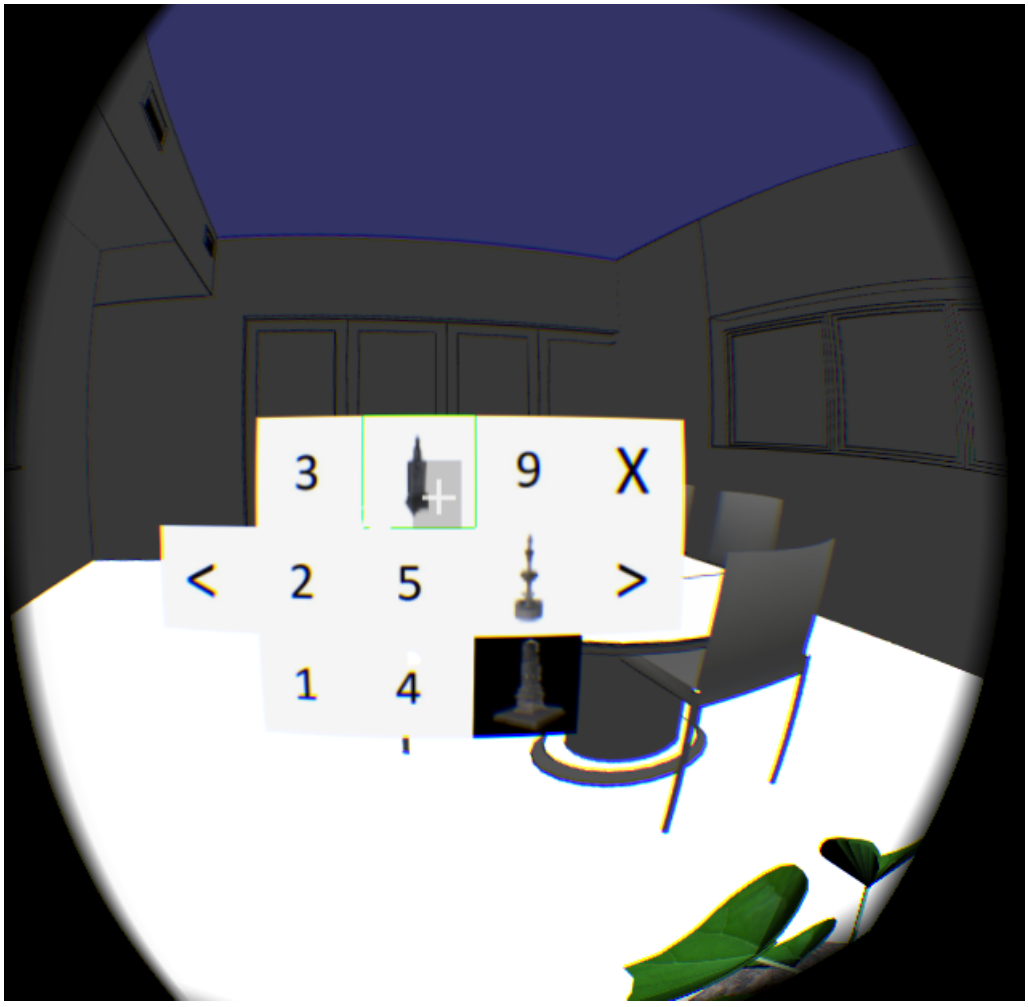


Figure 3.10: Gaze-directed menu

This grid is rendered after the scene, so it is never occluded by any object present in the scene. When the user looks at one of the options, it is highlighted with a green border, and when the user performs the input action (a hand grip in gesture mode, or a button press in controller mode), the option is selected.

```
<layout>
  <data>
    <model>
      <name>Model 1</name>
      <path>C:\Users\J\Documents\model1.obj</path>
      <thumbnail>C:\Users\J\Pictures\1.jpg</thumbnail>
    </model>
    <model>
      <name>Model 2</name>
      <path>C:\Users\J\Documents\model2.obj</path>
```

```
      <thumbnail>C:\Users\J\Pictures\2.jpg</thumbnail>
    </model>
    <model>
      <name>Model 3</name>
      <path>C:\Users\J\Documents\model3.obj</path>
      <thumbnail>C:\Users\J\Pictures\3.jpg</thumbnail>
    </model>
  </data>
</layout>
```

Listing 4: Menu configuration

3.3.2 SELECTION AND MANIPULATION METHODS

The application supports two separate categories of interaction. There is the option for interaction with a system of menus in order to browse content (System Control) such as PDF files, video files, image files, and text, and another option (Manipulation Control) that allows the manipulation of 3D objects within the virtual scene, regarding the rotation, scale and position of the object. Both interaction categories have support for two modes of interaction, namely gesture-based with information collected from the Kinect, or button-based with a physical controller such as a WiiMote.

INTERACTING WITH CONTEXTUAL CONTENT

When using the System Control style of interaction, the application supports two separate types of menus: The linear menu, and the pie menu. These menus contain contextual information of a certain object, as described in section 3.1.4, and upon activation will show the user some content

Content that involves images such as a PDF file (seen in figure 3.11), video (shown in figure 3.12), or a picture, is rendered on a textured rectangle placed in the center of the user's view.

Figure 3.11: PDF display

Figure 3.12: Video display

In the case of PDF content, the user can use gestures (right hand and left hand grips), or buttons on the controller to navigate between the pages.

MANIPULATING CONTENT

Regarding the manipulation task, we took into consideration the methods proposed by João Cardoso in his master thesis (Cardoso, 2015) for gesture-based manipulation. While his HandleBar method of manipulation, which uses the relative hand position to

rotate and scale the object (seen in figure 3.13), seemed ideal for our application, it did not consider an important dimension of movement that we needed; Positioning. As such, we decided not to fully implement his solution for the missing Degree of Freedom (DOF) in the object's rotation, which consisted in using the hand's up and down motion to rotate the object in that axis.

Instead, we used the hands' position in order to determine the position of the object in the scene, by calculating the midway point between the two hands in the virtual world.

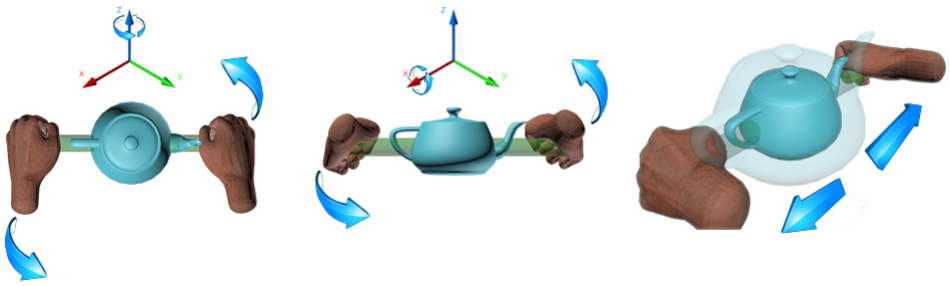


Figure 3.13: Handlebar manipulation (Cardoso, 2015)

As such, object manipulation using gestures in our application is done in three ways:

- Scale is changed according to the distance between the user's hands, scaling up when they go further away from each other and vice versa.
- Rotation is given by the angle between the x/z plane, for one axis, or the vertical plane given by the user's view direction, for the other axis, and a vector calculated from the position of the user's hands (vector and planes shown in figure 3.14).
- Position of the object is determined by the mid point between the user's hands in the world.

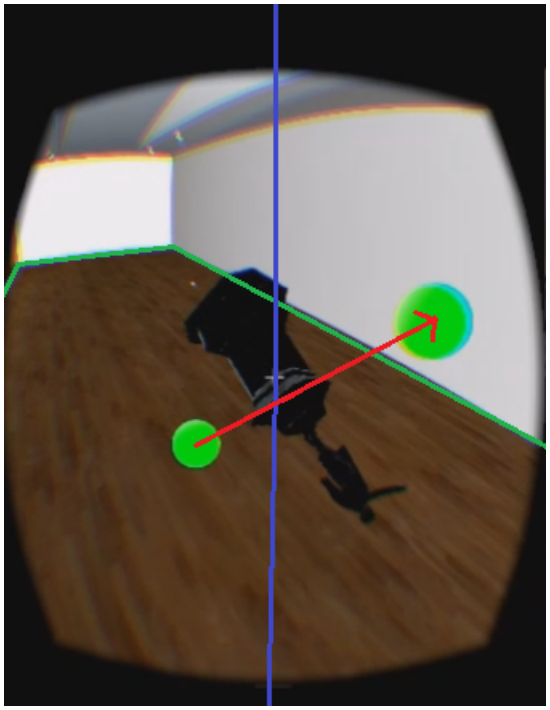


Figure 3.14: Hand vector, ground plane and view plane represented in red, green and blue respectively.

Scale and rotation changes are accumulated between interactions with the same object, meaning that the user can compound rotations to make up for the fact that there is a missing DOF, and scale the object up or down without being limited by the length of the user's arms.

Manipulation can also be done with a controller. In our testing environment, we used a WiiMote, but any controller with button input can be configured to work. This manipulation is split into 3 modes that can be toggled using the controller; Translation (x, y, and z axis), rotation (x, y, and z axis), and uniform scaling. The mode the user is currently in is depicted by an icon in the heads-up display (shown in figure 3.15).

