VR Plugin: A Virtual Reality Plugin for Unity Applications

Master Thesis

Jorge Troya Moreno

Madrid, January 2017
This thesis is submitted to the ETSI Informáticos at Universidad Politécnica de Madrid in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering.

Master Thesis
Master Universitario en Ingeniería del Software – European Master in Software Engineering
Thesis Title: VR Plug: A Virtual Reality Plugin for Unity Applications.
Thesis no: EMSE-2017-01
January 2017

Author: Jorge Troya Moreno
Academic title: Master of Science in Software Engineering
University of the presented title: Universidad Politécnica de Madrid

Supervisor: Cristian Moral Martos
Ph.D. in Computer Science
Universidad Politécnica de Madrid
Dpto. Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Co-supervisor: Angélica de Antonio Jiménez
Ph.D. in Computer Science
Universidad Politécnica de Madrid
Dpto. Lenguajes y Sistemas Informáticos e Ingeniería de Software
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

ETSII Informáticos
Universidad Politécnica de Madrid
Campus de Montegancedo, s/n
28660 Boadilla del Monte (Madrid)
Spain
# Table of Content

Table of Content ......................................................................................................................... I  
Table of Figures ............................................................................................................................ III  
Acknowledgments ........................................................................................................................... V  
Abstract ........................................................................................................................................ VI  
Abbreviations and Acronyms .......................................................................................................... VII  

1 Introduction ................................................................................................................................ 1  

2 Theoretical Framework ............................................................................................................... 3  

2.1 Interaction Techniques ............................................................................................................. 3  

2.2 Middle VR ............................................................................................................................... 3  

2.2.1 General description ............................................................................................................. 3  

2.2.2 Analysis .............................................................................................................................. 4  

2.3 VR Juggler .............................................................................................................................. 6  

2.3.1 General description ............................................................................................................ 6  

2.3.2 VR Juggler components ..................................................................................................... 7  

2.3.3 Analysis ............................................................................................................................ 8  

2.4 Devices Used in this Project ................................................................................................... 8  

2.4.1 Oculus Rift ......................................................................................................................... 9  

2.4.2 Leap Motion ...................................................................................................................... 11  

2.4.3 Unity 3D Assets ................................................................................................................. 13  

3 Unity 3D ..................................................................................................................................... 15  

3.1 Hierarchy Window .................................................................................................................. 15  

3.2 Scene Window ......................................................................................................................... 16  

3.3 Inspector Window .................................................................................................................. 17  

3.4 Game Window ....................................................................................................................... 18  

3.5 Project Window ..................................................................................................................... 19  

3.6 Console Window ..................................................................................................................... 20  

3.7 Scripting ................................................................................................................................. 20
3.8 Project Structure ........................................................................................................... 21

4 Solution ........................................................................................................................... 22
4.1 Criteria solution ............................................................................................................. 22
  4.1.1 Quality attributes .................................................................................................... 22
4.2 Features to consider ..................................................................................................... 24
  4.2.1 Basic features ........................................................................................................ 24
  4.2.2 Previous activities ................................................................................................ 25
  4.2.3 Devices .................................................................................................................. 25
4.3 Solution design ............................................................................................................. 27
  4.3.1 UI evolution .......................................................................................................... 27
  4.3.2 Final design ........................................................................................................... 31
  4.3.3 Plugin Design .......................................................................................................... 33
5 Updating the project components .................................................................................. 48
  5.1 Updating the Project Components ............................................................................. 48
    5.1.1 Oculus Rift ......................................................................................................... 48
    5.1.2 Leap Motion ......................................................................................................... 48
    5.1.3 Mouse 3D ............................................................................................................ 49
    5.1.4 StereoWall .......................................................................................................... 49
    5.1.5 Gamepad ............................................................................................................. 49
    5.1.6 Unity 3D .............................................................................................................. 49
  5.2 Minimum Requirements for Use ............................................................................... 51
  5.3 Project Structure ....................................................................................................... 52
6 Extensibility and Reuse ................................................................................................... 54
7 Evaluation ........................................................................................................................ 57
  7.1 Internal Test ............................................................................................................. 57
  7.2 Real Test Evaluation ................................................................................................. 58
8 Conclusions ..................................................................................................................... 60
  8.1 Future Works ............................................................................................................. 61
9 References ...................................................................................................................... 62
Table of Figures

Figure 1: Middle VR Product comparison .............................................................. 4
Figure 2: Headset and control box ............................................................... 10
Figure 3: Set of lens ............................................................................... 10
Figure 4: View of hands in desktop mode .................................................. 11
Figure 5: Example of using this device ....................................................... 12
Figure 6: Unity 3D Asset Store - Website ............................................. 13
Figure 7: Unity 3D Asset Store - Unity Editor ........................................... 14
Figure 8: Unity 3D Editor ..................................................................... 15
Figure 9: Hierarchy Window ................................................................... 16
Figure 10: Scene Window .................................................................... 16
Figure 11: Inspector Window .................................................................. 17
Figure 12: Game Window ..................................................................... 18
Figure 13: Play Mode Buttons ............................................................... 18
Figure 14: Project Window ................................................................... 19
Figure 15: Console window .................................................................. 20
Figure 16: MonoDevelop editor ............................................................. 20
Figure 17: Folder Structure Suggested .................................................. 21
Figure 18: First UI design ................................................................... 21
Figure 19: Second iteration, visualization category ................................ 28
Figure 20: Second iteration, manipulation category - mouse .................. 29
Figure 21: Second iteration, manipulation category – Leap Motion ........... 29
Figure 22: Visualization Device layout – for keyboard device ................. 30
Figure 23: Manipulation Device layout .................................................. 30
Figure 24: Navigation Device layout ...................................................... 31
Figure 25: Resolution dialog, presented to end-users ............................. 32
Figure 26: Visualization devices list ...................................................... 33
Figure 27: Manipulation devices list ...................................................... 34
Figure 28: Navigation devices list .......................................................... 34
Figure 29: Complete view of UI ............................................................. 35
Figure 30: Oculus Rift settings for player controller ............................. 36
Figure 31: Oculus Rift settings for camera rig ......................................... 37
Figure 32: Stereo Wall device ................................................................. 37
Figure 33: Stereo Wall settings ................................................................. 38
Figure 34: Point of view section .............................................................. 38
Figure 35: Camera settings with keyboard .............................................. 39
Figure 36: Camera settings with gamepad .............................................. 40
Figure 37: Set the objects to move in design mode ................................. 40
Figure 38: Script to add object to move in runtime ................................ 41
Figure 39: Keyboard Settings for navigation ......................................... 41
Figure 40: Gamepad Settings for navigation ......................................... 42
Figure 41: SpaceNavigator basic settings .............................................. 43
Figure 42: Mouse 3D Additional Settings .............................................. 43
Figure 43: Import a Custom Package .................................................... 44
Figure 44: Import of Unity Package dialog box ...................................... 45
Figure 45: Project window view ............................................................ 46
Figure 46: Input manager - project settings ......................................... 46
Figure 47: Import custom package ....................................................... 50
Figure 48: Select the components to import ......................................... 50
Figure 49: Copy the file structure from origin to destination .................. 51
Figure 50: Folder Structure for this project ......................................... 52
Figure 51: Global structure of project folders ...................................... 52
Figure 52: Export package window ...................................................... 53
Figure 53: ManagerDevice.cs script extract ....................................... 54
Figure 54: Visualization section ............................................................ 55
Figure 55: Manipulation section ............................................................. 56
Figure 56: Navigation section ............................................................... 56
Figure 57: Internal test with Ultimate Fantasy Creator 3D environment .... 57
Figure 58: Installing the solution on the test project ............................. 59
Acknowledgments

I would especially like to thank Viviana, my friend and wife. She has given me her support in every way so I can finish this master’s degree. To my children Gabriela, Federico and Beatriz who with their joy have been a great motivation to my life.

