LAMK
Lahden ammattikorkeakoulu
Lahti University of Applied Sciences

DEVELOPING A VIRTUAL REALITY APPLICATION IN UNITY

Case: SuperApp VR

LAHTI UNIVERSITY OF APPLIED SCIENCES
Information and Communications Technology
Media technology
Spring 2019
Simo Ahola

Tiivistelmä

Tekijä(t) Ahola, Simo	Julkaisun laji Opinnäytetyö, AMK	Valmistumisaika Kevät 2019
	Sivumäärä 29	
Työn nimi **Virtuaalitodellisuusapplikaation kehittäminen Unityllä** CASE: SuperApp VR		
Tutkinto Tieto- ja viestintätekniikan insinööri (AMK)		

Tiivistelmä

Opinnäytetyön tavoitteena oli luoda virtuaalitodellisuuspeli, jolla esitellään virtuaalitodellisuuden mahdollisuuksia. Peli on kehitetty käyttäen Unity pelimoottoria ja C# ohjelmointikieltä. Kehitetty peli tehtiin SuperApp-nimiselle yritykselle joka pääosin kehittää älypuhelinsovelluksia, mutta suunnittelee myös aloittavansa virtuaalitodellisuussovellusten kehittämisen.

Opinnäytetyössä käydään läpi virtuaalitodellisuuden peruskonsepteja sekä yleisiä asioita pelimoottereista sekä pelinkehityksestä. Opinnäytetyö myös vertailee millaisia eroja virtuaalitodellisuuspelien ja perinteisten videopelien välillä on kehittäjän näkökulmasta.

Työn tuloksena on virtuaalitodellisuuspeli, jota voidaan jatkossa kehittää pidemmälle lisäämällä siihen uusia ominaisuuksia, joilla voidaan esitellä virtuaalitodellisuutta. Peli tukee työpöytä- sekä mobiilivirtuaalitodellisuus alustoja.

Asiasanat

virtuaalitodellisuus, Unity, pelinkehitys, pelimoottori

Abstract

Author(s) Ahola, Simo	Type of publication Bachelor’s thesis	Published Spring 2019
	Number of pages 29	
Title of publication **DEVELOPING A VIRTUAL REALITY APPLICATION IN UNITY** Case: SuperApp VR		
Name of Degree Bachelor of Engineering, Information and Communications Technology		

Abstract

The objective of this thesis was to develop a virtual reality game that is used to demonstrate the capabilities of virtual reality. The game was developed in the Unity game engine and with the C# programming language. The developed game was made for a company called SuperApp, which mainly develops applications for smartphones, but is looking into virtual reality development.

This thesis covers general information about virtual reality games. It deals with what virtual reality is and how virtual reality games differ from traditional games. It also includes general information about game engines and game development.

The thesis work resulted in a virtual reality game that can be expanded upon by adding more features that help to demonstrate virtual reality. The game also supports both desktop and mobile virtual reality platforms.

Keywords

virtual reality, game development, Unity, game engine

TABLE OF CONTENTS

1 INTRODUCTION

A science fiction book called the Pygmalion's Spectacles written in 1935 by Stanley Weinbaum describes a pair of goggles that allow the wearer to experience a fictional world through stereoscopic images, smell and touch. The experience described in the book closely resembles the experience of the modern-day virtual reality (VR) headsets. It even describes that the main character feels a disconnect as he feels the chair that he is sitting on even though it does not exist in the virtual world that is projected to his eyes. (Weinbaum 1935.) Virtual reality has become from science to reality. Today virtual reality devices are available for the users to buy and for the developers to develop games.

The objective of this thesis is to develop a virtual reality game that is used to demonstrate the capabilities of virtual reality. The game will be developed in Unity game engine and C# programming language, but with some changes, the basic concepts covered in this thesis can be used with other software as well.

The first part of the thesis covers general information that is needed to develop virtual reality games. It deals with what virtual reality is and how virtual reality games differ from traditional games. It also includes general information about game engines and how game development is done in the Unity game engine.

The practical part of the thesis will consist of the development of a demo application made for a software company SuperApp to demonstrate the capabilities of VR for potential customers. The demo application will be developed for two virtual reality platforms, Oculus GO and HTC Vive, which differ in the way they work. These platforms have some different features that had to be taken into consideration and are explained in the thesis.

The game developed in the thesis will have these basic functions:

- Movement in a 3D environment
- Interaction with 3D objects and game engine physics using motion tracked controllers
- Different virtual reality user interfaces

2 VIRTUAL REALITY

Virtual reality (VR) is a term coined in 1987 by Jaron Lanier (Virtual Reality Society 2019a). The term is used to describe an experience where the user is placed in a computer-generated environment. The Sword of Damocles, which is thought to be the first head-mounted virtual reality device, was developed in the 1960s, and it featured rudimentary wireframe graphics. Even though the device was fairly primitive and so heavy that the user could not support it, it was still an incredible invention for its time. (simpublica staff, 2014.)

In the early '90s, many companies such as SEGA, Atari, and Nintendo were developing their own commercial VR headsets. None of these devices were released or gained any real measure of financial success due to the low performance of the computers at the time. (Virtual Reality Society 2019b.)

Today the computing power of devices is much better, and there are multiple commercial headsets on the market. Also, technology is getting more attainable for the average consumer. These are the reasons why it has never been a better time to get into the development of VR applications.

2.1 Virtual Reality Headset

Virtual reality headset is a device that is mounted on the user’s head. The device provides virtual reality for the user by using a screen and optics to provide a separate image for each eye, which creates a stereoscopic 3D image. VR headsets are also equipped with motion tracking hardware such as gyroscopes and accelerators. With these trackers, the image displayed by the headset can be matched to the user’s head movements, creating an immersive experience. (Rouse 2016.)

There are two types of motion tracking in virtual reality headsets. The first is called three degrees of freedom (3DoF). It means that the headset only tracks its rotation along three axes (x, y, and z). Because of this, if the user moves their head forward, the device will not recognize the movement. (Weis 2018.)

For tracking the position of the headset, six degrees of freedom (6DoF) are needed. The extra degrees are the position of the device on the three axes. 6DoF tracking can be achieved by using two external trackers that are positioned around a defined play area like in Figure 1. By using these trackers, the device can calculate the position of the tracked object. This type of 6DoF tracking is called outside-in tracking. A different kind of tracking called inside-out tracking uses cameras on the headset that track points around its environment like in Figure 2. (Weis 2018.)