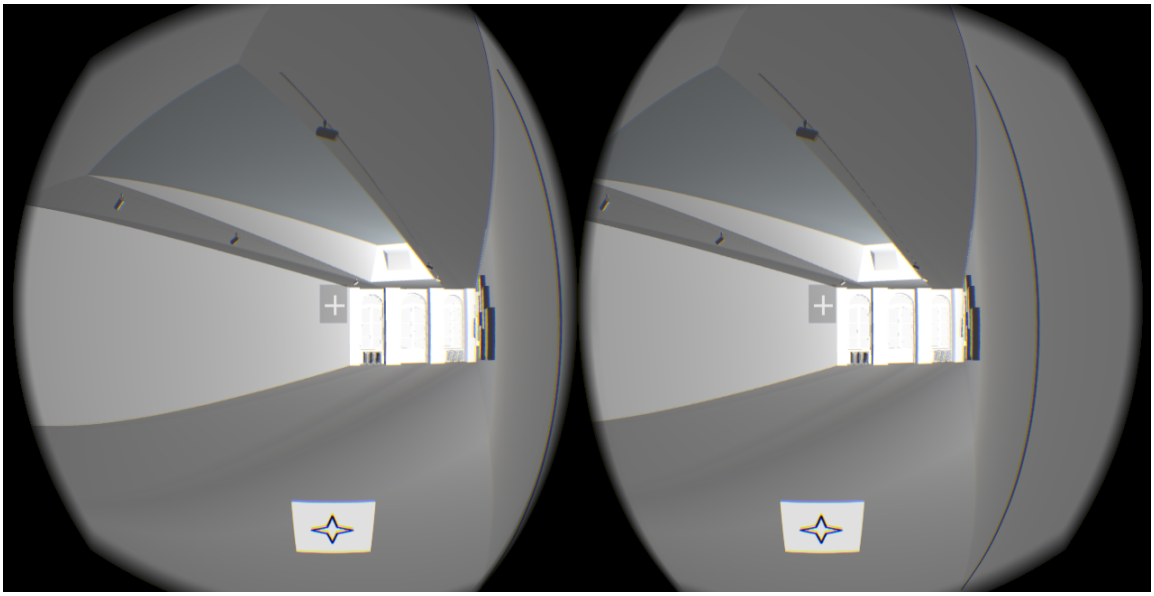


Figure 3.15: Manipulation mode menu

Manipulation is then done with button presses (in our case configured to the d-pad of the WiiMote, as shown in figure 3.16).

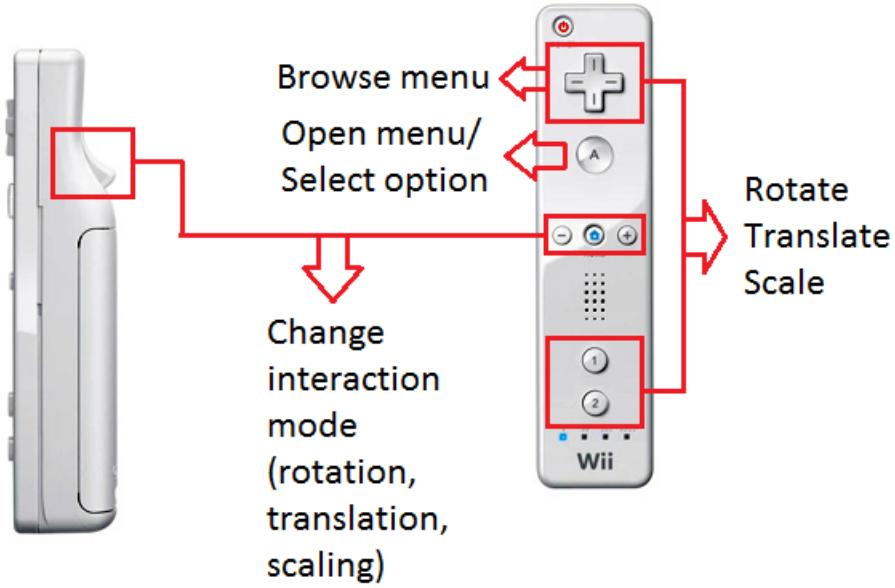


Figure 3.16: WiiMote interaction mapping

3.3.3 NAVIGATION METHODS

The client application supports three different methods of navigation in the world. The first one, and the main focus of our work, uses the skeleton information provided by the Kinect servers in order to position the user in the virtual scene, allowing him/her to navigate the virtual world by walking, with one to one position mapping between the real position and the virtual position. The second one uses a physical controller in order to navigate the virtual scene utilizing buttons. And lastly, we implemented the "Steering Wheel" method developed by João Cardoso (Cardoso, 2015), in which the user's hands are tracked by a Kinect server, and upon closing his hands a virtual steering wheel pops up, allowing the user to navigate the world as if s/he were driving a car.

When utilizing the physical controller, navigation is performed with four configurable inputs, mapped to four different directions (forward, backward, left and right). The forward and backward directions are determined by the user's view direction (gaze-directed), meaning s/he will always move towards or away from the direction s/he is looking at.

The "Steering Wheel" method of navigation we implemented works differently from how it was implemented by João Cardoso (Cardoso, 2015). In his implementation, there were two turning velocities, determined by a certain angle threshold. In our implementation, the turning velocity is proportional to the angle between the user's hands and the ground plane, allowing for tighter or looser turns, and giving the user more control over his movement and a more realistic steering wheel metaphor.

CHAPTER 4

Usability Studies

This chapter details two experiments performed with two separate groups of users, designed to evaluate the usability of the several navigation and interaction methods we developed.

The first experiment's goal was to evaluate the usability of the Kinect-based navigation and interaction versus the controller-based method, using our radial menu interface.

From the results of this preliminary experiment, we developed a hybrid interface that combines gaze-directed selection with gesture activation, or button selection and activation in a grid-like menu, in an attempt to speed up the selection times with gestures, that were lackluster in the previous implementation, and in order to test this new interface we ran another experiment comparing the Kinect-based interaction versus the controller-based interaction.

4.1 Preliminary Experiment with Gesture and Button Input Methods

In order to analyze and compare two of our implemented navigation and interaction techniques (System control with Kinect-based or controller-based interaction), we ran a

preliminary experiment with 12 volunteer participants with ages ranging from 15 to 17.

4.1.1 METHODOLOGY

This experiment aimed to verify if the two methods of interaction and navigation are equally usable in our demonstration environment. Our experiment had two input variables, namely the two different navigation and interaction methods. In one method, the users walk in the room to navigate, and use gestures to interact with the scene (shown in figure 4.1). In the other method, the user uses a controller with buttons to navigate and interact. Both methods display radial menus. The output variables of our experiment are in measures of time and number of mistakes made.

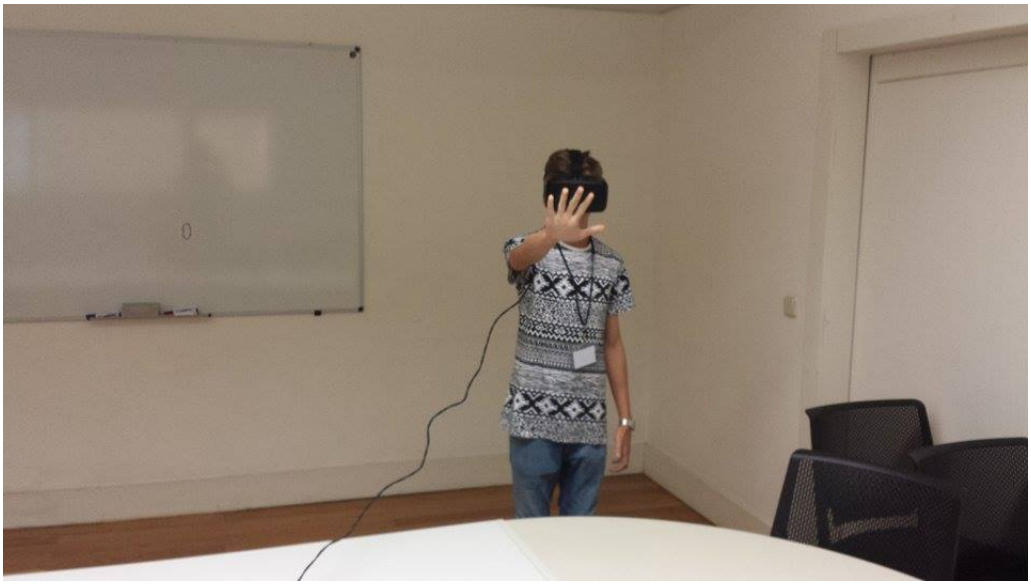


Figure 4.1: User performing gesture

The experiment consisted of a simple test of navigation and interaction: The participants had to navigate from one end of the virtual room to the other, and interact with an object (a plant pot) located on a table (shown in figure 4.2). Interaction is limited to the area delimited by the circle in the figure, in order to force the users to place themselves directly in front of the pot. Upon interaction, a menu is shown, and the goal is for the users to select the "Flower" option available on the menu.

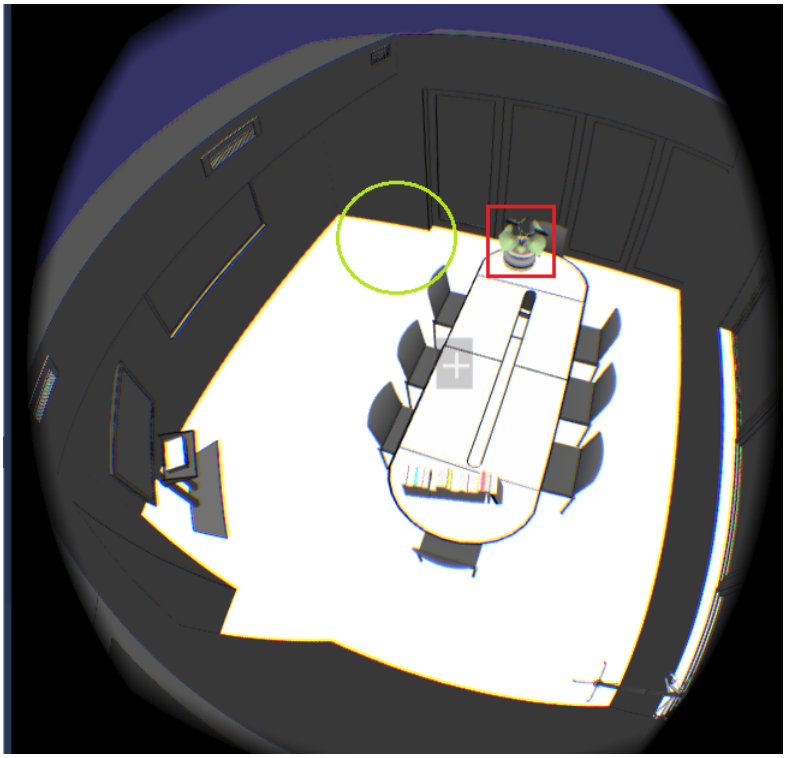


Figure 4.2: Experiment environment (interaction zone circled)

Users were given an ID at the start of the experiment (12 users, with IDs ranging from 0 to 11). Even numbered users ran the experiment using the controller first, and odd numbered users used the Kinect first, in order to attenuate possible bias due to learning effects. The participants were observed while performing the experiment and were asked to answer a questionnaire regarding their satisfaction, difficulties and preferred method. It is worth noting that while 12 users ran the experiment, only 11 delivered the questionnaire.