Finally, I would like to thank God that through this work and the master in general he has formed my character and taught me that to the one who believes, everything is possible.
Abstract

Software development for Virtual Reality (VR) has been popularized in 2016, alongside products such as Unity 3D and Oculus, especially in fields such as video games, tourism, media and marketing. But software development for VR is complex because additional requirements must be added to software that are not normally required. Newcomers to the Decoroso Crespo Laboratory, who join new groups to develop VR software using Unity 3D as a development platform, find it difficult to integrate some of the devices the laboratory has, such as Oculus, Leap Motion, Gamepad, Stereoscopy Screen and 3D mouse. They must learn to master Unity 3D on their own and then integrate the devices they need to use in the project. This process can take time for a newcomer, and time spent in digesting this knowledge and then implementing the development project can affect the duration of the project itself. This master thesis is a contribution for these groups to save time when starting a project in Unity 3D, through a plugin that can transparently configure the devices that they want to use in the project, minimizing the effort and time spent in configuring and integrating these devices independently.
# Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>CAVE</td>
<td>Cave Automatic Virtual Environment.</td>
</tr>
<tr>
<td>HMD</td>
<td>Head Mounted Display.</td>
</tr>
<tr>
<td>SDK</td>
<td>Software Development Kit.</td>
</tr>
<tr>
<td>VR</td>
<td>Virtual Reality.</td>
</tr>
</tbody>
</table>
1 Introduction

Virtual Reality has allowed users to immerse themselves in an artificial environment simulated by computer. This system lets the user navigate through the environment and interact with the elements created. The user experiences a different degree of immersion, depending on the hardware used to generate this environment and the software that generates the simulation of this environment (Okechukwu, Ezeani, & Raphael, 2012).

One promise of VR is to simplify the way in which humans and computers interact. This one has the potential to open new ways of interaction that are not currently possible. The creation of new experiences with VR has opened new expectations towards how to use the current technology. An example of the promise of VR is the possibility of using VR to take virtual tours of distant locations (Bierbaum et al., 2002). In 2001, it was not possible to carry out this promise because of the state of technology in that year. But in 2016, tourism is taking the first steps in the use of VR (Jiménez Caro, 2016; Las Vegas, 2016). This has been made possible by the development of new technologies, supported by private companies who provide the necessary financial resources to make these devices a reality, as well as seeking to open a new market in which these technologies can be used.

A particular case we can highlight is the Oculus Rift. Since 2015, this device has had great relevance, especially in the area of entertainment mainly in video games, but this has been extended to other areas such as education and research. Next to Oculus has been other manufacturers like Samsung, Sony, Google, HTC and Razer Hydra, which offer similar functionalities and they have focused on the same areas. But the Oculus Rift has been the device that has marked the trend since it made its SDK public. Anyone can create a virtual environment in which to use this device and this was adopted by a community of developers and companies dedicated to video games, media and marketing.

The tools for VR development have also experienced a great boom, especially Unity 3D, whose niche market is mainly video games, but has extended to other areas such as education and research. This great acceptance by developers, small or independent companies and educational institutions is due to the business model that Unity 3D has applied for this tool to be widely used. Unity 3D has done this by providing the community with a free tool in version 5 for personal or professional use which has all the features of the
engine and the editor. In version 4, it was necessary to pay to obtain better performances of this tool, the engine and the publisher, among others. Another way that Unity has achieved this great success is providing extensive technical support, lots of documentation and a wide community of developers. It has allowed Unity to be adopted more easily. It should be noted that in version 5, Unity 3D offers other features that are pay as cloud-powered, analytics and performance crash reporting.

It is not the same with other devices and tools for VR development, such as CAVE, graphics engines, manipulation devices, among others. Since these are not standard products, they apply to only certain areas of research, or they do not have sufficient support from public or private institutions, or they are not mass-use products like Oculus. Or they are simply hardware and software that have become outdated by the passage of time. Moreover, there is not enough support from the developer community because they do not have enough documentation and technical support from the manufacturers. So, when starting a VR Project, a newcomer needs time to adapt to the use of these tools, making it more difficult in achieving the goals of the Project. Since the systems in VR make use of many devices and components, each one behaves differently and requires different levels of systems to interact together, which makes these systems complex. These aspects distinguish VR applications from traditional interactive computer applications because they place many additional requirements on the software that are not normally required (Bierbaum et al., 2002).

Given these circumstances, this Master’s Thesis wants to contribute to the VR development in the Decoroso Crespo Laboratory with a plugin that will save time in using the devices that are in the laboratory like Oculus Rift, Leap Motion, and 3D Mouse. When a new Project is started, this will minimize the effort and time spent configuring and integrating these devices.
2 Theoretical Framework

2.1 Interaction Techniques

For interaction, the use of cameras is important, because through which the user can see the world and interact with these. It has two uses: camera rig and player controller. As camera rig, is fixed, it does not have a movement itself. Generally, is placed close to other object which has movement. It is similar when a player puts a camera like GoPro close to his helmet or his chest. As player controller, the camera has movement itself and it can interact with the world.

2.2 Middle VR

2.2.1 General description

Is a middleware for VR that simplifies the creation and development of VR applications (Ritter, Borst, & Chambers, 2016). Middle VR support different systems like:

- Any VR system: HMDs (like the Oculus Rift), Caves, stereoscopic walls, zSpace, 3DTVs.
- Interaction devices like 3D trackers.
- Stereoscopy: active, passive.
- Clustering: Selenock, swaplock both software & hardware.
- 3D interactions: Navigation, manipulation, menus and custom graphical user interfaces.

For these systems, MiddleVR has a series of presets configurations that can be used with different set of cameras, screened and viewports. For the platform Unity 3D MiddleVR has the following features (middle VR, 2015):

- Scale one visualization with user-centric perspective.
- Support for 3D interaction devices such as 3D trackers (see full list on the right).
- S3D – Stereoscopy (active, passive).
- Multi-screens / multi-computers synchronization for higher-resolutions and impressive VR systems.
- 3D interactions: navigation, manipulation.
- Immersive menus.
- Custom graphical user interfaces (in HTML5).
- Display any web page inside your virtual world.

2.2.2 Analysis

MiddleVR can be used under four editions: Free edition, HMD edition, Academic edition and PRO edition. In the Figure 1, each edition supports different features. The price is not public and it is necessary contact the company for more information. This middleware is a commercial product and the source code is closed.

<table>
<thead>
<tr>
<th>Devices</th>
<th>Features</th>
<th>MiddleVR &quot;Basic&quot; edition (forever line)</th>
<th>MiddleVR &quot;HMD&quot; edition (including the 30 days trial)</th>
<th>MiddleVR &quot;Academic&quot; edition (including the 30 days trial)</th>
<th>MiddleVR &quot;PRO&quot; edition (including the 30 days trial)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kinect</td>
<td>Full body tracking</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Oculus Rift</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Leap Motion (SDK 1)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Leap Motion (SDK 2)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>trackit</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Laser Hyra</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SpaceTrack</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SpaceMouse</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>InterSense i500</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>A.R. native driver</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Vicon native driver</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Optitrack native driver</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>CellPi</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Organic Motion’s WalkieTracker Manager</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>VRX (Oculus, A.R. Optitrack, Instrime...)</td>
<td>Limited to 2 trackers, 2 analog axes and 4 buttons</td>
<td>Unlimited</td>
<td>Limited</td>
<td>Unlimited</td>
<td>Unlimited</td>
</tr>
<tr>
<td>Haptic’s haptic devices</td>
<td>No</td>
<td>With additional license</td>
<td>With additional license</td>
<td>With additional license</td>
<td>With additional license</td>
</tr>
<tr>
<td>Interactions</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Navigators</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Manipulations</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Immersive Web/View</td>
<td>Limited to 128x768</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Immersive Menu (HTM/L3)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Immersive custom SLAs (HTM/L2)</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Displays</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Viewports</td>
<td>2, each limited to 1280x1024</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Unlimited</td>
<td>Unlimited</td>
</tr>
<tr>
<td>360 View</td>
<td>360° Unlimited</td>
<td>360° Unlimited</td>
<td>360° Unlimited</td>
<td>360° Unlimited</td>
<td>360° Unlimited</td>
</tr>
<tr>
<td>Cluster</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Platforms</td>
<td>Windows 32-bit</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Windows 64-bit</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Support</td>
<td>Support</td>
<td>Professional support</td>
<td>Professional support</td>
<td>Professional support</td>
<td>Professional support</td>
</tr>
<tr>
<td>Licensing</td>
<td>conditions</td>
<td>Commercial</td>
<td>Commercial</td>
<td>Commercial</td>
<td>Commercial</td>
</tr>
<tr>
<td>Side Screen</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Watermark</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>No royalty publishing</td>
<td>Non-commercial only</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

Note: 1 Kinect V2 requires Windows 8.