Outside-in tracking needs more setup work than the inside-out tracking because of the trackers that need to be set up around the room, but because the inside-out tracking utilizes cameras to track the position, the environment has to be lit better than with the outside-in tracking.

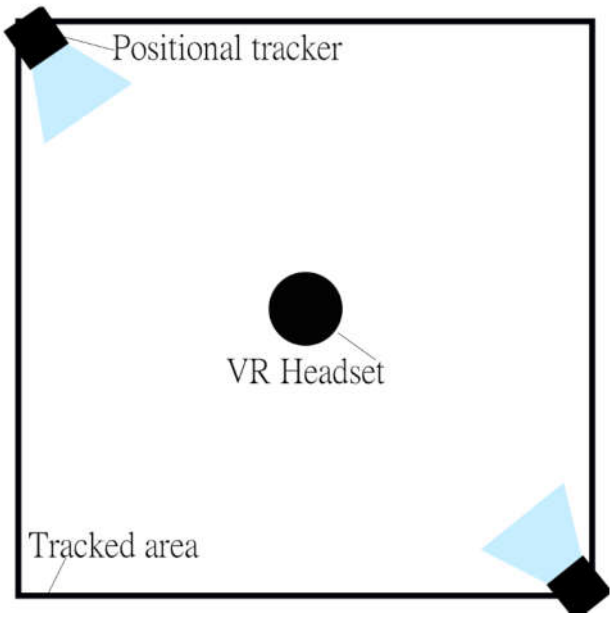


Figure 1 Representation of positional tracking of the HTC Vive

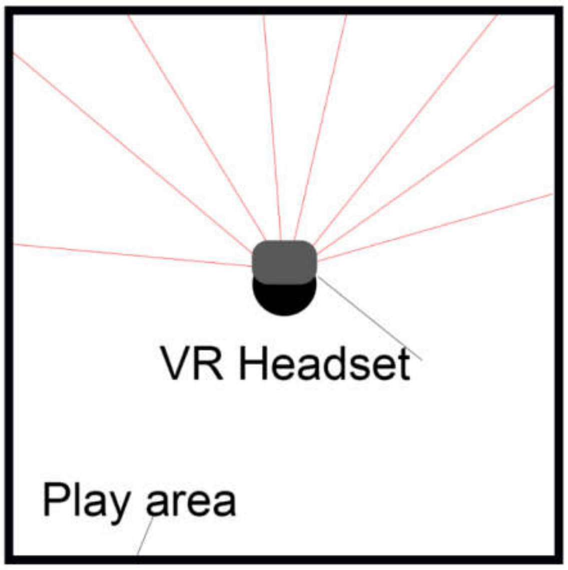


Figure 2 Example of inside-out tracking

VR headsets often come equipped with controllers that allow the user to interact with the virtual reality environment. The controllers are tracked in the same ways as the headsets with 3DoF or 6DoF technologies. 6Dof controllers allow for more natural interaction as the user can move them in the virtual space freely. The 3DoF controllers can only be rotated around their axes making the interaction very different than it is on the 6DoF controllers.

2.2 Movement in Virtual Reality

While 6DoF VR headsets provide positional tracking, it is confined by the space that is dedicated to the device. For example, if the device is set up in a smaller room than the virtual environment, the user will not be able to walk around the virtual space freely. Or if the device has only 3DoF, it does not support positional tracking at all. For these reasons, there need to be different movement systems in place.

One way to implement movement is to bind directional buttons or a joystick of the controller to the movement of the player. It is a simple solution that is usually supported straight in the game engine. The problem with this solution is that it may cause motion sickness in some users as the user perceives motion with their eyes but not with their bodies (MedlinePlus, 2019).

Another popular way to implement motion is the teleportation method that is used to alleviate motion sickness by "teleporting" the user around the virtual space. The teleportation is done by pointing a graphical indicator to the area where the user wants to teleport and pressing a button, and then the user is transferred to the end position of the indicator. A curve is often used to represent the indicator as it is very intuitive for the user. The easiest way to draw the curve indicator is to draw a bézier curve starting from the controller and ending at the point determined by the forward rotation of the controller. (Szauer, 2017a.)

2.3 Virtual Reality User Interfaces

Most of today's computing devices utilize a screen and input devices for user interaction. This is also true about virtual reality applications, but there is a crucial difference. The screen used in the virtual reality headset is mounted on the user's head so close to their face that it seems that there is no screen. Instead, the user is immersed in the graphics shown on the screen of the headset. Because of this, some of the traditional user interfaces do not work that well in virtual reality.

2.3.1 Non-Diegetic User Interfaces

In non-VR games, user interfaces (UI) are often overlaid on top of the game view like in Figure 3 to show relevant information such as player health or score. This type of UI does not exist in the game world but makes sense for the player. (Unity Technologies 2019a.)

This is called non-diegetic UI. The term comes from the film industry where it used to describe a certain type of sound. Sounds that do not originate from the world of the film, for example background music or the narrator's voice, are called non-diegetic sounds. (Dykhoff, Klas 2012.)

Non-diegetic UI does not usually work in VR as our eyes are unable to focus on something so close. (Unity Technologies 2019b.) For this reason, alternative UI methods must be used.