Measurements were logged during the experiment, and consisted of the time the participant took to get in front of the plant pot and activate the menu, the time the they took to select the correct option, as well as any incorrect options selected.

The gesture setup (depicted in figure 4.3) consisted of two kinects, two laptops and a wireless router. One kinect was connected to each of the laptops, and one server in each laptop was feeding information to the laptop running the client application. For the controller experiment, a WiiMote and a laptop were used.

Figure 4.3: Experimental setup

4.1.2 RESULTS

Regarding the times measured, the differences in the time taken to reach the plant pot are negligible with the controller-based method averaging 35,25 seconds to reach the plant pot versus the 35,41 seconds of the Kinect-based method while the interaction was faster when using the controller-based method, which averaged 16,6 seconds to activate the correct option versus the 30,3s that the users took with the Kinect.

Users also made more mistakes with the Kinect-based method, with three participants out of twelve making one mistake in selecting the right option, while nobody selected the wrong option using the controller.

The questionnaire results are detailed in tables **??**, **??** and **??**. Tables **??** and **??** indicate participant answers to several parameters, represented as a median of each index in a scale of 1 (disagree) to 5 (agree), in regards to navigation and interaction, respectively. Table **??** details the number of participants that claim to prefer one method or the other, or have no preference.

Table 4.1: Questionnaire results regarding navigation (1-5)

	Kinect	Controller
Ease of positioning	4	4
Intuitiveness	4	4
Has Annoying features	2	2
Requires training	5	5
Satisfaction	4	4

Table 4.2: Questionnaire results regarding interaction, from 1 to 5 (strongly disagree to strongly agree)

	Kinect	Controller
Ease of positioning	4	4
Intuitiveness	4	4
Has Annoying features	3	2
Requires training	5	3
Satisfaction	4	4

Table 4.3: Questionnaire results regarding preferences for navigation and interaction

	Navigation	Interaction
Kinect	7	6
Controller	1	3
Indifferent	3	2

4.2 EXPERIMENT WITH INTERACTION IN MENUS AND 3D MANIPULATION METHODS

Following our preliminary experiment, and derived from the conclusion that menu selection was significantly faster using a controller, we developed a new style of menu (detailed in section 3.3.1) designed to improve selection times in menus by using the user's gaze as the selection method, instead of utilizing the hands to navigate the menu. In order to obtain data to analyse and compare this new method of interaction, the grid menu was also adapted to be navigated using a controller, providing a basis of comparison between the two methods.

Taking advantage of this second round of usability tests, we also decided to perform a study comparing the manipulation interaction styles developed in the latter phase of

this work. As such, in addition to testing the menu interaction between gestures and a controller, we have also gathered data regarding the manipulation times and accuracy, performing a comparison between gestures and a controller.

4.2.1 METHODOLOGY

This experiment aimed to verify if the two methods of menu interaction and 3D manipulation are equally usable in our demonstration environment. Our experiment had two input variables, namely the two different menu interaction and manipulation methods. The experiment can be divided in three separate phases: The first phase regard the menu activation and option seletion. In this phase, the users interact with the menu (shown in figure 4.4) by looking at the option they would like to select, and perform a "close hand" gesture to activate that option. In the other method, the user uses a controller with buttons to navigate between the menu options and to activate the option they selected. Both methods display grid menus. The users are tasked with selecting three options (opening the menu each time), namely options 1, 3 and 7.

The second phase consists of a manipulation regarding the positioning of an object. In the gesture based method, users perform a grip gesture with the two hands and then move the hands to position the object, and in the controller based method they use buttons to translate the object. In both methods, the user's task is to position the object as such that it matches the ghost model of that object as closely as possible. The experiment begins the next phase once the user verbally signals that they are satisfied with their placement of the object.

The third phase concerns the manipulation of the object's rotation and scale. After an initial run with half a dozen users, we reached the conclusion that inexperienced users had difficulties grasping the gestures needed to properly rotate the object. As such, for this experiment we decided to give users two and a half minutes of training times before resetting the test and gathering data. Similarly to the second phase, in this one the users have an object centered inside the ghost model of that object, with a different orientation and scale (shown in figure 4.5), and they must use gestures or the controller's buttons (detailed in section 3.3.2) to rotate and scale the object to match

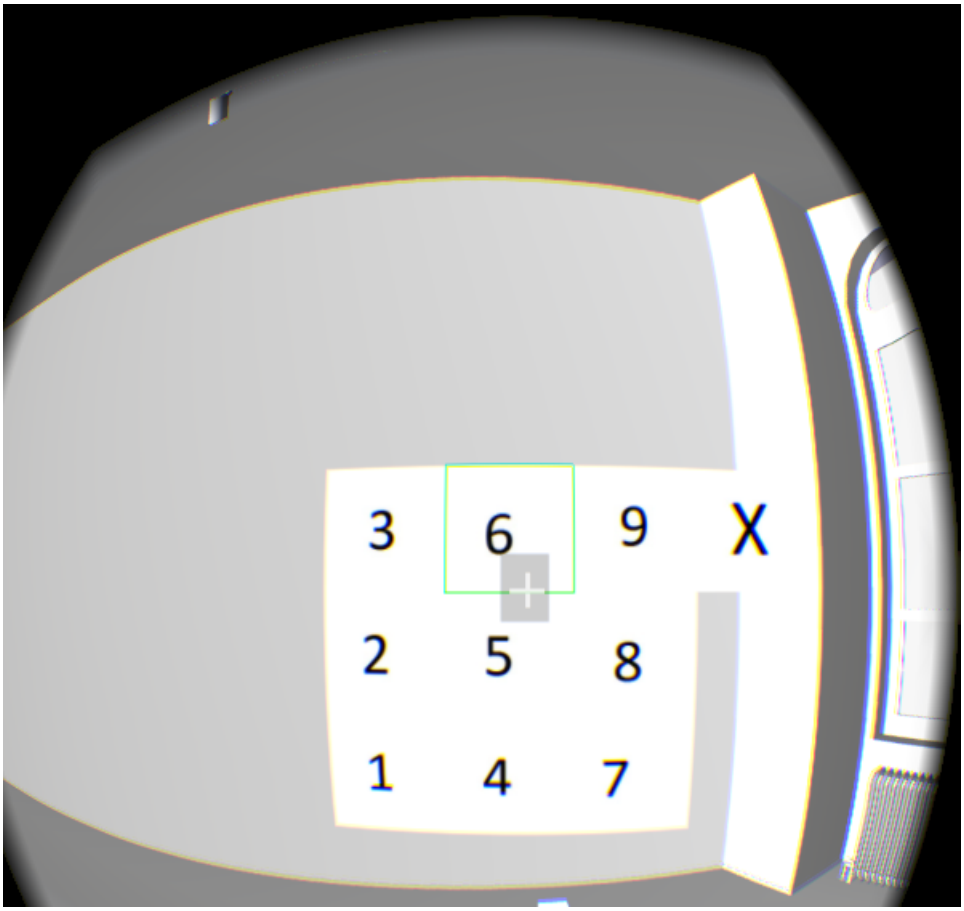


Figure 4.4: Grid menu (1-9 options)

the ghost model as closely as possible. The experiment ends when the user verbally signals that they are satisfied with their matching between the two models.

During the three phases, several measurements are automatically taken; during the first phase, the time between the menu opening and an option selection is taken, as well as which options were selected (to determine how many erroneous options were selected); during the second phase, the time between the start of the interaction and the end of the interaction, as well as the distance measured in the magnitude of the vector between the center of the ghost model and the object that is being moved is recorded; during the final phase, the time between the start of the interaction and the end of the interaction is taken, as well as the angular and scale difference between the two models.

Users were given an ID at the start of the experiment (28 users, with IDs ranging from 0 to 27). All users performed the tasks with both experimental conditions (a within group experimental design was used). Even numbered users ran the experiment using

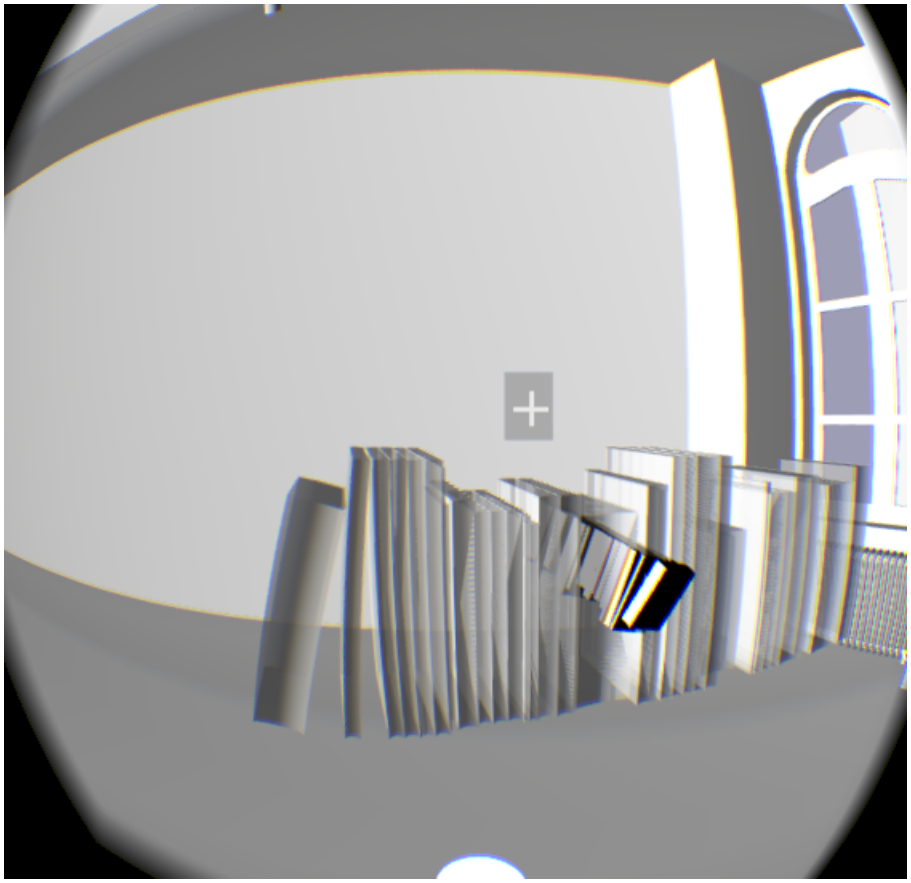


Figure 4.5: Ghost model and rotated model

the kinect first, and odd numbered users used the controller first, in order to attenuate possible bias due to learning effects. The participants were observed while performing the experiment and were asked to answer a questionnaire regarding their satisfaction, difficulties and preferred method. Some of the questions in this questionnaire were inspired by (Santos et al., 2015).