Figure 1: Middle VR Product comparison
As displayed on Figure 1, the free edition has less support in devices, interactions, display, cluster and support sections, for the case of infrared cameras there is no support in free edition. All the editions, except Free edition has a free 30 days’ trial. With which it is possible to test all the functionalities. For this project, it was used the Academic edition 30 days’ trial. To use with Unity 3D, the workflow is created on Configurator Window a description of the VR system and then use this file into Unity 3D editor. The description file includes the devices to use, trackers, physical screens and cameras used for the project in Unity 3D.

MiddleVR has in its favor a powerful graphical configuration tool that let the user to adapt his application to his hardware available. It has support for different types of hardware like display environments, viewers and provides distributed rendering capability, it provides immersive menus, immersive webview and immersive custom GUI. MiddleVR has disadvantages the support only for Windows OS, the price of this tool, the lack of technical support for the free edition, the limitation for large high-resolution displays for free edition. For educational organizations where most of their developments are made with free software this tool does not provide such support (Sigitov, Scherfgen, Hinkenjann, & Staadt, 2015). The Configurator window is complex to use and since it works independently from Unity 3D.

MiddleVR provides drivers for different VR devices like Oculus Rift, Kinect, Leap Motion, Razer Hydra, and so on. And this features offers an abstraction layer that enable the application not depend on particular hardware, making it flexible (Kuntz, 2015). This tool does not support the definition and analysis of gestures (Odio-Vivi, 2015). However for CAVE-like infrastructure (Cave Automatic Virtual Environment), the loss of synchrony among the unity instances created by MiddleVR for each node of the cluster is a limitation because of non-deterministic features like physics or IA are used in the project (Gaugne, Barreau, Cousseau, & Gouranton, 2012), although for this project represents a minor issue, since the Laboratory does not have this type of infrastructure.
2.3 VR Juggler

2.3.1 General description

VR Juggler is middleware designed for the creation of cross-platform, cross-VR-system immersive software applications. This project was started in 1997 at Iowa State University's Virtual Reality Applications Center (Virtual Reality Applications Center, 2012). During the preparation of this document, the latest version is 3.0. Its main features are:

- Run-time reconfiguration so that applications can be reconfigured while they are running.
- Support for multi-pipe hardware.
- Support for a wide variety of input devices used in VR systems.
- Passive and active stereoscopic rendering.
- Cross-platform abstractions to make VR applications much more portable across operating systems.
- Support for clusters (VR Juggler 2.0 and newer).
- Support for use of different graphics programming interfaces (OpenGL, OpenGL Performer, OpenSG, and Open Scene Graph).
- Application programmers use the graphics API of their choice directly and let VR Juggler set up the projections and other window- and pipe-related parameters.

In VR Juggler, user applications are objects. The VR Juggler system uses the application object to create the VR environment in which the user interacts. The application object implements interfaces needed by the virtual platform to create the virtual environment. The kernel maintains control over the environment and calls the methods defined in the application interface. When the kernel calls the application's methods, it gives up control to the application object so the application can execute the code needed to create the virtual environment (Bierbaum et al., 2002).

The VR Juggler allows nearly every parameter to change at run time. It is possible for this system to change applications, start new devices, reconfigure devices, and send reconfiguration information to the application object. One of the strengths of VR Juggler is the ability to modify the behavior of the system at runtime, because the application developed is an object with a standard public interface (Bierbaum et al., 2002).
The VR Juggler goals is to maximize flexibility, portability, extensibility, and maintainability – of both applications and the library itself. VR Juggler was designed with these long-term considerations in mind. Starting with a clearly defined, object-oriented design, and written from the ground up in C++, VR Juggler aims to give the application developer support for writing any kind of VR application, in an easily extended or modified environment, without sacrificing performance. VR Juggler provides a "virtual platform" for application development. It provides a single interface to the functions of the underlying operating system. This system supports several other kinds of input data, including analog, digital, and keyboard inputs. Each input type has an associated proxy interface which can be accessed by the application.

2.3.2 VR Juggler components

This is an overview of the main components of this system (C. D. Just, 2000), to provide some background of its structure.

- **Cross-platform system primitives.** It provides a set of primitives for accessing various low-level operating system capabilities (threading, synchronization, network sockets), These primitives are also used internally by the other VR Juggler components.

- **Microkernel.** It has a small Microkernel object which acts as a central communication broker between the application and the different components of this framework. The Microkernel also contains a single thread which controls several aspects of the system.

- **Input manager.** This component owns and controls access to all input devices. It is responsible for initializing and running all input device drivers. It also provides access to input proxies to the rest of the VR Juggler components.

- **Display manager.** This component is responsible for creating and managing graphics windows, and managing the drawing threads. It also works with the kernel to synchronize buffer swaps of all graphics windows.
• **Draw manager.** This component works in concert with the Display Manager, and provides graphics API-specific functionality.

• **Environment manager.** Their primary responsibility is to receive dynamic reconfiguration commands from the outside world. It creates a network socket through which other applications can query the current configuration and send instructions for changing the configuration. These instructions are forwarded to the Kernel and then onto the various managers.

2.3.3 Analysis

VR Juggler attempts to provide a "virtual platform" for application development. Operating System and hardware specific details (process/thread management, input device access, window and display creation, etc.) are hidden behind common, abstract interfaces. This system does not have a GUI library, although it does handle input from a wide variety of devices. VR Juggler works with VjControl, a GUI which allows users to connect to VR Juggler applications. It allows to collect and display performance information (C. Just, Cruz-Neira, & Baker, 2000).

The learning curve for VR Juggler is remarkable for new users, while they have trouble to understand the basic concept of this system like how to configure the system and how proxies works. Because the VR Juggler Team does not offer complete tutorials and documentation of how to use this system, the new users does not desire to take advantage of all features offered by this system because many of these concepts are new for the users (Bierbaum et al., 2002).

2.4 Devices Used in this Project

For this Thesis, the devices used are: Oculus Rift, Leap Motion, Mouse 3D, StereoWall Display and Gamepad, as well as using keyboard, mouse and display.
2.4.1 Oculus Rift

The version of this HMD (Head Mounted System), that the Laboratory has is DK1 (see Figure 2). The main features of this device are:

- Screen Resolution: 1280 x 800 px (640 x 800 per eye).
- Pixel Layout: RGB.
- OLED: No.
- Screen Size: 7”.
- Screen Manufacturer and model: Innolux HJ070IA-02 D 7” LCD.
- Latency: 50ms – 60ms.
- Low Persistence: No.
- Refresh Rate: 60Hz.
- Orientation Tracking: Yes.
- Positional Tracking: No.
- Gyroscope, Accelerometer, Magnetometer: Yes.
- FOV: 110.
- 3D: Stereoscopic.

This device falls short is the positional tracking and compared with DK2, the screen works at 60Hz, and the recommendation for HMD (Head Mounted Display) is 90Hz. For people who wear glasses is uncomfortable because the pressure their glasses make (see Figure 3).
Figure 2: Headset and control box

Figure 3: Set of lens
2.4.2 Leap Motion

This device supports hand and finger motions as input, analogous to a mouse, but requires no hand contact or touching. Leap Motion is composed of infrared LEDs and two cameras underneath its black glass top, the software tracks the movement of the fingers as you move them above the sensor. It has a little connector port for its included USB cable, and a small green LED light on the front that lights up when it’s plugged in.

As displayed on Figure 4, the sensors have a field of view of about 150 degrees. The effective range of the Leap Motion Controller extends from approximately .03 to .6 meters above the device (1 inch to 2 feet).

This device is compatible with Windows and Mac. It mostly works with software from the Airspace Store, both on desktop and from the website (Leap Motion, 2016). The Airspace Store is designed for this device and where you can download the apps developed for Leap Motion. The driver is updated frequently, so you can be sure you have the latest version.