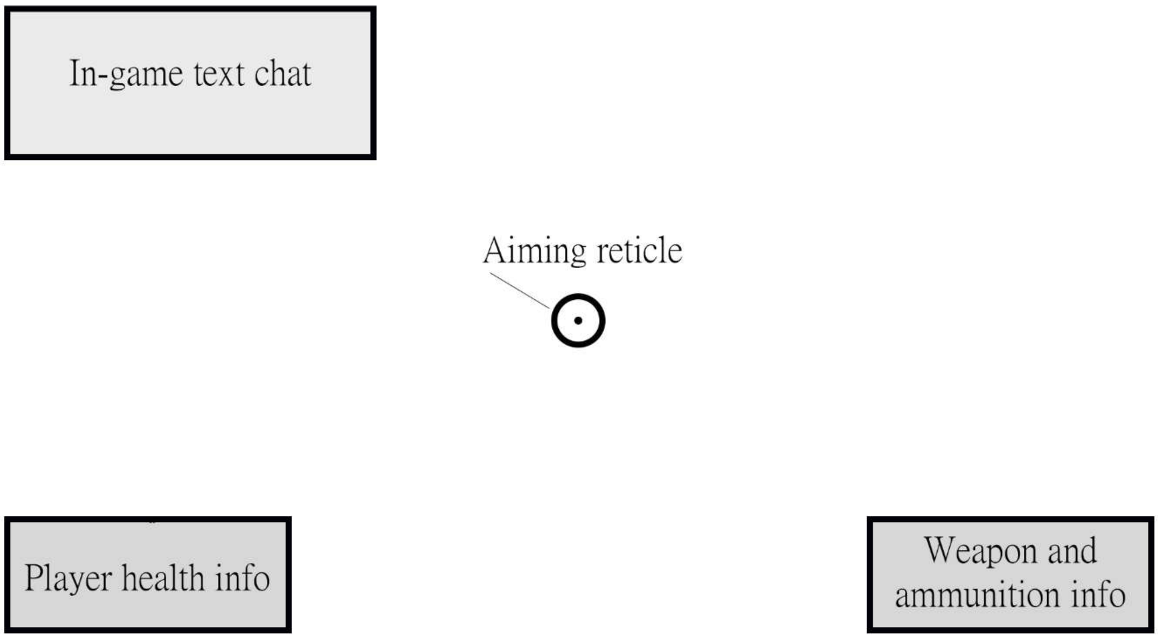


Figure 3 Representation of non-diegetic UI

2.3.2 Spatial and Diegetic User Interfaces

Spatial UI is positioned in the game environment itself as a 2D plane. This makes it easier for the player to focus on the UI. The placement of the UI is also important as it needs to be at a comfortable reading distance from the user. This will help to reduce eyestrain from the player. Attaching the spatial UI to the camera and making the UI follow it may cause nausea in some users. (Unity Technologies 2019c.)

Diegetic UI is like spatial UI as it is also positioned in the game environment. However, there is a key difference in that the diegetic UI relies on elements of the game environment to display information. For example, the clock in Figure 4 shows the player the time. (Unity Technologies 2019d.)

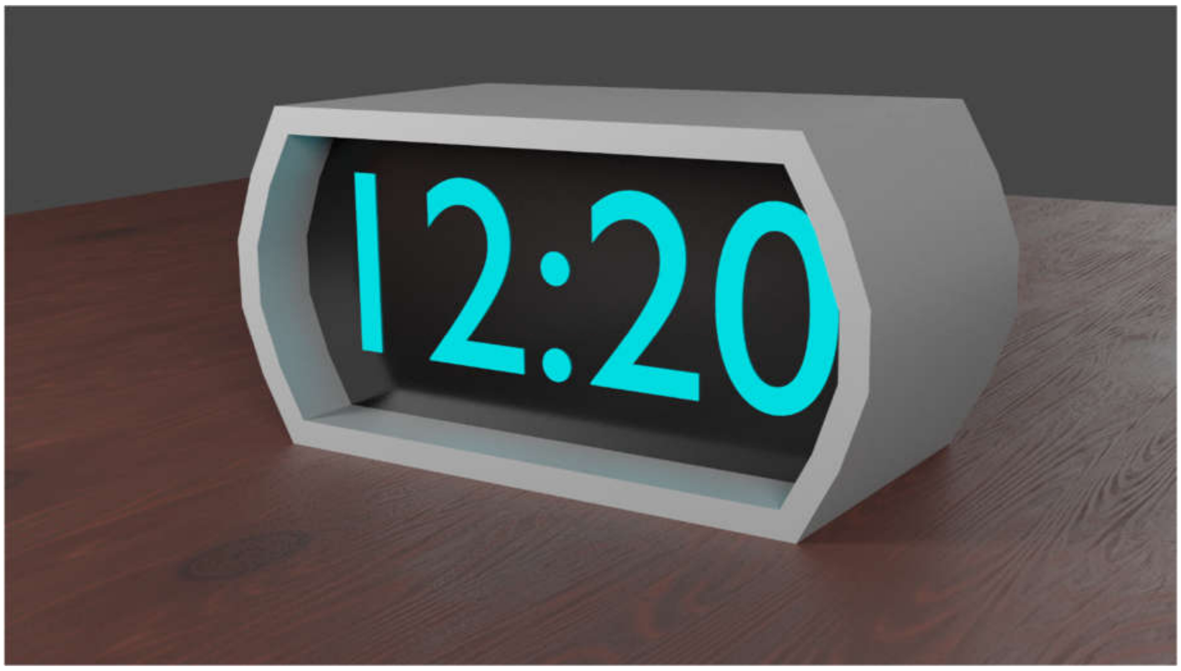


Figure 4 Diegetic clock UI

2.3.3 Interaction with UI

Interaction with UI in virtual reality can be handled in multiple ways. Most virtual reality controllers have some directional buttons, a directional pad or a joystick; these can be used to control the UI in the same way as a television remote. Another way is to make the controller act as a laser pointer that can be pointed at a UI element to select it. A third way is to use a 3D object as a button that can be "pushed" with the motion controller utilizing its 6DoF tracking and physics colliders in the controller game objects.

Diegetic UI also provides interesting opportunities for UI interaction. Because diegetic UI can essentially be any object, levers, volume knobs, switches or anything like these can be used to interact with the UI.

3 GAME ENGINE STRUCTURE

3.1 Game Engine

A game engine is a collection of modular code that does not directly handle the behavior of the game but handles most of the "background" procedures like input, output, and generic physics. To explain it further, a game engine reads data from different types of files and transforms it into a form that can be displayed on the screen. For example, it can take texture data from an image file and transform it into a material asset that the game engine understands and can render to the screen. Game engines were initially developed by game studios to keep the cost of developing new games down by reusing code from older similar games. (Lewis & Jacobson 2002.)

Today, multiple third-party game engines are used by game studios and indie developers that do not have the resources or the knowledge to create their own game engines. For example, Unity, Unreal and CryEngine are popular third-party game engines. These third-party game engines usually have licensing models that generate revenue for the companies that make them. For example, the Unreal Engine is free to use, but it charges 5% royalty for the developed games (Unreal Engine 2019).

3.2 Graphics Engine

Video games are real-time applications, meaning that if the user gives input commands, the game responds almost immediately. This requires that the game updates its view fast enough to create an illusion of moving images. A graphics card is a component that is designed to execute calculations that are needed to draw the image as fast as possible. So to achieve real-time graphics, the game engine must communicate with the graphics card. The graphics engine conducts this communication using graphics application programming interfaces (API). (Chernikov 2019.)

APIs are sets of tools that give commands to supported graphics cards. This means that the graphics engine is not programmed to run on a specific GPU, making it easier to develop games for multiple different devices. (Samsung 2019.) Unity supports various graphics APIs like Vulkan, iOS Metal, DirectX12, NVidia VRWorks or AMD LiquidVR (Unity Technologies 2019e).