The test was conducted with two laptops (seen on figure 4.6), where one is used to remotely monitor what the user is seeing, and the other is used to run the application and the VRPN servers that interface with the Kinect and WiiMote.

Figure 4.6: Setup for the second test

4.2.2 RESULTS

In this subsection we will analyse the empirical results of the three phases of testing, as well as the results from the questionnaires.

MENU SELECTION

For the menu selection, the times from the opening of the menu and the selection of the option were logged, as well as which options were selected for error analysis. Participants were tasked with the selection of three different options.

	Kinect	Controller
Average selection time (s)	9.8± 6.4	7.4± 4.0
Average selection time (s)(excluding first option)	6.8± 2.9	5.8± 3.7
Average error (Number of errors)	0.78	1.45

Table 4.4: Menu selection data

As we can see in table 4.4, the average selection time changes drastically when we

consider the selection of the first option. In the case of the Kinect-based selection, discarding the first selected option results in an improvement of 31.2% of the average selection time. For the controller-based selection, that improvement is smaller, corresponding to a 22% reduction of the average selection time, which suggests that both options benefit from learning, with the Kinect-based method having a larger improvement when compared to the controller-based method (29.5% larger).

When looking at the situation where we exclude the selection of the first option, the Kinect-based method is around 1 second slower than with the controller-based method (6.8s vs 5.8s). This alone is a significant improvement from our previous results described in section 4.1, greatly reducing the difference between the two methods of interaction.

When we consider that we have introduced a 3 second activation delay in the Kinect-based selection method in order to prevent false positives in the gripping gesture detection, the Kinect-based method becomes as fast as the controller-based method if we reduce the 3 second activation delay to a 2 second activation delay. Further decreases to the activation delay should result in a faster selection time than the controller-based method, but most likely at the cost of more errors due to false positives.

Kinect	Controller	Indifferent
11	12	5

Table 4.5: Table of user preference for menu selection, in number of users

As we can see in table 4.5, the participants were split among the two methods concerning preferences, with the Kinect-based method being preferred by 39.3% of the users and the controller-based method being preferred by 42.9% of the users. Table 4.6 details the responses to the questionnaire, represented as the median value among users, from 1 to 5 (strongly disagree to strongly agree).

A detailed overview of the number of errors made by users can be seen in charts 4.7 and 4.8, for the controller-based method and Kinect-based method, respectively. For the purpose of this task, we consider an error to be the selection of an option that wasn't part of the task, or selecting the same option multiple times. Contrary to our beliefs, the Kinect-based method appears to be less prone to errors on average than

	Kinect	Controller
Ease of Selection	4	5
Intuitiveness	4	5
Is tiresome	3	1
Has Annoying features	2	1
Requires training	5	5

Table 4.6: Questionnaire results regarding selection, from 1 to 5 (strongly disagree to strongly agree)

the controller-based method (seen on table 4.4). This could be due to the difference in the speed of the gaze-directed selection of the option versus the controller-based selection, as all the options (except the center option) are at the same distance from the user's gaze when he opens the menu, while using a controller some options are in the far opposite corner of the grid.

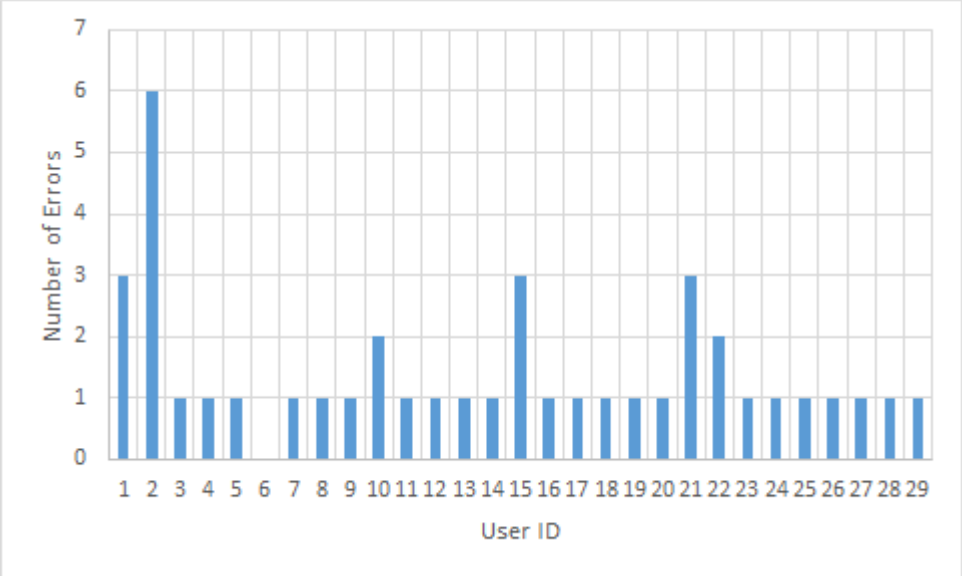


Figure 4.7: Number of errors by user for the controller-based method

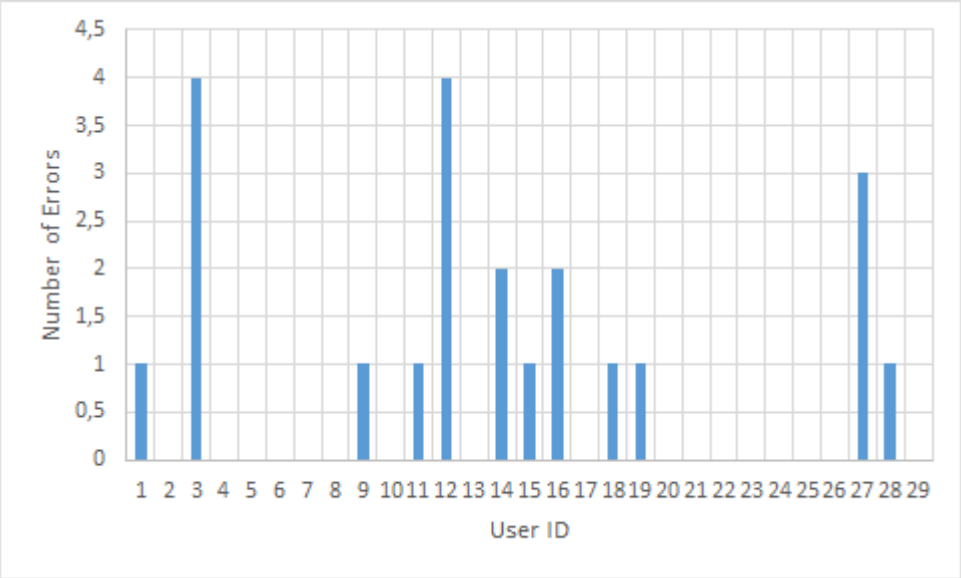


Figure 4.8: Number of errors by user for the Kinect-based method

OBJECT POSITIONING

For the object positioning test, the times and the positions of the object in relation to the goal position were taken. To simplify analysis, we will consider three key moments: The start of the interaction, the timing to the first instance of a distance of <0.002 (measured as the magnitude of the distance between the center of the object and the goal position in world-coordinates), and the timing of the best position.

Charts 4.9 and 4.10 show the time each participant took to position the object, both for the first instance where it was within our criteria (<0.002), and for their best position timing.