![Figure 4: View of hands in desktop mode.](image)

Leap Motion has an integration with virtual reality through Unity3D and Unreal game engines. For Unity3D include a built-in support for VR development (see Figure 5). For this topic, the controller must be placed into the HMD (Head Mounted Device), for better experience with the help of a small device that is attached to the front of the HMD.
Leap Motion has the following limitations when is used with Unity 3D:

- By putting hands singing, the gestures are not correctly detected.
- Keeping arms in the air while seated, can cause fatigue shortly after use. That’s why you should consider correct positions for its proper use.
- It is recommended not to wear rings, metal bracelets or watches while you are using the device, as it may cause it to fail its accuracy.
- If there is much light in the room, the device can fail and cannot be correctly render hands.
- There were problems with the assets that the company has, since from the demos they provide, to its implementation requires considerable refinement to make them work as Unity 3D demos.
- The device heats up soon after use, which can cause it to stop working properly.

When this device is used to supplement the Oculus Rift inside Unity 3D, has the following limitations:

- The user must to keep the arms in the air close to the HMD, to use both devices. This can cause fatigue to leave hands in the air all the time.
2.4.3 Unity 3D Assets

It is a repository of free and paid-for assets like editor extensions, scripts, 3D models, audio, etc. where developers, game designers, and 3D modelers can find anything they need for their projects. The asset store has twelve categories (3D Models, animation, applications, audio, complete projects, editor extensions, particle systems, scripting, services, shaders, textures & materials and Unity essentials), as displayed on Figure 6.

![Figure 6: Unity 3D Asset Store - Website]

To download an asset, it is necessary to install Unity 3D because it is opened from Unity (see Figure 7). It is not possible to download it in whatever folder of the PC, else it is downloaded in background to be used in the project that you want to work. Additionally, it is necessary to register in this repository and do login to download the asset.
For the devices used in this thesis, almost all the manufactures provide their assets in package format and can be downloaded outside the asset store. Except for the device “SpaceNavigator” from 3DConnexion, which it is available in asset store and GitHub. Despite not being an official 3DConnexion application. Only there are two applications for SpaceNavigator: 3DThrough and SpaceNavigator Drive. The most updated is SpaceNavigator Drive (March 03, 2016). 3DThrough (Feb 14, 2014)

![Figure 7: Unity 3D Asset Store - Unity Editor](image)
3 Unity 3D

This section describes what provides this platform for development in virtual reality. Along this Master Thesis the VR development with Unity 3D has been changed and evolved positively (from 4.5 to 5.3 at the time of this writing), facilitating the development. It should only consider this software updating that at some point will require hardware update. For this Master Thesis, the hardware devices have been the same, so the software updates arrive until a certain point to get the most out of the hardware.

Initially, Unity 3D has the following structure when it is opened as displayed on Figure 8.

![Unity 3D Editor](image)

Figure 8: Unity 3D Editor

3.1 Hierarchy Window

The Hierarchy window contains every GameObject in the current Scene (see Figure 9). Some of these are direct instances of asset files like 3D models, and others are instances of Prefabs, custom objects that will make up much of your game. It possible to select objects in the Hierarchy and drag one object onto another to make use of Parenting. As objects are added and removed in the scene, they will appear and disappear from the Hierarchy as well. By
default, the Game Objects will be listed in the Hierarchy window in the order they are made. It is possible to re-order the Game Objects by dragging them up or down, or making them children or parents.

![Hierarchy Window](image)

**Figure 9: Hierarchy Window**

### 3.2 Scene Window

This window is the interactive view into the created world. As displayed on Figure 10, it will use the view to select and position scenery, characters, cameras, lights, and all other types of Game Object. Being able to Select, manipulate and modify objects in this view.

![Scene Window](image)

**Figure 10: Scene Window**
3.3 Inspector Window

The Inspector is used to view and edit the properties and settings of Game Objects, Assets, and other preferences and settings in the Editor (see Figure 11). When is selected a GameObject in the Hierarchy or Scene View, the Inspector will show the Properties of all Components and Materials on that object and allow you to edit them.

![Inspector Window](image)

*Figure 11: Inspector Window*
3.4 Game Window

The Game View is rendered from the Camera(s) in the project (see Figure 12). It is representative of the final project. It is necessary to use one or more Cameras to control what the player actually sees when they are running the project.

![Game Window Image](image)

*Figure 12: Game Window*

Use the buttons in the Toolbar to control the Editor Play Mode and see how the project play (see Figure 13). While in Play mode, any changes made are temporary, and will be reset when exit Play mode. The Editor UI will darken to remind this.

![Play Mode Buttons](image)

*Figure 13: Play Mode Buttons*
3.5 Project Window

The left panel shows the folder structure of the project as a hierarchical list. When a folder is selected from the list by clicking, its contents will be shown in the panel to the right. You can click the small triangle to expand or collapse the folder, displaying any nested folders it contains. Hold down Alt while you click to expand or collapse any nested folders recursively (see Figure 14).

The individual assets are shown in the right-hand panel as icons that indicate their type. The icons can be resized using the slider at the bottom of the panel; they will be replaced by a hierarchical list view if the slider is moved to the extreme left. The space to the left of the slider shows the currently selected item, including a full path to the item if a search is being performed.

![Figure 14: Project Window](image-url)
3.6 Console Window

The Console Window shows errors, warnings and other messages generated by Unity, as displayed on Figure 15.

![Figure 15: Console window](image)

3.7 Scripting

For scripting Unity 3D provides IDE MonoDevelop, which is installed by default. This tool combine the familiar operation of a text editor with additional features for debugging (see Figure 16). MonoDevelop works with C# and JavaScript mostly. When Unity launched 5.0 version, they dropped the support for Boo language (Aleksandr, 2014).

![Figure 16: MonoDevelop editor](image)
3.8 Project Structure

Each project could have the folder structure that is deemed necessary as displayed on Figure 17, but generally can set the following structure:

![Folder Structure Suggested](image)

- **Editor.** In this folder, it should be the scripts for custom some components (more friendly), in the Inspector Window.
- **Fonts.** This is for store the font files will be used in the project.
- **Materials.** With the materials is possible to control the visual appearance of the game objects.
- **Models.** A model file may contain a 3D model, such as a character, a building, or a piece of furniture. The model is imported as multiple assets.
- **Resources.** Unity supports Resource Folders in the project to allow content to be supplied in the main project file yet not be loaded until requested.
- **Scenes.** Contain the objects of your project. Here it will place the environments, obstacles, and decorations, essentially designing and building your project in pieces.
- **Scripts.** In this folder, it should be included the scripts files written for the project.
- **Sounds.** If the project will use audio clips (.aif, .mp3, .wav, .ogg), these files should be placed in this folder.
- **Textures.** A texture is just a standard bitmap image that is applied over the mesh surface. The positioning of the texture is done with the 3D modelling software that is used to create the mesh.
4 Solution

4.1 Criteria solution

To find a solution for this project and better understand the problem, interviews were conducted with members of a research laboratory. Initially they were informal interviews, contact-making, familiarization with the laboratory environment. Later, more detailed interviews were conducted to solve doubts as to the operation of the devices that the laboratory has and with respect to the functionalities of the solution to establish the software requirements. Subsequently in the development phase, meetings in which the progress was presented were held. The decisions made regarding the design of the solution were discussed and feedback was received from the users, with whom the initial requirements were refined during the design and development of the project.

4.1.1 Quality attributes

This section describes the quality attributes that have been taken into account for the design of the solution.

- **Availability.** It is defined as the proportion of time that the system is functional and working. It can be measured as a percentage of the total system downtime over a predefined period (Microsoft Patterns & Practices Team, 2009). Availability will be affected by system errors, infrastructure problems, malicious attacks, and system load. Availability is always a concern when considering a system’s dependability, though to varying degrees, depending upon the application (Barbacci, Klein, Longstaff, & Weinstock, 1995).

For this project the availability depends on the development platform on which it has been implemented (Unity 3D), and this project is installed correctly, following the form of work that Unity 3D recommends. Its correct functioning will depend on not changing the structure of directories created for this purpose.
• **Reliability.** It is a measure of the ability of a system to keep operating over time. The reliability of a system is typically measured as its mean time to failure (MTTF), the expected life of the system (Barbacci et al., 1995).

In order to comply with this attribute, it was established that when an error was made by the user when entering a value in a certain parameter, the system has to inform the user of an unexpected behavior that can generate these values or that Unity 3D stops working.

• **Integrity.** It is the non-occurrence of the improper alteration of information. Also, in more general terms, it may be used for either data or processes to specify how modifications are made to data or how control is passed to processes (Barbacci et al., 1995).