There are standalone graphics engine programs that are designed only to render graphics. These programs can be extended to create a full game engine. An example of a standalone rendering engine is the open-source project Ogre3D. It features support for several graphics programming APIs and basic features to display graphics like material, mesh and animation support. (Ogre3D 2019.)

3.3 Physics Engine

In order to feel natural, game objects must accelerate correctly and be affected by forces like gravity and collisions. This is handled by the physics engine component of the game engine. (Unity Technologies 2019f.)

A physics engine often works in two stages: the setup and the draw. The setup is a stage where the physics engine detects all the objects in the game world or scene that are meant to interact with physics. The draw stage happens every time the graphics engine draws a new frame. During this, the physics engine calculates where the game objects that are affected by physics should be positioned in the next frame. (Shiffman 2017.) For example, if a sphere is falling, the physics engine calculates its next position based on the mass property of the sphere and the global gravity. If the sphere hits the ground, the physics engine handles the collision making sure the sphere does not just fall through the ground.

Some game engines like Unity have two different physics engines, one for 3D games and the other for 2D games. (Unity Technologies 2019f.)

3.4 Audio Engine

When creating immersive games like virtual reality games, audio is an important part of the experience. Audio engines are made to handle sound-related problems such as file format compatibility, positional audio, and moving audio or the Doppler effect.

In Unity, audio sources and audio listeners can be attached to game objects. The audio listener simply listens to audio that is generated by audio sources. Its position determines how the sound is played through the speakers. In first-person games, the audio listener is often connected to the main camera, making the spatial sound correspond to the view of the player. The audio source component has multiple properties that control which audio clip is played, or how it is played. (Unity Technologies 2019g.)

4 TOOLS

4.1 Unity

Unity is a game engine released in 2005 that supports development for over 25 platforms, more than any other creation engine. It supports the development of 2D and 3D games. Unity’s primary scripting language is C#, which it supports natively.

Unity also has native support for multiple virtual reality and augmented reality devices such as Oculus Rift, Steam VR/Vive, Playstation VR, Gear VR, Microsoft HoloLens, and Google’s Daydream View. Other notable features of Unity are its two physics engines (3D and 2D), animation tools, shader graph (node-based shader programming) and asset store (Unity 2019).

4.2 Blender

Blender is an open source graphics program developed by the Blender Foundation. Blender's development started in 1994 by Ton Roosendaal who was the co-founder, art director and the lead in software development at an animation studio called NeoGeo. At first, it was developed to replace the company's old animation toolset, but Ton quickly realized that artists outside the company could use it. In 2003 the first truly open source version of Blender was released. (Blender 2019a.)

Blender has a wide range of tools for 3D modeling, rendering, animation, visual effects, and video editing. (Bender 2019a) Blender is licensed under the GNU General Public License (GPL) making it free to use and sell your work. (Blender 2019a.)

Blender 2.8, the latest version currently being developed, will bring many improvements to Blender’s user interface, completely overhauling it and making it more user-friendly. The 2.8 release of Blender also features a new real-time physically based rendering engine called EEVEE. Being a real-time engine, EEVEE can be used to preview materials while texture painting or sculpting. This will also make the animation workflow faster as rendering is not needed to preview how the animation looks with textures. (Blender 2019b.)

4.3 HTC Vive

HTC Vive is a desktop virtual reality device developed by HTC and Valve Corporation and released in 2016 (Niko Pino 2018). HTC Vive consists of a headset that has an LCD screen inside it and motion controllers for each hand. It offers a "room-scale" experience with its 6DoF outside-in motion tracking.

Being a desktop VR device, it needs a computer to run the software, as it does not have the hardware to do this itself. This allows it to run games that have higher fidelity graphics than a standalone device that is dependent on a mobile computing device. Because the desktop headset needs a powerful PC to process its graphics, the total price of the device is a lot higher than the price of a standalone VR device like the Oculus GO.

4.4 Oculus GO

Oculus GO is a standalone virtual reality device developed by Oculus VR, released in 2018 (VRFocus 2018). It features an Android-based computer with a 2560 x 1440 resolution display and a Qualcomm's Snapdragon 821 processor. The device does not track the position of the headset or the included controller, making it a 3DoF headset. (VR Bound, 2019.)

Before developing Oculus Go, Oculus worked together with Samsung to develop Samsung Gear VR. It was a VR device that used a separate Android phone as its computing device. Because the Samsung Gear VR and Oculus Go are so similar, all the software that is developed for one of these devices is compatible with the other (Android Central 2018).

4.5 Steam and SteamVR

Steam is a digital game and software store and a social media platform developed by Valve Corporation (Valve Corporation 2019.) It was initially released in 2003 for updating Valve's online shooter game called Counter-Strike. Before 2005 Steam was just a tool to update Valve's games, but it changed when they started signing external publishers to their platform. (Plunket 2013.) This led Steam to become the largest online pc game retailer (Edwards, 2013).

5 GAME DEVELOPMENT IN UNITY

5.1 The User Interface

The first step in learning any new software is to get familiar with its user interface. Unity's default UI shown in Figure 5 consists of four windows that serve their own purposes.