	Kinect	Controller
First position timing (s)	23.4± 13.0	48.1± 18.3
Best position timing (s)	38.7± 29.0	54.8± 20.3
Best position	0.00015	0.00013

Table 4.7: Table of results, expressed in average values among all users, for the positioning test

As we can see in table 4.7, the user takes roughly half of the time to achieve the minimum required distance than the controller-based method while using the Kinect-based method. On the other hand, after achieving the minimum distance, the adjustments required until the best position was achieved took only 13.92% of the

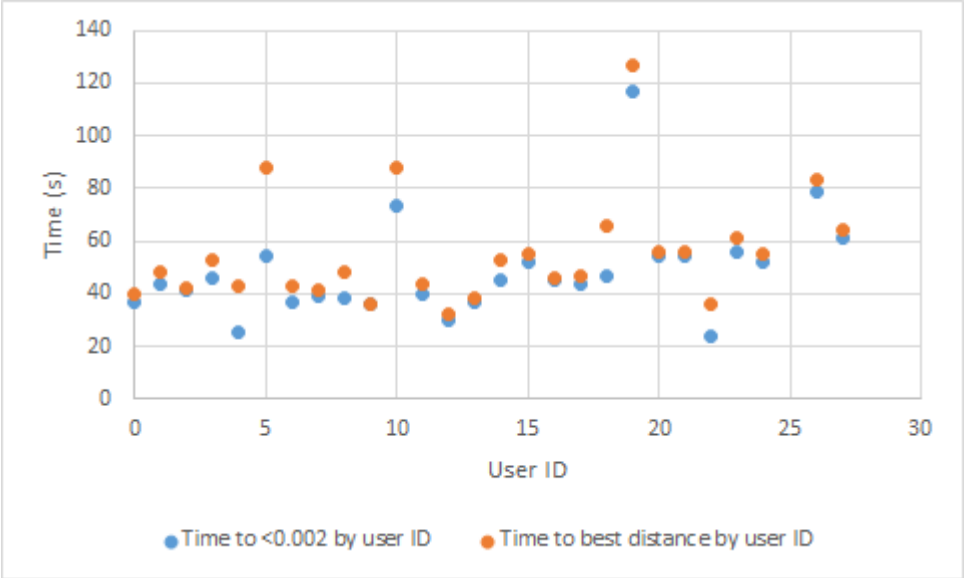


Figure 4.9: X axis is the user ID, Y axis is time in seconds using the controller

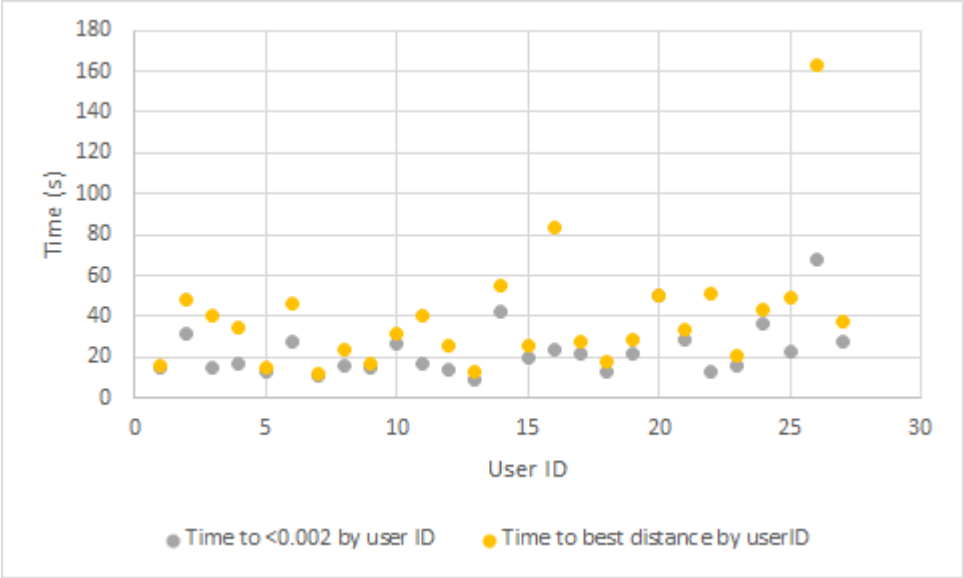


Figure 4.10: X axis is the user ID, Y axis is time in seconds using gestures

time of achieving the minimum distance for the controller-based method, while the Kinect-based method was slower in the fine-tuning, taking 65.68%, which means it took almost 5 times longer (471%) than the controller-based method to fine tune the object to the best position.

The difference in the average best position for both methods is so small that is is completely negligible, which suggests that both methods are equaly accurate.

2 out of the 28 participants failed this test by not being able to achieve the minimum

distance required (<0.002). One failed using the controller-based method, and another failed using the Kinect-based method.

Kinect	Controller	Indifferent
13	14	1

Table 4.8: Table of user preference for object positioning, in number of users

Similarly to the last phase of testing, users were split among the two options in terms of preference, with 46.4% preferring the Kinect-based method and 50% preferring the controller-based method, as shown in table 4.8.

OBJECT ROTATION

For this phase of the experiment, the timings and angles of the object during interaction were logged. To simplify analysis, three key moments were considered: The start of the interaction, the first moment that the object hits our minimum requirement (<5° in the x and z axis), and the moment where the object's x and z axis angle error is minimal (this angle error is the sum of the difference between the target angles and the absolute value of the current angles).

Figures 4.11 and 4.12 show the timing for the first instance where the object's rotation fits our criteria, as well as the instance where it most closely matches the target angles.

	Kinect	Controller
First angle time (s)	75.5± 53.6	53.5± 37.0
Best angle time (s)	88.3± 57.3	75.8± 41.4
Best angle error (°)	1.19	1.03

Table 4.9: Table of results, expressed in average values among all users, for the rotation test

As we can see in table 4.9, users achieve similar accuracy with both methods, with the Kinect-based method being 30% slower than the controller-based method. However, despite being slower to arrive at <5° in the x and z axis, the Kinect-based method is faster after that point in reaching the minimum angle error (17s vs 22s). This

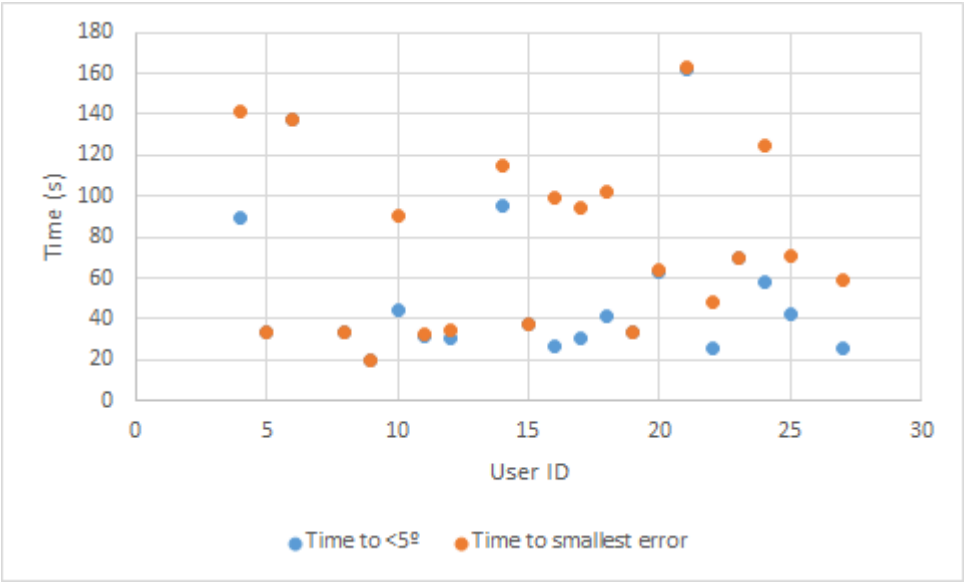


Figure 4.11: X axis is the user ID, Y axis is time in seconds, using the controller

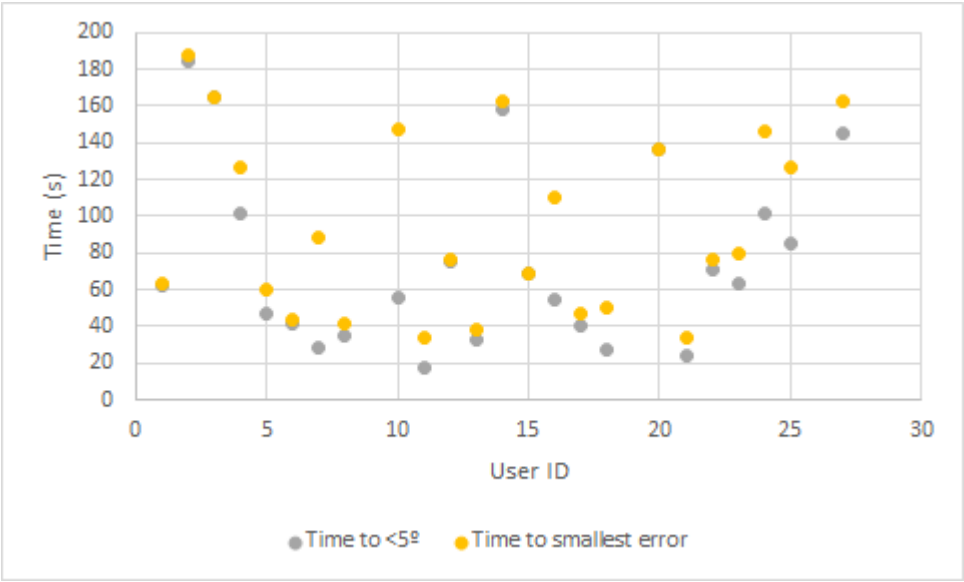


Figure 4.12: X axis is the user ID, Y axis is time in seconds, using gestures

difference can by explained by the necessity to accumulate rotations in the Kinect-based method (only 2 degrees of freedom in rotation), where the controller-based method has 3 degrees of freedom in rotation.

Kinect	Controller	Indifferent
12	13	3

Table 4.10: Table of user preference for object rotation, in number of users

As with the previous 2 tests, users preferences are split between the two methods, as we can see in table 4.10, where 42.9% of users prefer the Kinect-based method and 46.4% of users prefer the controller-based method.

There were 11 failed attempts at this test, either with the controller-based method or the Kinect-based method, by not being able to achieve the minimum angle required in both axis (<5º in x and z), 7 of which were using the controller-based method and 4 of them using the Kinect-based method.

MANIPULATION QUESTIONNAIRE RESULTS

In table 4.11 we present a general overview of the results for the questionnaire answers regarding manipulation (positioning & rotation) for this study. All results are expressed as the median value between all users of the 1-5 (strongly disagree - strongly agree) scale answer given. From these results we can see no clearly defined preferences between the two methods, apart from a reasonable expectation regarding how tiresome the interaction methods are (we expected the gesture-based method to be more tiresome, for obvious reasons).

	Kinect	Controller
Ease of Rotation	3	3
Ease of Positioning	4	4
Is tiresome	3	2
Intuitiveness	4	4
Has Annoying features	3	3
Requires training	5	4

Table 4.11: Questionnaire results regarding manipulation, from 1 to 5 (strongly disagree to strongly agree)

4.3 CONCLUSION

Regarding the first usability study, while tables **??** and **??** seem to indicate little difference between the two options, the Kinect-based method seems to be predominantly preferred in both navigation and interaction, as shown in table **??**.