To comply with this attribute, it is established that the information that has been entered as parameter values is not altered by other processes. It has been determined that these solution parameter values are given at the design time, so that at runtime they are not altered by other processes.

• **Maintainability.** The maintainability of a system is its aptitude to undergo repair and evolution (Barbacci et al., 1995). These changes could impact components, features, and interfaces when adding or changing the functionality, fixing errors, and meeting new requirements (Microsoft Patterns & Practices Team, 2009).

For this attribute a modular design was established, in which each device must operate independently and that does not affect the operation of the solution in general. All the changes that are made or new devices that are added must be allowed can do without major restrictions or side effects on other devices.

• **Reusability.** It defines the capability for components and subsystems to be suitable for use in other applications and in other scenarios. Reusability minimizes the duplication of components and also the implementation time (Microsoft Patterns & Practices Team, 2009).
In order to comply with this attribute, the solution must work transparently with the different devices, so that if one of them is changed, the system can adapt and continue with its execution normally.

4.2 Features to consider

As a first approach to the solution, the main goals to be achieved in this project were established:

*The purpose is to develop a plugin that allows laboratory newcomers to save time in the use of the devices when starting a new project using Unity 3D as a development tool.*

*Facilitate the use of devices in new virtual reality projects, minimizing the effort and time of configuration and integration of these in the project.*

Since beginners will have to familiarize with this tool, the objective of this solution is to help them incorporating one or more devices into a new virtual reality project, so that they are fully functional and work in a coordinated way. Delegating in this solution its operation and coordination will allow the users focusing on other areas of the project.

4.2.1 Basic features

These are the functionalities to be implemented during the initial phase.

- The user can select the devices to use according to three categories: Visualization, Manipulation and Navigation:
  - For display devices, the user must be able to select between two modes: first person or third person. On first person, it is a view that is used in video games which the world is seen from the perspective of the player. On third person, the player being controlled is seen in full body and usually on his back.
  - For manipulation devices, the user has to be able to select the objects that it wants to move in the 3D environment
  - For navigation devices, the user has to be able to select the buttons (joystick), or keys (keyboard) that he wants to use for navigation in the 3D environment.
4.2.2 Previous activities

Once the basic functionalities were established, two previous activities were carried out. First, we became familiar with the use of the tool on which the developments are done in the laboratory (Unity 3D), then adopt the use guides that this platform establishes at the moment of implementing a plugin.

Secondly, we learn how to use each of the selected devices and we studied how they can be incorporated into a project developed in Unity 3D. Once these activities were completed, the solution we started the design of the solution.

4.2.3 Devices

4.2.3.1 Devices to be used

In the first instance, it was determined that the devices to be used would be:

- **Oculus Rift.** It is a virtual reality system that immerses the user into virtual worlds. The lab uses the DK1 version of this system.
- **Leap Motion Controller.** It is a gesture-control device. This device has two cameras and infrared LEDs underneath its black glass top.
- **Stereoscopic screen.** It is a visualization 3D system used to enabled a three-dimensional effect, adding an illusion of depth to a flat image.
- **Infrared cameras.** They are an optical motion system. In the lab there are four cameras installed. They are synchronized to compute 3D position via triangulation. The lab uses the cameras: OptiTrack Flex 3 model.
- **Mouse 3D.** It is a 3D motion controller with 6 degrees of freedom, specially designed for manipulation and navigation computer-generated 3D imagery. For this project the product used was Space Navigator.
- **Joystick and keyboard.**

Given the number of devices that had to be integrated, it was decided to carry out an iterative and incremental development, i.e. establishing working blocks with each device to be used, so that at the end of the project the expectations of the laboratory members could be met. Because of this, while the development was carried out, the requirements were refined in each iteration until these requirements were fulfilled by the laboratory.
Once the familiarization phase with Unity 3D (previous activities) has been completed, it has been investigated how to use these devices within this platform. For this, three-week work blocks were dedicated to each device to investigate its implementation in Unity 3D. As a result, for devices like Oculus Rift and Leap Motion, it was found that manufacturers directly supply a component for Unity 3D. For Stereoscopic screen and mouse 3D, the components developed by third parties were supplied for this project. For devices such as infrared cameras and joystick it was necessary to develop a driver for Unity 3D mainly for Windows OS. Given the technical difficulty of creating a driver for these devices and the short development time for this solution, it was decided to avoid implementing the use of infrared cameras. In the case of the joystick, it was changed by the gamepad, which is a similar device, and given the wide support it has for its use in Unity 3D, it was investigated its integration in this platform.

The main reasons for looking for components made for Unity 3D by manufacturers or by third parties, instead of developing them from scratch were:

- **Maintainability project requirement.** This project can be easily updated and easy to maintain, external components were sought. These components have a technical support for possible bugs.

- **The components have already been tested beforehand, so that before possible bugs, the manufacturer would have solved and subsequently tested the patches. Developing three controllers for these devices (infrared cameras, stereoscopic screen and joystick), would have taken too much time and would have exceeded the amount of time available to carry out this project, mainly because the time required to properly test and debug a software system.

These decisions have counterparts as loss of freedom in defining certain functionalities. The technical support can end unilaterally. But for this project these issues are minor, because the basic functionalities are covered.
4.2.3.2  Device limitations

Since the devices that were used for this project were not updated in regards to hardware, it is not possible to apply all software updates for their drivers, as these updates apply to a specific hardware version. This establishes a limitation in terms of the errors found in the versions in which they work. This means that errors of newer versions will not be fixed as they have been corrected in later versions that do not apply to the hardware version.

In the case of Oculus Rift, the version that has been worked is the DK1 and the maximum runtime to use is 0.6.0.0. For Leap Motion, the runtime to use should be: 2.2.6.29154 for Windows. For the 3D Mouse, since the controller is not supplied by the manufacturer but by a third party, depends on the issues reported by the users and the subsequent update by the author.

For teams or workgroups, it is recommended to standardize both the version of the device drivers and of the development platform (Unity 3D) in order to avoid possible failures or unexpected behaviors. This project initially was developed in 4.6 version and continued until version 5.1.1f1 of Unity 3D and at the time of writing this document is in version 5.5.0. Unity 3D is continuously releasing updates, so it is an important aspect to take into account at the moment to develop a project with this platform.

4.3  Solution design

In this section, we first explain how the UI has evolved given the refinement of the requirements and details the final design of the UI, represented by the three categories in which the devices to be used are classified: visualization, manipulation and navigation.

4.3.1  UI evolution

As the corresponding advances were presented and feedback was received, the requirements were refined, which is reflected in the changes of the UI to the final version. In the figure 18 below it was the first version, where all the options where placed on the screen.
After the review, it was concluded that it is not clear which options belong to which device, it is only grouped by main categories (visualization, manipulation and navigation). e.g. for Oculus Rift, the possible options are: Player Controller and Camera Rig. In the next figure 19, shows the changes made.

In this new UI version is clearer the options available for each device for every category. Also, it was added information to guide the user through the possible actions to do. The visual design became more compact, as can be seen in the following figures 20 and 21.
As displayed on Figure 21, for the Leap Motion device is clear to see for the user all the options available, from combining with Oculus Rift when the user check the option *Is Head Mounted*, to display on screen the tracking info generated by Leap Motion when the user check the option *Debug tracking info*.

After the corresponding review, it was found that this project assumed for the keyboard and the gamepad the keys and buttons used to move the camera and the player were defined beforehand, following the standard keys to navigate with keyboard: Up arrow, down arrow, left arrow and right arrow, and the keys: W, A, S, D. This assumption did not make possible to custom the keys and buttons.

In the new version, the changes introduced were: The user can custom the keys and buttons for control the camera and for move the player with the keyboard or gamepad. Set the value
between a minimum and maximum value for fields like: Rotation Speed, Max Angle for axis X, Y, Z. Zoom Factor, Run Factor and Slow Factor. This changes is displayed on Figure 22.

![Figure 22: Visualization Device layout – for keyboard device](image)

As displayed on Figure 22, these are the final layout for Visualization Devices, when the device for control the camera is the keyboard. For Oculus Rift device, the camera is controlled by viewer (hardware).