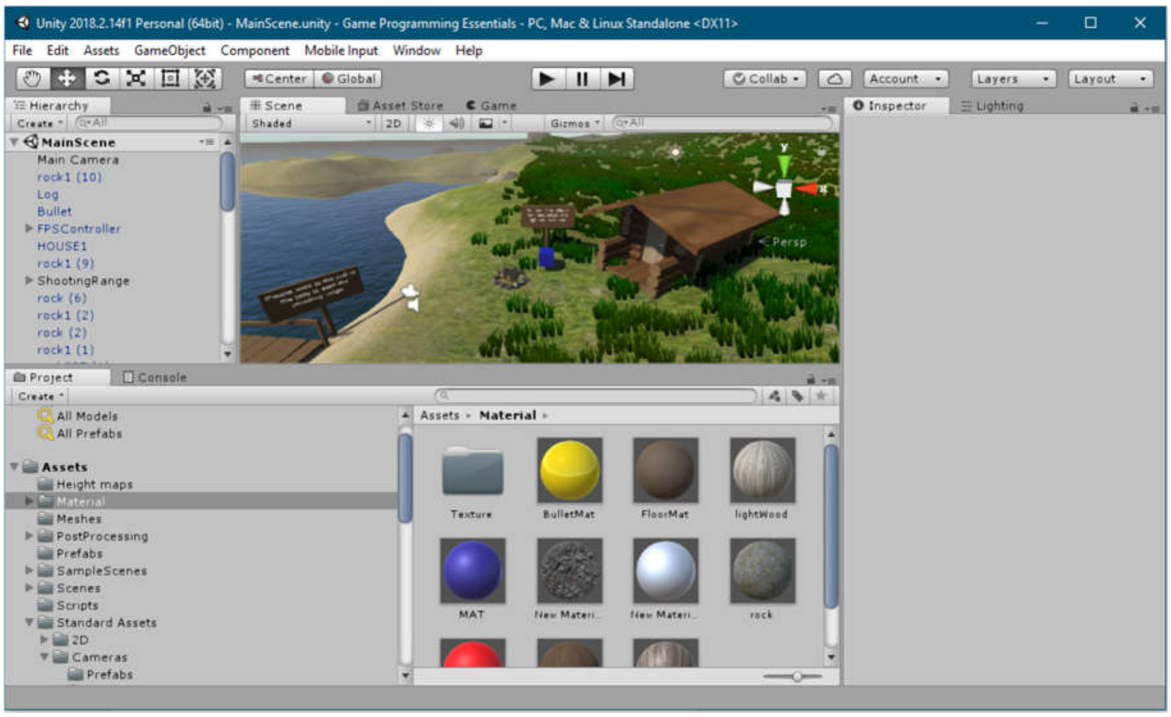


Figure 5 Unity's default UI

The hierarchy window is located on the left side of the UI. It contains all the game objects situated in the scene that is currently open. Using the hierarchy window, the developer can select an object to view or edit its properties in other windows. Like many other programs, Unity supports object parenting. This means grouping the "child objects" under a "parent object". Parenting makes managing a large number of objects easier by making the hierarchy window cleaner. Transform operations like moving and scaling applied to the parent object are also applied to its child objects. (Unity Technologies 2019g.)

The scene window is used to view and edit the environment of the opened scene. Selected objects are highlighted in orange and can be moved, scaled and rotated using the gizmo tool. The view mode of the scene window can be changed from the drop-down menu on the top left. These view modes can be used to view relevant data such as the 3D wireframes of the objects or baked lighting textures. This window also has a game view and the asset store as its tabs. The game view is enabled when testing the game that is being developed, and the asset store is used to browse and acquire assets from the Unity Asset Store. (Unity Technologies 2019h.)

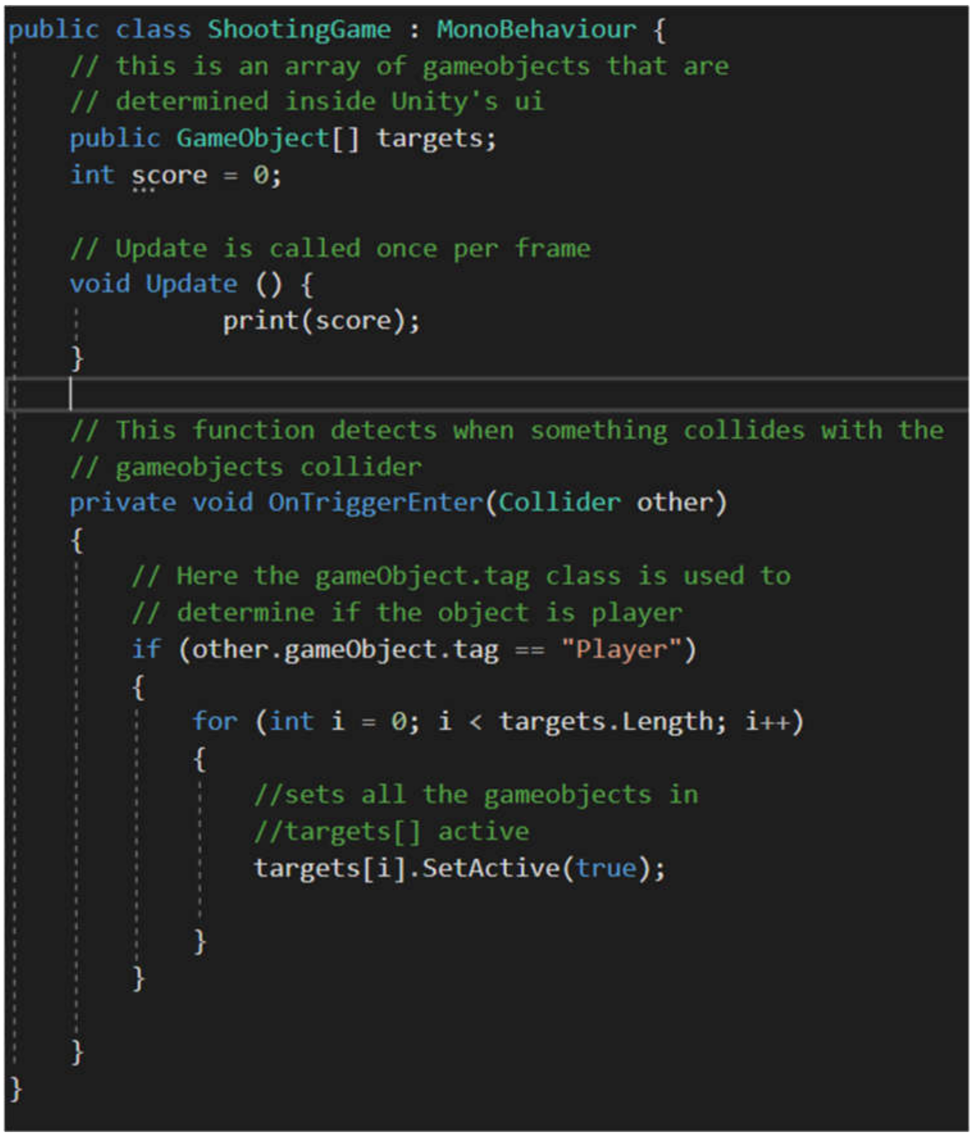
The inspector window is used to edit the properties of the selected items like GameObjects, materials, models or almost anything. Here the components of GameObjects can also be accessed and created. (Unity Technologies 2019i.)

The project window is Unity's file manager. It can be used to view the assets imported and created into the project folder. The left side shows the folder structure, and the right side shows all the assets inside the selected folder. It also has a favorites list and a search bar that can be used to search assets with keywords. Unity has more specialized windows that can be accessed under the window tab on the top menu. (Unity Technologies 2019j.)