While we deem the difference in navigation times to be negligible, the Kinect-based method of interaction was significally slower. With all this in mind, our results suggest that even though the Kinect-based method for navigation and interaction is slower on both accounts, participants seem to prefer it, possibly due to the novelty factor.

From our second usability study, where we took what we learned previously to improve our menu interface, we achieved much more comparable results, with the Kinect-based method still trailing behind the controller-based method for interaction in menus, but only by a very small margin, and mostly due to the introduction of a delay to prevent false-positive grip detection interfering with the interaction. We also took this opportunity to compare interaction methods between controller-based methods and Kinect-based methods. From our results, we concluded that both methods fare about the same when it comes to accuracy in the positioning and rotation of the object, with the Kinect-based method being the faster one in positioning, and the controller-based method being the faster when it comes to rotation.

CHAPTER 5

Imaginary Museum

5.1 INTRODUCTION

The Imaginary Museum (Eliseu et al., 2015) was an artistic installation set up during four days (October 2015), in the Museum of the City of Aveiro in the scope of the ART#14 conference .

This installation uses our framework to configure a virtual world that allows visitors of the museum to navigate one of the museum's rooms in virtual reality and set up their own virtual exhibition by placing and manipulating virtual art pieces in the scene.

5.2 CONCEPT

The initial concept of the Imaginary Museum was based on an empty room where the visitors would be able to use 3D models to populate the empty room with their own exibition. The room used for the installation and provided by the museum was not devoid of content, as seen on figure 5.1. It had several portraits relating to notable historical figures of the city.

Given the context of the room, we adapted our installation to better fit the overall theme present in the room. As such, we opted to provide models that depicted public monuments of the people depicted on the portraits, for visitors to freely manipulate and set up in the room. However, three of those people did not have any public monuments.

Figure 5.1: Museum room

For those three, we decided to use the small, blank squares that surround the portraits in order to depict some of their artistic work. For instance, when the visitor looks at Licinio Pinto, a renowned painter of ceramic tiles, the small squares transform to depict those tiles (shown in figure 5.2).

Figure 5.2: Small squares depicting ceramic tiles in the virtual room

5.3 SETUP

To achieve full coverage of the room, we used 3 Kinect sensors connected to 3 laptops running the custom VRPN server (pictured in figure 5.3). These sensors provided an ample area where the visitors were detected, allowing them to freely navigate the entire room (excluding the space distancing 1 meter in front of the Kinect and behind, due to hardware limitations). Interaction with the 3D models was then confined to the side of the room that had the Kinect sensors, because the visitor has to be facing the sensor in order for it to properly detect hand gestures.

Figure 5.3: Museum Kinect setup

The visitors were equiped with a backpack containing a laptop running our client application, receiving information from the three VRPN servers and connected to an Oculus Rift HMD (pictured in figure 5.4).

Figure 5.4: Museum backpack setup

5.4 INTERACTION

For interaction in this installation we used the gaze-directed grid menu detailed in section 3.3.1. This grid menu contained thumbnails of the 3D models (shown in figure 5.5) of the monuments that could be placed in the scene for manipulation.

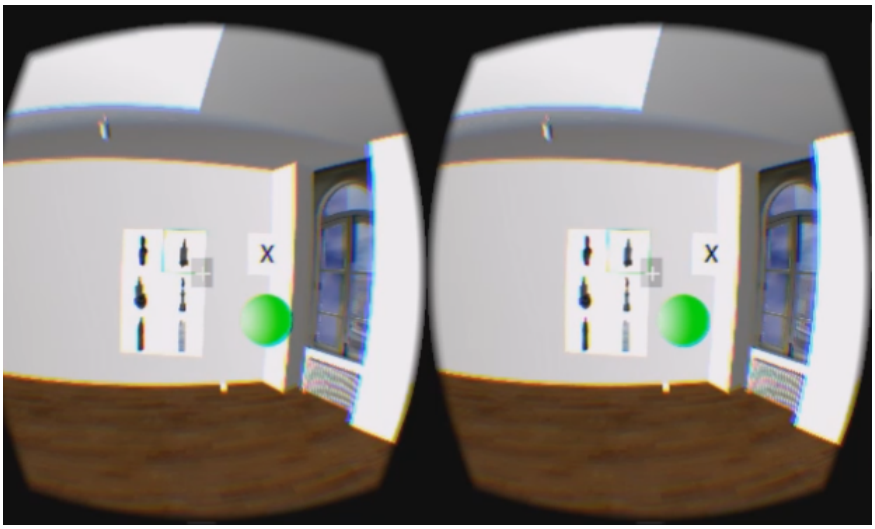


Figure 5.5: Grid menu containing monuments

In order to prevent false-positives in the grip detection when the visitor leaves a sensor's zone of interaction and enters another's, we introduced an artificial 3 second delay between the hand grip gesture and the opening of the menu or selection of an option.

For the object manipulation, we used the gesture system described in section 3.3.2, in

which the user used his/her hands to scale, position and rotate the object (manipulation in action shown in figure 5.6).

Figure 5.6: User performing object manipulation

5.5 CONCLUSION

A result of the work between artists and engineers, this installation provided an excellent environment to field test our application with people of various backgrounds, with and without virtual reality experience, while also giving us a real use case scenario that had thematical coherence between the real and the virtual worlds.

Overall, the reception to this virtual reality experience was overwhelmingly positive. Not all visitors used the full array of interactions available, as the gesture based and menu interaction required some initial learning, however the users that did learn the basics soon became proficient in menu interaction and object manipulation.

CHAPTER 6

Conclusion

The main goal of this work was to provide a framework that allows the creation of an interactive virtual world in virtual reality, complete with navigation and interaction. To that end, we developed features upon a previously developed framework (pSIVE), that allow the user to either interact with a configurable environment, in the form of accessing content such as videos and PDF files within models in the scene, or to directly insert and manipulate objects in the environment.

In order to provide an immersive experience, we adapted the previously developed framework to work with one or several Kinect cameras, tracking the user in a physical room and mapping his movements to the virtual scene. To do this we developed tools to calibrate the Kinect camera's point cloud of the real room to the virtual model of the room. In the future, it would be interesting to provide a fully automatic tool that would perform the calibration. We also added features for gesture-based interaction for browsing and manipulating content.

To adapt the framework to gesture-based interaction, we proposed, studied, and implemented several types of menus (linear, radial, grid-like), that can use this new gesture-based method of interaction as well as the controller-based interaction method present in the previous version of the framework.

We also ran two user studies, the first one comparing input methodology (controller vs gestures in a radial menu), and movement methodology (tracked by the kinect versus buttons on a controller). From this study we concluded that users were interested in

gesture-based controls and one-to-one mapped navigation, and while the navigation timings with the kinect tracking method were comparable to the timings when using the controller, menu selection wasn't as good. From the results of the first study we decided to develop a new menu interface (grid-like) designed for a gaze-directed selection of options combined with gestural input for selection. In order to validate this new menu as well as the manipulation features, we ran a second user study comparing gaze-directed gesture-based menu selection and gesture-based manipulation with controller-based selection and manipulation in the same kind of menu. The results of this second experiment show a clear improvement in selection times with the new menu style in both methods of interaction, as well as comparable selection times for both methods. Regarding the manipulation test results, the gesture-based method was significantly faster than the controller-based method while achieving similar accuracy when it comes to positioning. On the other side, when dealing with rotations, the controller-based method was faster by a margin of about 30% when compared to the gesture-based method.

From the results of these user tests, as well as the experience from developing and testing the framework, we consider that both controller-based methods and gesture-based methods have their place in interaction inside immersive virtual environments. Controller-based methods have the advantage of providing clearer and more accurate actuation when compared with gesture-based methods, where we found some cases of false-positives and false-negatives and had to introduce tolerance on the activation of the menu or the selection of an option. On the other hand, gesture-based interaction methods provide us with a more natural type of interaction with the virtual environment, particularly in the case of object manipulation. We believe that a hybrid method of interaction could prove to be benefitial, where a physical controller is used to activate menus and select options, while combined with depth sensor to provide 6 degrees of freedom on manipulation tasks.

To wrap up this work, we put it to the test in a live scenario, in the museum of the city of Aveiro, within the context of an art installation created using our framework, where users could navigate a room in the museum and experience virtual content, as well as setting up their own virtual exposition by adding models of monuments through

a grid-like menu and manipulating them by positioning, rotating and scaling them in the virtual room.

In the future, an implementation of a hybrid method of interaction using gestures and a controller, and the usage of a 6DOF Hands technique for manipulation could wield better results for a more precise and accurate experience, that would perhaps require less time to learn than the purely gesture-based methods of interaction. Augmented reality is also another expansion point, where mobile applications or augmented reality glasses could be used, removing the necessity to create a virtual room modelled after the real room. Spacial audio support with an acoustic model of the virtual environment, as well as physics and collision detection could be another step in immersing the user deeper in the experience.

The following publications resulted from the work presented in this thesis:

Pinto J., Dias P., Eliseu S., Sousa Santos B. “Interactive configurable virtual environment with Kinect navigation and interaction”. Atas do Encontro português de Computação Gráfica e Interação – SciTecIn/EPCGI 2015. Departamento de Engenharia Electrotécnica Polo II da Universidade de Coimbra, Portugal, 12-13 Novembro 2015.

Eliseu S., Bernardino P., Pinto J., Dias P., Sousa Santos B. "Museu Imaginario". 14º Encontro Internacional de Arte e Tecnologia: 14.ART: ARTE E DESENVOLVIMENTO HUMANO, UA Editora, University of Aveiro, 2015, pp. 340-344.

References

Bastien (2013). *A virtual environment interfaced with an e-learning platform*. URL: http://www.openspace3d.com/lang/en/2013/03/19/dcns-training-schooldcns-training-school.