![Figure 23: Manipulation Device layout](image)

As displayed on Figure 23, these are final layout for Manipulation Devices. For mouse device when the user wants to move an object, it just need to drop and load the object to the field Add GameObject to move. For Leap Motion device, the user should select the hand model that will be rendered in 3D environment.
Figure 24: Navigation Device layout

As displayed on Figure 24, these are the final layout for Navigation Devices. For keyboard and Gamepad, the layout is the same, it is just change the values for the options from keys to buttons. For Mouse 3D, the figure display the layout created and the component imported (Hogenboom, 2014).

4.3.2 Final design

After refining the requirements, the visual design of the plugin was established according to the classification of the devices to be used (visualization, manipulation and navigation). As for the overall design of the solution, it was established that this plugin has to be as customizable as possible with respect to the keys or buttons that the user may want to use in order to perform the basic actions of an avatar and of the cameras within the 3D environment.

4.3.2.1 Visualization devices

For conventional displays, it was not needed to perform any extra configuration, as it only depends on the graphics card and on the operating system where the application is going to be run. The settings for the screen dimensions are set on Unity 3D by the designer. Additionally, when the application is built and is launched, Unity 3D displays a window where the user can select the screen resolution. See Figure 25.
Figure 25: Resolution dialog, presented to end-users

As displayed on Figure 25, the user can select the best option according with the computer hardware before launch the application.

For Oculus Rift, on the contrary, we had to consider several issues. First, it is necessary to install a runtime environment so that the device works normally. Second, the Unity 3D packages for the device version (DK1) had to be installed. These packages can be downloaded from Oculus developer center (Oculus, 2015). For Stereoscopic Screen, the Lab provided the FOV2GO package, as it allows managing the 3D stereoscopy of a virtual environment. It was required to search which component works better for Stereo Wall and to integrate it with the rest of project. The only drawback is that the scripts are developed in JavaScript and the project requires C# scripts. These made impossible to integrate them in runtime mode, and then it was necessary to integrate it into design mode.
4.3.2.2 Manipulation devices

For Mouse, a script was developed to allow the designer of 3D environment adding the elements that can be manipulated in runtime can. Also, in design time the designer can add the elements previously. For Leap Motion, it is necessary to install the OS driver and the Unity 3D package. This bundle includes an integration with Oculus Rift. To download this package is necessary to register into Leap Motion developer web (Leap Motion, 2015). The approach has taken the company has been continuously providing updates for the device on the Unity 3D platform.

4.3.2.3 Navigation devices

The keyboard is the most basic device to navigate through a 3D virtual environment. Based on the functionalities described in the section 4.2.1 (basic features), the developer can set the keys for 3 axis displacements according to his convenience. For the gamepad, the selected device was the one used in the Microsoft Xbox, because it has a greater support for Unity 3D than other gamepads. To integrate it in this project, a XInputDotNet DirectX API was used to manage the connected controller (Syed, 2015). It was necessary to adjust the input project settings to use their methods. These settings are described in section 4.3.3.4 How were incorporated third-party plugins. For the 3D Mouse, we found a free component for Unity 3D which allows managing the device and offers a UI for Unity 3D. For this reason, it was not necessary to create a new UI, but just to display it when the gamepad is selected. To integrate it into this project is necessary to install the drivers provided, install the package for Unity and change the input project settings.

4.3.3 Plugin Design

To design a user interface for Unity 3D for design time, the selected option was to develop a custom editor (Unity Technologies, 2015b). According to requirements the devices are displayed in three categories displayed in the following figures below.

Figure 26: Visualization devices list
As displayed on Figure 26, the user can select on visual devices the three devices. Allowing the user to easily switch device.

![Figure 27: Manipulation devices list](image)

As displayed on Figure 27, these are manipulation devices that user can select. Depending on the user selection, the available options will be displayed. Making the options to choose less confusing.

![Figure 28: Navigation devices list](image)

As displayed on Figure 28, this is the navigation devices list available to choose for the user. According user selection, between keyboard and gamepad, the sections: translation and rotation changes.
Figure 29: Complete view of UI
The Figure 29 is a complete view of the plugin. It is possible to differentiate the main categories (visualization, manipulation and navigation), as well as, all their options available according to the device selected.

4.3.3.1 Visualization Device

The designer can select which device to use. According to the device the configuration changes.

Oculus Rift

For this device, the manufacturer provided a component to use in Unity 3D. In this project were made changes to adjust the navigation with a keyboard or a gamepad. The main options to visualize a 3D virtual environment with Oculus Rift are: Player Controller and Camera Rig (see Figure 30 and Figure 31). For these options, it is not possible to control the camera with other devices, because it is controlled by the hardware. The camera moves as the viewer moves.

For Player Controller (see Figure 30,) the user can make the avatar/player travel within the environment by using the navigation device. To set the position on the 3D environment for the avatar/player, it should set the position to Manager Device, so that this device take the position.

![Manager Device (Script)](image)

*Figure 30: Oculus Rift settings for player controller*

For Camera Rig (see Figure 31), the avatar/player cannot travel by itself within environment, as it automatically follows the object it has been attached to. Then, the designer has to attach Oculus to another object with displacements. The object selected should set in “Object to Follow” field.
Stereo Wall

For this device, the project used the FOV2GO component in order to generate a three-dimensional and stereoscopic environment. As the component was developed in JavaScript and the project used C#, it was not possible to pass arguments at runtime because Unity 3D does not provide this option. The approach used was to allow the designer to add in design time the component to scene and to make the more suitable adjustments for the project, as displayed on Figures 32 and 33.

As displayed on Figure 32, the selection is *Stereo Wall* device. This figure displays an information message for the user to apply the changes and the component will display on 3D environment.
As displayed on Figure 33, when the component is placed on the 3D environment, is possible to the user select different options in design mode.

Point of View
In this section the user can select between first person or third person. On first person, it is a view which the world is seen from the perspective of the player. On third person, the player being controlled is seen in full body and usually on his back For Oculus Rift the third person is not enabled, because this device is designed to provide an immersive visualization, and then it is suited for first-person visualization. For both options the user has to select the element or object as avatar/player, which the plugin (solution) will follow when the application is running. As displayed on Figure 34, the field: rotation speed (from 1 to 20), will apply the rotation factor for the camera when it moves, as well as the fields: Point of View and Object to Follow.

Figure 33: Stereo Wall settings

Figure 34: Point of view section
Camera Settings

Rotation and zoom are actions that have to be performed using navigation devices (keyboard or gamepad), as displayed on Figure 35 and 36. For each camera movements the user can set the key or button to perform the action and set max angle rotation. Likewise, for zoom actions, it is possible to set a key or button and set a zoom factor. If the user wants to return to the start position it is possible setting a key for this action.

It should be noted that, if these actions are performed with a gamepad, there are not enough buttons to map all them. In this case, then, it would be necessary to combine the gamepad with another button (see Figure 36). The user must to check the field Combine with Button and select the corresponding button.

![Camera Settings - Rotation](image)

*Figure 35: Camera settings with keyboard*
### Camera Settings - Rotation

**X Axis**
- Turn Up and Down: Right Stick Y
- Max Angle: 30
- checkbox Combine with button

**Y Axis**
- Turn Left and Right: Right Stick X
- Max Angle: 75
- checkbox Combine with button

**Z Axis**
- Twist: Left and Right: Right Stick X
- Max Angle: 30
- checkbox Combine with button

- Select the button: Left Stick

**Zoom**
- Zoom In: Right Trigger
- Zoom Out: Left Trigger
- Zoom Factor: 4

- checkbox Reset Camera Rotation

---

**Figure 36:** Camera settings with gamepad

#### 4.3.3.2 Manipulation Device

The user can use the devices: Mouse, Leap Motion or None (see Figure 37). For mouse device, the designer can choose the objects by two ways. The first one, in design mode just need to drag and drop the objects into an array. The second one, in runtime can use the method: `AddGameObjectToMoveMouse`, which receive an object type: GameObject (see Figure 38).

---

**Figure 37:** Set the objects to move in design mode
As displayed on Figure 37, the user selects two objects type *Cubes* and one object type *Sphere* on design mode. When the application is running, these objects can be moved on the 3D environment.

```
void Start(){
  // Add gameobject to move
  GameObject go_prueba = GameObject.FindGameobjectWithTag("CanMove");
  if (go_prueba != null) {
    AddGameObjectToMoveMouse(go_prueba);
  }
}
```

*Figure 38: Script to add object to move in runtime*

As displayed on Figure 38, the user can select different objects by scripting using the method *AddGameObjectToMoveMouse*.