5.2 Game Object

Every item in a Unity game is a game object including 3D assets, characters, and lights. A game object on its own has no functionality, so it needs to have components attached to it. Every game object contains the Transform component, which contains position, rotation and scale data of the object. Other components include scripts, audio sources, UI options and many more. Game objects can be created from the top left dropdown menu in the Hierarchy window. The menu contains many preset game objects that have some components added to them. All these objects can be created from an empty game object by adding components to it from the bottom of the Inspector window. (Unity Technologies 2019k.)

Game objects also contain properties called Tags and Layers. Tags are used to categorize objects into different groups, for easier manipulation with scripts. These are especially useful in collider control scripts. For example, a default tag “Player” is useful in detecting if a player enters a collider that triggers an event to start. Figure 6 demonstrates a function where the script will enable selected objects, if the player enters a trigger collider. Layers are mostly used by cameras to determine which game objects to render, but it can also be used to choose what objects are illuminated by a certain light or to create raycasters that ignore particular game objects.



```
public class ShootingGame : MonoBehaviour {
    // this is an array of gameobjects that are
    // determined inside Unity's ui
    public GameObject[] targets;
    int score = 0;

    // Update is called once per frame
    void Update () {
            print(score);
    }

    // This function detects when something collides with the
    // gameobjects collider
    private void OnTriggerEnter(Collider other)
    {
        // Here the gameObject.tag class is used to
        // determine if the object is player
        if (other.gameObject.tag == "Player")
        {
            for (int i = 0; i < targets.Length; i++)
            {
                //sets all the gameobjects in
                //targets[] active
                targets[i].SetActive(true);

            }
        }

    }
}
```

Figure 6 Example usage of the tag class

5.3 Character Controller

Character controller is a game object used to control a first-person or third-person character inside Unity. It contains all the necessary components that make up the character such as the mesh renderer, colliders, and movement scripts. Virtual Reality games are most often first-person games because they are designed to make the user feel like they are inside the game world.

To move a character in a game engine, the character must be scripted to react to user input from an input device. In case of a traditional game controller or keyboard and mouse, this is simpler than in the case of a Virtual Reality headset that has multiple positional trackers and gyroscopes. For this reason, companies that develop VR headsets, like Oculus or Valve, provide Software Development Kits (SDK) for popular game engines. These

SDKs contain scripts like character controllers that help developers to develop software for their devices.

5.4 Scripting

Unity supports the programming language C# natively. It can be used to write scripts that are used to make objects react to user input, trigger game events, modify properties of game objects and much more. New scripts can be created from the top menu or right-clicking in the project window and selecting **Create > C# Script**. Scripts are edited with an external editor, which by default is Visual Studio. This can, however, be changed in the settings. A script cannot function by itself and must be attached to a game object as a component. (Ivanov, G 2016.)

Until recently, Unity supported scripting with a Java-Script like language called Unity Script. Unity Script is currently being deprecated from the software because Unity is only developing new features for their C# Scripting Runtime, making Unity Script outdated.

5.5 Lighting

Like in the real world light is needed to see objects in a game. Lighting dramatically affects the look of a game and it is one of the most essential parts in creating realistic graphics. Figure 7 shows a scene with only the color information of the textures and the same scene fully lit side by side. Lighting in Unity can be divided into two categories: real-time lighting and precomputed lighting. Usually, these techniques are combined to make believable lighting setups.

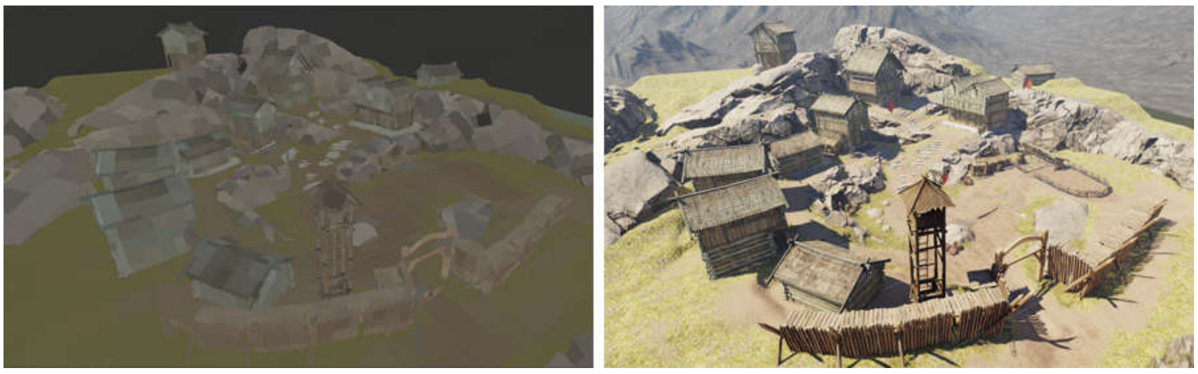


Figure 7 Lighting example (Unity 2019)

Real-time lighting is the most basic and default way to achieve lighting in Unity. It is mainly used to light movable objects such as characters or vehicles. Because real-time lighting is calculated when running the game, it can be very resource-intensive. It is also why real-time light rays do not produce any bounced light. (Unity Technologies 2019l.)

In order to create more realistic lighting, global illumination (GI) is needed. Global illumination is a term used to describe the light that is reflected or bounced off other objects. Calculating global illumination can be too slow to run in real time so a workaround is needed. Lightmap baking is a method that calculates global illumination data and stores it in textures called lightmaps. These calculations are done during the game's development. A downside to this method is that it can only be used for static objects because the lightmap textures cannot be updated in real time. This is why most games use a combination of both baked lighting and real-time lighting. (Unity Technologies 2019m.)

Lighting settings can be opened from the top menu: **Window > Rendering > Lighting Settings**. The menu has a range of options to control the environment lighting and GI settings. This window is also used for baking lightmaps for baked global illumination. Before baking, properties like the number of light bounces calculated for global illumination and the lightmap resolution must be defined. These settings affect the length of the baking process.

5.6 Building the Game

To test the game outside of the Unity editor or to publish the game it must be first built. The building process packs the scenes that are selected in the build menu to a format that can be executed by the target device. For example, if the target device is a Windows PC, Unity will create an .exe file and a data folder that contains all the assets used by the build. The building can be done from the build settings menu that can be opened from **Files > Build Settings**. Here the scenes to build, target platform and the relevant options like compression method are chosen. Then pressing the build button will ask for the directory where the built game will be compiled to and after that start the process. Depending on the target platform, external modules might be needed; for example Oculus GO is an Android device and needs Android SDK to be installed.