Besl, Paul J. and Neil D. McKay (1992). «A Method for Registration of 3-D Shapes». In: *IEEE Trans. Pattern Anal. Mach. Intell.* 14.2, pp. 239–256. ISSN: 0162-8828. DOI: 10.1109/34.121791. URL: http://dx.doi.org/10.1109/34.121791.

Bierbaum, Allen et al. (2001). «VR Juggler: A Virtual Platform for Virtual Reality Application Development». In: *Proceedings of the Virtual Reality 2001 Conference (VR'01)*. VR '01. Washington, DC, USA: IEEE Computer Society, pp. 89–. ISBN: 0-7695-0948-7. URL: http://dl.acm.org/citation.cfm?id=580521.835847.

Bobick, Aaron F. et al. (1999). *The KidsRoom: A Perceptually-Based Interactive and Immersive Story Environment*.

Bowman, Doug A. et al. (2004). *3D User Interfaces: Theory and Practice*. Redwood City, CA, USA: Addison Wesley Longman Publishing Co., Inc. ISBN: 0201758679.

Burdea, G. and P. Coiffet (2003). «Virtual reality technology». In: *Presence: Teleoperators & Virtual Environments*. Vol. 12. 6, pp. 663–664.

Callahan, J. et al. (1988). «An empirical comparison of pie vs. linear menus». In: *Proceedings of the SIGCHI conference on Human factors in computing systems - CHI '88*. New York, New York, USA: ACM Press, pp. 95–100. ISBN: 0201142376. DOI: 10.1145/57167.57182. URL: http://dl.acm.org/citation.cfm?id=57167.57182.

Cardoso, João A. C. (2015). *3D manipulation and navigation methods with gestures for large displays*. Campus Universitário Santiago, Aveiro, Portugal.

Chertoff, Dustin B., Ross Byers, and Joseph J. LaViola (2009). «Poster: Evaluation of menu techniques using a 3D game input device». In: *2009 IEEE Symposium on 3D User Interfaces*. IEEE, pp. 139–140. ISBN: 978-1-4244-3965-2. DOI: 10.1109/3DUI.2009.4811225. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4811225.

Craig, Alan B., William R. Sherman, and Jeffrey D. Will (2009). *Developing Virtual Reality Applications: Foundations of Effective Design*. Morgan Kaufmann, p. 448. ISBN: 0080959083. URL: https://books.google.com/books?hl=en\&lr=\&id=2P91gPYr5KkC\&pgis=1.

Deering, Michael F. (1995). «HoloSketch: a virtual reality sketching/animation tool». In: *ACM Transactions on Computer-Human Interaction* 2.3, pp. 220–238. ISSN: 10730516. DOI: 10.1145/210079.210087. URL: http://dl.acm.org/citation.cfm?id=210079.210087.

Eliseu, Sergio et al. (2015). «Museu Imaginario». In: *14º Encontro Internacional de Arte e Tecnologia: 14.ART: ARTE E DESENVOLVIMENTO HUMANO*, pp. 340–344.

Feiner, Steven et al. (1993). «Windows on the world». In: *Proceedings of the 6th annual ACM symposium on User interface software and technology - UIST '93*. New York, New York, USA: ACM Press, pp. 145–155. ISBN: 089791628X. DOI: 10.1145/168642.168657. URL: http://dl.acm.org/citation.cfm?id=168642.168657.

Foxlin, E., Michael Harrington, and George Pfeifer (1998). «Constellation». In: *Proceedings of the 25th annual conference on Computer graphics and interactive techniques - SIGGRAPH '98*. New York, New York, USA: ACM Press, pp. 371–378. ISBN: 0897919998. DOI: 10.1145/280814.280937. URL: http://dl.acm.org/citation.cfm?id=280814.280937.

Gebhardt, Sascha et al. (2013). «Extended pie menus for immersive virtual environments.» In: *IEEE transactions on visualization and computer graphics* 19.4, pp. 644–51. ISSN: 1941-0506. DOI: 10.1109/TVCG.2013.31. URL: http://www.ncbi.nlm.nih.gov/pubmed/23428449.

Gerber, D. and D. Bechmann (2005). «The spin menu: a menu system for virtual environments». In: *IEEE Proceedings. VR 2005. Virtual Reality, 2005.* IEEE, pp. 271–272. ISBN: 0-7803-8929-8. DOI: 10.1109/VR.2005.1492790. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1492790.

Hernandez, L. et al. (2003). «The empty museum. Multi-user interaction in an immersive and physically walkable VR space». In: *Proceedings. 2003 International Conference on Cyberworlds.* IEEE Comput. Soc, pp. 446–452. ISBN: 0-7695-1922-9. DOI: 10.1109/CYBER.2003.1253488. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1253488.

Ii, Russell M Taylor et al. (2001). «VRPN : A Device-Independent , Network-Transparent VR Peripheral System». In: *VRST*, pp. 55–61. DOI: 10.1145/505008.505019.

Lowgren, Jonas et al. (2014). *The Encyclopedia of Human-Computer Interaction, 2nd Ed.* The Interaction Design Foundation. ISBN: 978-87-92964-00-7.

Messinis, Ioannis et al. (2010). «Investigation of the Relation Between Interaction and Sense of Presence in Educational Virtual Environments». In: *2010 International Conference on e-Education, e-Business, e-Management and e-Learning.* IEEE, pp. 428–431. ISBN: 978-1-4244-5680-2. DOI: 10.1109/IC4E.2010.137. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5432499.

Mine, M. (1995). *Virtual environment interaction techniques.* Tech. rep. Department of Computer Science, University of North Carolina, pp. 1–18. DOI: 10.1.1.38.1750. URL: http://dl.acm.org/citation.cfm?id=897820$\backslash$nhttp://staffwww.itn.liu.se/~karlu/courses/TNM086/papers/VEinteractionTechniques.pdf.

Richard H. Jacoby, Stephen R. Ellis (1992). «Using virtual menus in a virtual environment». In: URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.109.16.

Rodrigues, Vasco et al. (2015). «Mid-Air Manipulation of 3D Models in (Semi-)Immersive Virtual Environments». In: *Atas do Encontro português de Computação Gráfica e Interação – SciTecIn/EPCGI 2015.* URL: http://scitecin.isr.uc.pt/Proceedings/Papers/EPCGI/26.pdf.

Ryan, Tao Ni, P. McMahan, and Doug A. Bowman (2008). «Tech-note: rapMenu: Remote Menu Selection Using Freehand Gestural Input». In: *2008 IEEE Symposium on 3D User Interfaces.* IEEE, pp. 55–58. ISBN: 978-1-4244-2047-6. DOI: 10.1109/3DUI.2008.4476592. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4476592.

Santos, Marc Ericson C et al. (2015). «Toward Standard Usability Questionnaires for Handheld Augmented Reality.» In: *IEEE computer graphics and applications* 35.5, pp. 66–75. ISSN: 1558-1756. DOI: 10.1109/MCG.2015.94. URL: http://www.ncbi.nlm.nih.gov/pubmed/26416363.

Schroeder, Will, Kenneth M. Martin, and William E. Lorensen (1998). *The Visualization Toolkit (2Nd Ed.): An Object-oriented Approach to 3D Graphics.* Upper Saddle River, NJ, USA: Prentice-Hall, Inc. ISBN: 0-13-954694-4.

Schuemie, Martijn J. and Charles A.P.G. van der Mast (1999). *Presence: Interacting in VR?*

Slater, Mel and Martin Usoh (1994). «Body Centred Interaction in Immersive Virtual Environments». In: *Artificial Life and Virtual Reality*. John Wiley and Sons, pp. 125–148.

Song, Peng et al. (2012). «A Handle Bar Metaphor for Virtual Object Manipulation with Mid-air Interaction». In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: ACM, pp. 1297–1306. ISBN: 978-1-4503-1015-4. DOI: 10.1145/2207676.2208585. URL: http://doi.acm.org/10.1145/2207676.2208585.

Souza, Danilo (2013). *Platform for Setting up Interactive Virtual Environments*. Campus Universitário Santiago, Aveiro, Portugal.

Souza, Danilo, Paulo Dias, and Beatriz Sousa Santos (2014). «Choosing a selection technique for a virtual environment». In: *Lecture Notes in Computer Science*. Vol. 8525 LNCS. PART 1. Springer Verlag, pp. 215–225.

STUDEC. URL: http://www.studec.fr/uk/skills/it_systems.php (visited on 09/08/2015).

Suma, Evan A. et al. (2011). «FAAST: The flexible action and articulated skeleton toolkit». In: *Proceedings - IEEE Virtual Reality*, pp. 247–248. ISBN: 9781457700361. DOI: 10.1109/VR.2011.5759491.

Sutherland, Ivan E. (1968). «A head-mounted three dimensional display». In: *Proceedings of the December 9-11, 1968, fall joint computer conference, part I on - AFIPS '68 (Fall, part I)*, p. 757. ISBN: 158113052X. DOI: 10.1145/1476589.1476686. URL: http://dl.acm.org/citation.cfm?id=1476589.1476686.

Tarr, Michael J and William H Warren (2002). «Virtual reality in behavioral neuroscience and beyond.» In: *Nature neuroscience* 5 Suppl, pp. 1089–92. ISSN: 1097-6256. DOI: 10.1038/nn948. URL: http://www.ncbi.nlm.nih.gov/pubmed/12403993.

Tiefenbacher, Philipp, Andreas Pflaum, and Gerhard Rigoll (2014). «[Poster] Touch gestures for improved 3D object manipulation in mobile augmented reality». In: *2014 IEEE International Symposium on Mixed and Augmented Reality (ISMAR)*. IEEE, pp. 315–316. ISBN: 978-1-4799-6184-9. DOI: 10.1109/ISMAR.2014.6948467. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6948467.

Wang, Robert, Sylvain Paris, and Jovan Popović (2011). «6D Hands: Markerless Hand-tracking for Computer Aided Design». In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. Santa Barbara, California, USA: ACM, pp. 549–558. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047269. URL: http://doi.acm.org/10.1145/2047196.2047269.