### 4.3.3.3 Navigation Device

The basic device for navigation is keyboard. According to requirements the user can set the keys for all the movements in the environment. Similarly, for actions like running or walking slowly, the user can assign a value from 2 to 20 for running and for walking slowly from -0.75 to 1, as displayed on Figure 39.

*Figure 39: Keyboard Settings for navigation*
As displayed on Figure 39, the plugin display all the options for keyboard device. Also, display information about avatar rotation and reset the position of the avatar. To perform the avatar/player rotations, the user can set a key for each orientation, which is used in combination with the translations keys (move forward, backward, left, right, etc.).

For gamepad device, the settings are similar to keyboard (see Figure 40). The designer must set the buttons to performs these actions. For rotations, the gamepad buttons are not enough to accomplish it. So, it is necessary to combine it with another button.

![Gamepad Settings for navigation](image)

*Figure 40: Gamepad Settings for navigation*
Mouse 3D

For this device, the project uses the component for Unity 3D: *SpaceNavigator driver for Unity 3D*. As displayed on Figure 41, the UI design uses an Editor Window, a custom editor can float or be docked as a tab (Unity Technologies, 2015e), on the Unity 3D editor. In this Editor Window the basic features are set and is complemented by other features on project settings as displayed on Figure 42. The basic feature window display automatically when Mouse 3D device is selected. Similarly, this windows is closed when is selected another navigation device.

![SpaceNavigator basic settings](image)

*Figure 41: SpaceNavigator basic settings*

![Mouse 3D Additional Settings](image)

*Figure 42: Mouse 3D Additional Settings*
4.3.3.4 How Were Incorporated Third-party Plugins

Unity 3D has a way to incorporate ready-to-use components. Combining basic elements of this platform with scripts and external libraries (e.g. dll’s), these components are called in Unity: Prefabs. This acts as a template which it is possible to create new instances in the project. Through this medium is possible to reuse these components in other projects.

For devices like Oculus Rift, Leap Motion the manufactures provided prefabs to incorporate to the Unity 3D projects. In the case of Mouse 3D the prefab were developed by independent developer (Hogenboom, 2014). All these Prefabs are packed into a file type named Asset Package, allowing to distribute more easily all these components created and reuse in different projects. For the StereoWall device the project use one of several Prefabs which meets the needs of this display.

To use one of these prefabs first it was necessary to import a package. Usually this file are stored in the Asset Store (Unity Technologies, 2015a). Also, is possible to use other platforms to distribute them as GitHub and similar.

To import a package, it should select the option: Custom Package, as displayed on the Figure 43. Then the user must to select the components to import into the project and finally check the new components installed in the project window.

![Import a Custom Package](image)

*Figure 43: Import a Custom Package*
As displayed on Figure 44, after select the menu option Custom Package, the user must to select the components to be added to the Unity 3D project and press the Import button.
As displayed on Figure 45, after the import process the project window display all the component imported before.

For gamepad device, this project use the XboxCtrlrInput (Syed, 2015), which is a wrapper handles XBox 360 controller input. This component make simple to work with the controller input on different OS’s. To use this device first is necessary install the driver either for Windows or Mac OS, then import the package for Unity and finally adjust the settings for input devices.
To apply these settings, go to the menu option: *Window / XboxCtrlInput / Replace InputManager.asset*. As displayed on Figure 46, it is possible to verify the new inputs created to use the gamepad on the project settings view.
5 Updating the project components

According as the hardware and software evolve this project needs to update their components and adapt to the new devices as far as possible.

5.1 Updating the Project Components

Since this project has third-party components every so often comes new updates and before to install them, is necessary to read the release notes to check if the new updates support the hardware version and the Unity 3D version. These are the recommendations for the used devices in this project.

5.1.1 Oculus Rift

Because the hardware version is DK1, the runtime and Unity Package versions that works best are: 0.4.4-beta and 0.6.0.0-beta. It is not possible to use the newest versions because they don't support DK1. Unless a new version of Oculus is acquired where the process is easier. Go to Oculus developer website (Oculus, 2015) and select Rift Runtime to download the newest runtime.

5.1.2 Leap Motion

The OS driver is updated periodically. Before to update is recommended to read the release notes to check if supports the hardware version. The Unity 3D component that best works for this device is 2.2.6.29154 in Windows OS and for Unity 3D uses 2.3.0 version. To update this component is important to review the Unity 3D version recommended and the hardware version. At the time this document was written there is a new beta version of this hardware called Orion, which is oriented to VR (virtual reality).
5.1.3 Mouse 3D

For this device is necessary to visit the manufacturer website (3D Connexion, 2015) to check for new updates according the OS. For the Unity 3D component is recommended to follow the GitHub repository (Hogenboom, 2014) to check new updates.

5.1.4 StereoWall

To updates for this device is recommended visit the manufacturer’s website (Stereoskopix, 2012) and the Asset Store to check new updates. This is a paid component, it was brought by the Lab, so the new updates can be downloaded free.

5.1.5 Gamepad

This project uses XBox gamepad, so first is visit the Microsoft website to check new updates for the driver in Windows OS. for Mac OS the driver used was Tattie Bogle driver (Tattiebogle, 2013). Second, for the Unity 3D component is recommended to follow the GitHub repository (Syed, 2015), to check new updates. This plugin requires to set the input settings.

5.1.6 Unity 3D

The version of this platform used for this project is 5.0.1f. To use this project in other PC’s is recommended to install the same version (5.0.1f or later), to assure compatibility at the time of move the Unity 3D project to another PC’s.

In a general way, to update whatever component installed where this project (Manager Device) is installed, following this steps:

1. Create a new Unity 3D project.
2. Import the new version package (Assets - Import Package - Custom Package). Using this Unity packages is the way how to share and reuse component from one project to another (see Figure 47).
Then, select all the package components to install pressing Import button, as displayed on the figure 48.

3. Using the file explorer (finder for Mac), situate the projects folders (new Unity 3D project and the project where Manager Device is installed). Copy and replace all the directory structure from the new project to the destination folder.
As displayed on Figure 49, this is a manual mode to update all the new components, there is no other mode to update this resources.

4. Open the project where the Manager Device is installed to update the new files updated.

5.2 Minimum Requirements for Use

This solution has the following requirements according for the hardware used and which was developed.

- Unity 3D version 5.0.1f or later.
- Oculus Rift DK1 uses the runtime 0.4.4-beta for SDK and Unity Package.
- Leap Motion uses runtime 2.2.6.29154, and for Unity uses the package 2.3.0 version.
- Mouse 3D uses the driver 3.18 version for windows and 1.0.4 for Mac OS. For Unity 3D, the version used is 1.5.1.
5.3 Project Structure

To structure this project in term of folders, there is no an official guide how to setup the folders in a new project. As displayed on Figure 50, this solution has followed third-party recommendations (Talibzade, 2015; Zasadnyy, 2014). The project has the following structure:

- **Editor.** In this folder are stored the scripts for the UI used in editor.
- **Images.** To store the Lab logos.
- **Prefabs.** In this folder are stored all the prefabricated components to be used according to the devices.
- **Scenes.** In this folder are some example scenes of this project.
- **Scripts.** In this folder are located all the scripts used for this project.

After install all the third-party components the global structure for this project is as displayed in the figure 51:

---

**Figure 50:** Folder Structure for this project

**Figure 51:** Global structure of project folders
• **Editor.** This folder is for the gamepad device.
• **Gizmos.** This folder is for Leap Motion.
• **LeapMotion.** In this folder are the resources for this device.
• **LeapMotion+OVR.** In this folder, there are the necessary components to use Leap Motion with Oculus Rift.
• **ManagerDevice.** This folder is for this project.
• **OVR.** This folder is for Oculus Rift.
• **Plugins.** In this folder are located the libraries for Windows (dll) and Mac OS (lib) the devices like Oculus and Leap Motion.
• **SpaceNavigator.** This folder is for Mouse 3D.
• **XboxCtrlrInput.** This folder is for gamepad device.

To distribute this project is necessary to create a Unity package. With a single file this structure is grouped and it is possible to include it in a new project. As displayed in Figure 52, to export all these resources the option menu is: Assets ▶ Export Package. Select all the components and set the folder where the file will be exported.