6 CASE: SUPERAPP VR

6.1 Use Case

The SuperApp VR application was developed to demonstrate the possibilities of virtual reality for clients that are interested in the technology. Most people have little to no experience with virtual reality so it is important that the clients can experience it. The experience may provide the client with ideas they would not have thought if they had not experienced it. Marketing of these applications usually happens during meetings that can take place in offices that are not the company's that is advertising the application. Then a mobile VR headset is a crucial tool in presenting the application because it does not require a complicated setup. Also, Mobile VR headsets will not take as much space when traveling as desktop VR headsets do.

6.2 3D Environment

The 3D environment of SuperApp VR was created by using modular environment assets that were created with the modeling program called Blender and then assembling these assets into a scene in Unity. These assets were created with mobile VR in mind; this means that the topology had to be as optimal as possible. In other words, the polygon count of the 3D objects is as low as possible while maintaining a distinct silhouette. The environment is a space ship that has multiple rooms built to demonstrate different VR features. Figure 8 shows an orthographic view from the top of the 3D environment.

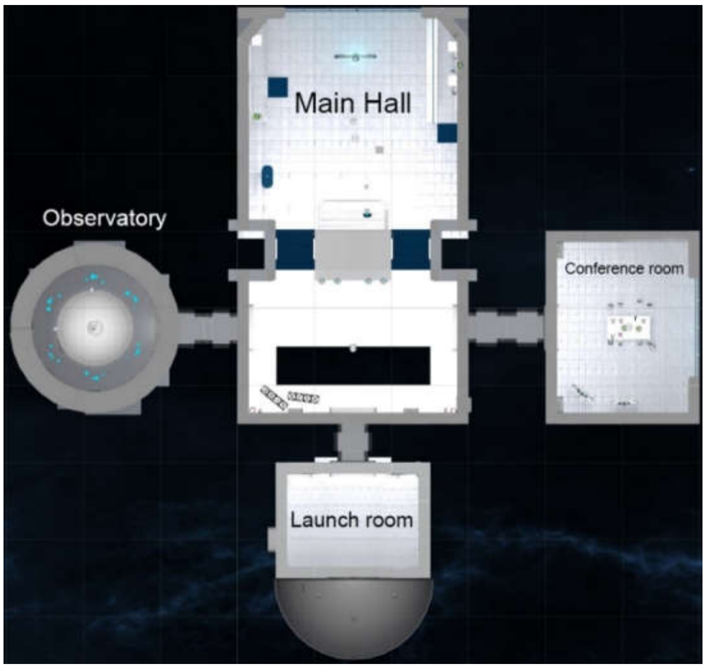


Figure 8 Map of the 3D environment

6.3 Movement

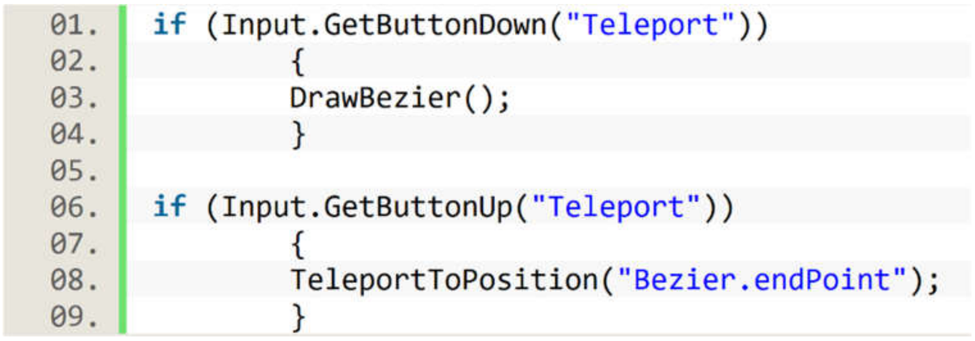
The project was created to support two different VR headsets, HTC Vive, and Oculus GO. SuperApp VR uses the character controllers that are provided by the makers of these headsets. The steps to import these vary depending on the headset.

Importing the character controller for the Oculus Go happens by first downloading the Oculus Integration package from the Unity Asset Store. Once the download is completed, the package needs to be imported to the Unity project. After importing the character controller, the prefab can be found in the directory **Oculus/VR/Prefabs/** and it is called **OVRCameraRig**. Dragging this prefab to the game scene will add basic VR functionality for Oculus Devices.

Before importing character controller for HTC Vive, Steam and SteamVR must be installed. Steam can be found from the steam website: https://store.steampowered.com/about/. After installing Steam, SteamVR can be installed from the Steam application. After these steps, importing the character controller is very similar to the Oculus method. First SteamVR Plugin is downloaded and imported from the Unity Asset Store and then dragging a character controller called **[CameraRig]** from **/SteamVR/Prefabs/** to the scene will add basic VR functionality for HTC Vive.

Character controllers are used for the tracking of the headsets and their controllers, but not for the movement of the player relative to the game environment. For that, the application uses a separate script. SuperAppVR uses a teleportation system to minimize nausea.

The script is attached as a component to the controller game object inside the character controller of the VR headset. When the script gets a specific button press from the controller, it draws a bézier curve from the end of the controller to a point on the game environment based on the direction and rotation of the controller. When the pressed button is released, the character controller is moved to the position that the bézier curve is pointing to. Figure 9 shows the logic in C#. The script is so general it can be used for both of the VR headsets.



```
if (Input.GetButtonDown("Teleport"))
        {
        DrawBezier();
        }

if (Input.GetButtonUp("Teleport"))
        {
        TeleportToPosition("Bezier.endPoint");
        }
```

Figure 9. Code for the teleportation

6.3.1 Bézier Curve

Bézier curve is used in many graphics programs to draw smooth curved lines. It is named after Peter Bézier, who wanted to create a tool to help design engineers to define curves and surface better than previously. (Mortenson 1999.)

The most straightforward bézier curve is the quadratic bézier curve. It can be constructed by defining three points: start point P1, control point P2, and end point P3. When these points are chained together in ascending order, two lines, L1 and L2, are created. When interpolating along these lines by some value t, two new points are created, Q1 and Q2. When connecting these lines with a line that is also interpolated by the value, t a point Q3 is created. As the value t approaches value 1, Q3 draws the bézier curve. (Szauer, 2017b.) This is illustrated graphically in Figure 10.