Wang, Rui and Xuelei Qian (2010). *OpenSceneGraph 3.0: Beginner's Guide*. Packt Publishing. ISBN: 1849512825, 9781849512824.

Zeltzer, David (1992). «Autonomy, Interaction, and Presence». In: *Presence: Teleoper. Virtual Environ.* 1.1, pp. 127–132. ISSN: 1054-7460. URL: http://dl.acm.org/citation.cfm?id=128947.128957.

Zsolt Szalavari, Michael Gervautz (1997). «The Personal Interaction Panel - a Two-Handed Interface for Augmented Reality». In: URL: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.898.

Appendices

first experiment questionnaires

ID:____

Questionário ao utilizador para o Teste de navegação e interação com um menu em realidade virtual

Instruções: Agradecemos a sua colaboração na realização deste estudo, que tem por objectivo avaliar dois métodos de interação em realidade virtual usando o Kinect e o WiiMote.

A sua colaboração constitui um factor importante para o êxito desta avaliação, por isso solicitamos-lhe o preenchimento deste questionário, cujos dados serão usados com total anonimato apenas para fins científicos.

1. Dados Pessoais

(assinale com uma cruz as opções correctas)

Género: ☐ Feminino ☐ Masculino

Lateralidade: ☐ Esquerdino ☐ Destro ☐ Ambidestro

Idade: _____

Tem experiência com dispositivos como ☐ Kinect, o ☐ comando da consola Wii, o ☐ Playstation Move, ou ☐ outros (_____)?

2. Teste de utilização de navegação

Após o teste de navegação e tendo em conta a sua avaliação, assinale com uma cruz o círculo que melhor reflecte a sua opinião em relação à utilização do sistema. Caso considere que estas quantificações não são aplicáveis, escolha NA.

2.1. Opinião sobre o método de navegação com e sem comando (preencha o círculo da opção que melhor corresponde à sua posição)

2.1.1. Sem comando, andando na sala (Kinect):

É fácil chegar à posição desejada.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A navegação é intuitiva.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A navegação tem algumas características irritantes.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A navegação melhoraria com treino.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○

2.1.2. Com comando (WiiMote):

É fácil chegar à posição desejada.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A navegação é intuitiva.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A navegação tem algumas características irritantes.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A navegação melhoraria com treino.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○

Se pretender pode deixar aqui outros comentários, ou expandir sobre a sua resposta à pergunta anterior:

2.2 Por favor indique o grau de satisfação na utilização de cada uma das formas de navegação:			
Sem Comando (Kinect):	1 Nada Satisfeito	○ ○ ○ ○ ○	5 Muito Satisfeito
Com Comando (WiiMote):	1 Nada Satisfeito	○ ○ ○ ○ ○	5 Muito Satisfeito

2.3 Das duas formas de navegação qual preferiu:		
Sem Comando (Kinect)	*Com Comando (WiiMote)*	Igualmente preferido
○	○	○

2.4 Se pretender, pode deixar aqui a razão da sua preferência:

3. Teste de utilização de menu

Após o teste de utilização de um menu e tendo em conta a sua avaliação, assinale com uma cruz o círculo que melhor reflecte a sua opinião em relação à utilização do sistema. Caso considere que estas quantificações não são aplicáveis, escolha NA.

3.1. Opinião sobre o método de seleção com e sem comando (preencha o círculo da opção que melhor corresponde à sua posição)

3.1.1. Sem comando (Kinect):

É fácil seleccionar a opção desejada.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção é intuitiva.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção tem algumas características irritantes.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção seria mais fácil com treino.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○

3.1.2. Com comando (WiiMote):

É fácil seleccionar a opção desejada.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção é intuitiva.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção tem algumas características irritantes.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção seria mais fácil com treino.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○

Se pretender pode deixar aqui outros comentários, ou expandir sobre a sua resposta à pergunta anterior:

3.2 Por favor indique o grau de satisfação na utilização de cada uma das formas de seleção:			
Sem Comando(Kinect):	1 Nada Satisfeito	○○○○○	5 Muito Satisfeito
Com Comando(WiiMote):	1 Nada Satisfeito	○○○○○	5 Muito Satisfeito

3.3 Das duas formas de seleção qual preferiu:		
Sem Comando (Kinect)	*Com Comando (WiiMote)*	Igualmente preferido
○	○	○

3.4 Se pretender, pode deixar aqui a razão da sua preferência:

4 Outros comentários/sugestões:

FIM

Muito obrigado pela sua colaboração

SECOND EXPERIMENT QUESTIONNAIRES

ID:____

Questionário ao utilizador para o Teste de interação com um menu e manipulação de objectos em Realidade Virtual

Instruções: Agradecemos a sua colaboração na realização deste estudo, que tem por objectivo avaliar dois métodos de interação com um menu e de manipulação em Realidade Virtual usando gestos (Kinect) ou um comando (WiiMote).

A sua colaboração constitui um factor importante para o êxito desta avaliação, por isso solicitamos-lhe o preenchimento deste questionário, cujos dados serão usados com total anonimato apenas para fins científicos.

1. Dados Pessoais

(assinale com uma cruz as opções correctas)

Género: ☐ Feminino ☐ Masculino

Lateralidade: ☐ Esquerdino ☐ Destro ☐ Ambidestro

Idade: _____

Tem experiência de utilização de dispositivos como ☐ Kinect, o ☐ comando da consola Wii, o ☐ Playstation Move, ou ☐ outros (_____)?

2. Interação com o menu (Kinect)

Assinale com uma cruz o círculo que melhor reflecte a sua opinião em relação à interação com o menu. Caso considere que estas quantificações não são aplicáveis, escolha NA.

2.1. Opinião sobre a interação com o menu (preencha o círculo da opção que melhor corresponde à sua posição)

É fácil selecionar a opção desejada	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção é intuitiva.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção tem algumas características irritantes.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○
A seleção causa cansaço	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente	NA	○

A seleção melhoraria com treino.	Discordo totalmente	○○○○○	Concordo totalmente	NA	○

Se pretender pode deixar aqui outros comentários, ou expandir sobre a sua resposta à pergunta anterior:

2.2 Por favor indique o grau de satisfação na interação com o menu:		
1 Nada Satisfeito	○○○○○	5 Muito Satisfeito

3. Interação com o menu (WiiMote)

Assinale com uma cruz o círculo que melhor reflecte a sua opinião em relação à interação com o menu. Caso considere que estas quantificações não são aplicáveis, escolha NA.

3.1. Opinião sobre a interação com o menu (preencha o círculo da opção que melhor corresponde à sua posição)

É fácil selecionar a opção desejada	Discordo totalmente	○○○○○	Concordo totalmente	NA	○
A seleção é intuitiva.	Discordo totalmente	○○○○○	Concordo totalmente	NA	○
A seleção causa cansaço	Discordo totalmente	○○○○○	Concordo totalmente	NA	○
A seleção tem algumas características irritantes.	Discordo totalmente	○○○○○	Concordo totalmente	NA	○
A seleção melhoraria com treino.	Discordo totalmente	○○○○○	Concordo totalmente	NA	○

Se pretender pode deixar aqui outros comentários, ou expandir sobre a sua resposta à pergunta anterior:

3.2 Por favor indique o grau de satisfação na interação com o menu:		
1 Nada Satisfeito	○○○○○	5 Muito Satisfeito

4. Manipulação de objectos (Kinect)

Assinale com uma cruz o círculo que melhor reflecte a sua opinião em relação à utilização do sistema para manipulação de objectos. Caso considere que estas quantificações não são aplicáveis, escolha NA.

4.1. Opinião sobre o método de manipulação (preencha o círculo da opção que melhor corresponde à sua posição)

É fácil rodar o objecto desejado.	Discordo totalmente	○○○○○	Concordo totalmente
É fácil escalar o objecto desejado.	Discordo totalmente	○○○○○	Concordo totalmente
É fácil posicionar o objecto desejado.	Discordo totalmente	○○○○○	Concordo totalmente
A manipulação causa cansaço	Discordo totalmente	○○○○○	Concordo totalmente
A manipulação é intuitiva.	Discordo totalmente	○○○○○	Concordo totalmente
A manipulação tem algumas características irritantes.	Discordo totalmente	○○○○○	Concordo totalmente
A manipulação seria mais fácil com treino.	Discordo totalmente	○○○○○	Concordo totalmente

Se pretender pode deixar aqui outros comentários, ou expandir sobre a sua resposta à pergunta anterior:

_____

_____

4.2 Por favor indique o grau de satisfação na manipulação do objecto:		
1 Nada Satisfeito	○○○○○	5 Muito Satisfeito

5. Manipulação de objectos (WiiMote)

Assinale com uma cruz o círculo que melhor reflecte a sua opinião em relação à utilização do sistema para manipulação de objectos. Caso considere que estas quantificações não são aplicáveis, escolha NA.

5.1. Opinião sobre o método de manipulação (preencha o círculo da opção que melhor corresponde à sua posição)

É fácil rodar o objecto desejado.	Discordo totalmente	○○○○○	Concordo totalmente
É fácil escalar o objecto desejado.	Discordo totalmente	○○○○○	Concordo totalmente

É fácil posicionar o objecto desejado.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente
A manipulação causa cansaço	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente
A manipulação é intuitiva.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente
A manipulação tem algumas características irritantes.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente
A manipulação seria mais fácil com treino.	Discordo totalmente	○ ○ ○ ○ ○	Concordo totalmente

Se pretender pode deixar aqui outros comentários, ou expandir sobre a sua resposta à pergunta anterior:

_____

_____

5. Das duas formas de interação com o menu, qual preferiu		
Sem Comando (Kinect)	*Com Comando (WiiMote)*	Igualmente preferido
○	○	○

6. Das duas formas de manipulação, qual preferiu (para as diferentes manipulações)			
	Sem Comando (Kinect)	*Com Comando (WiiMote)*	Igualmente preferido
Posicionamento	○	○	○
Rotação	○	○	○
Escala	○	○	○

FIM

_____