![Export package window](image)

*Figure 52: Export package window*
6 Extensibility and Reuse

To add a new device for this project it should take into account the following aspects:

- Search in the Unity 3D Asset Store if there is a component ready to use. Also, search into the manufacturer website.
- Establish the feasibility to buy a component or to find a free component made by third parties.
- If there is no support for Unity 3D, consider to develop a plugin for this platform following their recommendations (Unity Technologies, 2015c). To do this is required to download the SDK the manufacturer provides for the OS selected.

Once is founded a component or develop a plugin, it should be added to this project, checking that no conflicts occur for filenames or folder names. To include the new device into this project the steps should be follow are:

1. Open file ManagerDevice.cs. This file is stored into ManagerDevice • Scripts folder.
2. Add the corresponding line to the new device name according to the categories: Visualization, Manipulation or Navigation into the Enums: VisualizationDevices, ManipulationDevices or NavigationDevices, as displayed in Figure 53.

```csharp
// Visualization options
public enum VisualizationDevices{
    Display,
    OculusRift,
    StereoWall,
    NewVisualDevice
}

// Manipulation Options
public enum ManipulationDevices{
    None,
    Mouse,
    LeapMotion,
    NewManipulationDevice
}

// Navigation Options
public enum NavigationDevices{
    Keyboard,
    Mouse3D,
    Gamepad,
    NewNavigationDevice,
    None
}
```

*Figure 53: ManagerDevice.cs script extract*
3. To customize the options for the new device like drop down list, text field, checkbox and so on, it is necessary to edit the file ManagerDeviceEditor.cs located on ManagerDevice ▶ Editor. This file is the UI of this project, so any changes can affect the whole interface.

4. To set the options from the UI to the component it should open the ManagerDevice.cs file and edit the functions SetUpVisualOptions(), SetUpManipulationOptions() or SetUpNavigationOptions(), according to the device category.

This file is divided into three sections (Visualization, Manipulation and Navigation), so according to the type of device edit this script, as displayed in Figures 54, 55 and 56.

```csharp
// Visualization
ControllerDevice.VarVisualDevices = (ManagerDevice.Vis...

// Options for oculus rift
if (ControllerDevice.VarVisualDevices == ManagerDevice.
    EditorUILayout.BeginVertical("box");
    ControllerDevice.Ovr_PlayerController = EditorUILayout;
    ControllerDevice.Ovr_CameraRig = EditorUILayout;
    if (ControllerDevice.Ovr_PlayerController){
        ControllerDevice.Ovr_CameraRig = false;
    }
    EditorUILayout.EndVertical();
}

// Options for stereo wall
if (ControllerDevice.VarVisualDevices == ManagerDevice.
    // Because the StereoWall Script controller is in J
    // to setup on runtime. It is necessary setup on de
    // Button for apply the changes in design
    EditorUILayout.BeginHorizontal();
    if (GUILayout.Button("Apply Changes", GUILayout.He
        // Verificar si existe otro objeto con el mismo
        if (GameObject.Find("StereoWall_Camera"){
            Debug.LogWarning("Manager Device - StereoWa
```

Figure 54: Visualization section
Set log messages. It is recommended for display messages into the user Console, use the method: Debug.LogError (Unity Technologies, 2015d).
7 Evaluation

This section describes the evaluation of this project faced a new project from scratch and faced one finished project, with the purpose of establish how the adaption of this project in these two scenarios is.

7.1 Internal Test

To test this project from scratch, begins with a test scenario (see Figure 57). This solution was installed via Unity Package and the 3D environment used was Ultimate Fantasy Creator LITE (Johnson, 2015). Once installed it is proceeded to adjust the input settings for gamepad device and finally, test the different devices inside the 3D environment.

![Figure 57: Internal test with Ultimate Fantasy Creator 3D environment.](image)

For a project from scratch it should be noted the requirements and how this project can help to accomplish them. In this case is possible to provide a foundation as the project develops and it is possible to make the necessary adjustments in the development stage (Troya Moreno, 2016a, 2016b). For this case the results are:
• Visualization devices. For devices: standard display, Oculus Rift and Stereo Wall, the solution works satisfactory.
• Manipulation devices. For mouse device, there were some errors because of how the basic elements were built. For device Leap Motion the solution works satisfactory.
• Navigation devices. The keyboard there were no problems to displacement. It works satisfactory.
• Project Settings. To support gamepad, it was necessary adjust the input settings to map the gamepad buttons and joysticks. Using the 3D environment, it was necessary adjust the settings for the player / avatar to displacement.

7.2 Real Test Evaluation

The master thesis used to test was: “Interacción combinada táctil 3D para manejo de colecciones de documentos”. This project presents a prototype that enables communication and coordination between two applications, one showing documents represented by spheres in an application with touch interaction developed in Unity 3D and deployed in Android, and a second application also developed in Unity 3D which interact through gestures made in midair (Pazmiño Torres, 2016).

Once obtained a copy of source code, the next step was to install this solution via Unity Package and apply the settings to use the gamepad (see Figure 58), the solution was installed and it was ready to work.
In this project the player is located on a fixed position and manipulates the content displayed on StereoWall using the infrared cameras and mobile device. During the test was not possible to receive technical support from the author to understand the design and to know how to make the respective changes in the code for the solution to work. As a final result, the test was not satisfactory. This solution could not be implemented for the test application. If the technical support is not available from the author or there is no person in charge of software maintenance, the implementation of this solution can be difficult and long in other projects developed.

In the event that this solution works, the use of this test application with another device it is just could possible to apply only with visualization devices (display, oculus rift and StereoWall), because this solution doesn’t support infrared cameras and mobile devices, so that, this application cannot be changed to another computer.
8 Conclusions

Applying the development process in an iterative and incremental manner has allowed the design of the solution to be more satisfactory for the user, because the requirements were refined during each iteration, which allowed it to satisfy completely the needs of the Laboratory.

Given the general goals (section 4.2), and after completing this project, it can be established that this project is an initial contribution to creating a baseline so that the laboratory groups can use the most used devices that the laboratory has in a transparent way, and customize them according to the needs of the project as much as possible.

In order to take full advantage of this developed solution, it is important to have a basic knowledge of Unity 3D, because this plugin followed the usage guides to implement plugins for this platform, so that the design of the UI is not intrusive, despite having many customization options. And as the newcomers becomes familiar with Unity 3D, they will find the design of this plugin appropriate for their use. It is worth mentioning that improper use of Unity 3D can lead to misuse of this plugin.

The time savings for Lab newcomers when using this plugin will depend on the degree of familiarity with Unity 3D, because this is the main development tool. The level of customization offered by this plugin is very complete as it allows the user to select the buttons or keys on the keyboard or gamepad, to execute the different actions within the 3D environment to develop. It should be noted that for the player / avatar navigation system with the display and the gamepad, a different system was designed than Unity 3D normally offers. The new system allows the independent management of camera movement and player/avatar movement while navigating the 3D environment, making it resemble what is happening in the real world, when a person moves around an environment while watching what is around him when he turns his head.

The use of components made by third parties that were incorporated into this project has saved some time in development, since minor adaptations were made to these components so that they can work together with the project. It was also possible to satisfy the requirements given by the laboratory, which were refined during the iterations made. As a
counterpart, the free technical support provided by the authors could stop at any time by different issues. But they are external factors to this project that cannot be controlled. A positive aspect to the use of free components made by third parties is that they enjoy the support of the community of developers who keep these projects active and continue to make improvements and updates, as development tools such as Unity 3D and the hardware used today evolves over time.

8.1 Future Works

Because this project is an initial contribution to assist the Laboratory groups in the projects they undertake, this project has been designed so that new devices can be added. It is possible to include devices such as infrared cameras or Kinect, following the steps described in section 6.

When it comes time to upgrade hardware such as Oculus Rift, Leap Motion and 3D Mouse, an upgrade process should be performed in Section 5.

For Stereoscopic Screen (StereoWall), the component used for this project was written in JavaScript, and did not allow a complete integration of the device (section 4.3.2.1), at runtime. So, it would be interesting to re-write the code in C-Sharp so that this device can be used transparently for the user. With this change, it will be necessary to modify the UI to use a Stereoscopic Screen.
9 References


Troya Moreno, J. (2016a). Install Manager Device in Unity 3D. Retrieved from https://youtu.be/P7q0yK_sw1g