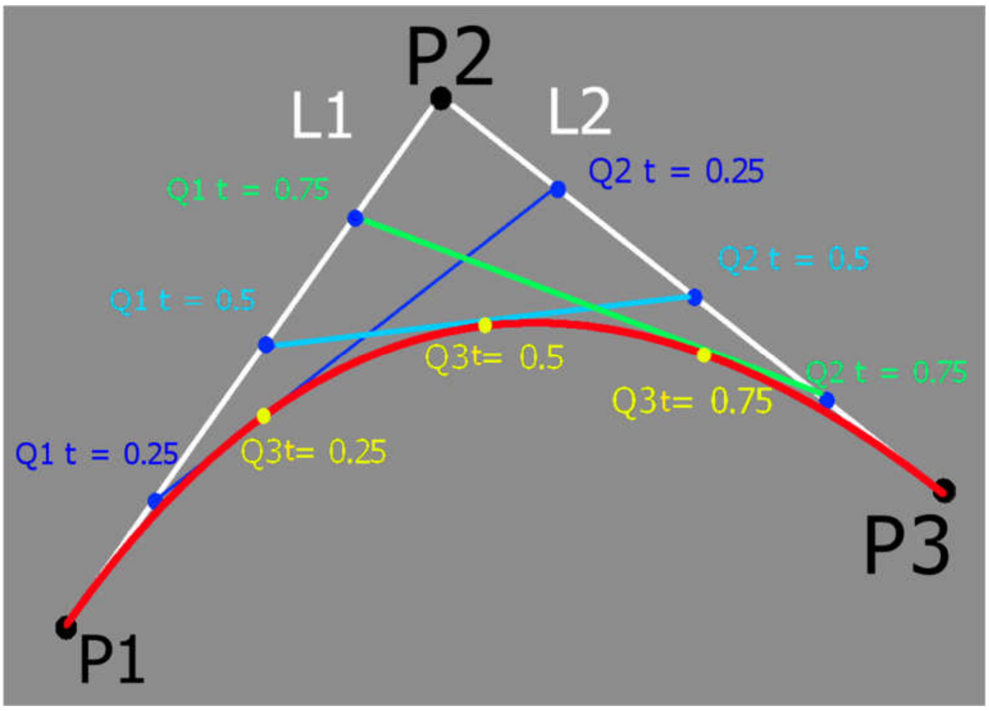


Figure 10. Representation of a quadratic bézier curve

6.4 Observatory

The observatory is a room that contains "planets" that orbit the center of the platform. These planets are spherical game objects that represent SuperApp's work references. They also have a relevant 360° photo applied to them as a texture. The user can use the user interface of the center console to choose a reference scene to be transported into. In the HTC Vive version, the user can use the 6DoF controllers to grab the planets straight from their orbit. When the user puts the grabbed planet on their head like a helmet, the game switches the scene to one that showcases the reference on the planet.

How this works is that the planet has an empty game object with an invisible collider inside it and a script that detects collisions. If the colliding object is the headset, then the script changes the scene to the corresponding scene.

6.5 Reference Scene

Reference scene is a scene where a single reference is showcased. It is enveloped by a sphere that has a relevant 360° image applied to it. The image is plugged into the material of the sphere as the diffusion and the emission texture. The emission channel allows the environment to be lit from the sphere as it emits light. It is the same image that is displayed on the observatory. There are multiple spatial UIs that display media relating to the reference that is being showcased. Figure 11 shows the scene from outside the sphere to explain how the scene works. The player is situated in the middle and perceives the scenery around them.

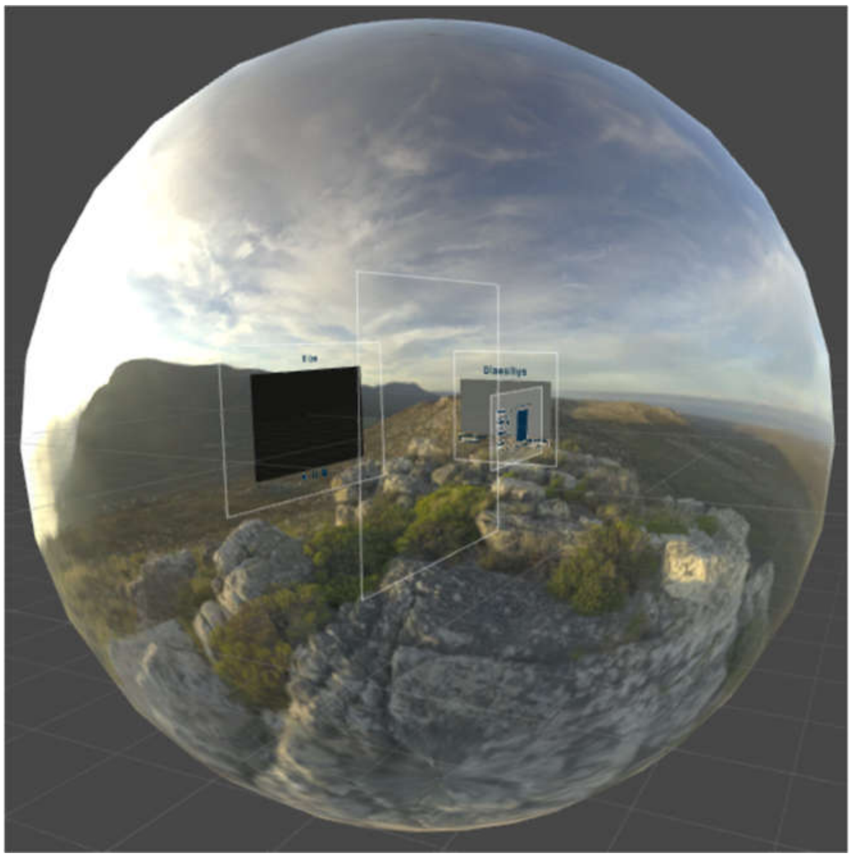


Figure 11 Outside of the reference scene

6.5.1 Video Player

The video player is a spatial UI that can play different videos. It consists of a plane that acts as a "screen" and UI buttons that are used to control the playback of the videos (pause, play, previous and next). The screen is a game object that has a Video Player component. The Video Player component can play a video that is imported to the Unity project or from a URL. The video player component has several render modes:

- Material override renders the video straight to a specified material property, for example the albedo channel (_MainTexture).
- Camera far plane and camera near plane modes render the video to the view of the selected camera. The camera near plane renders the video before the 3D environment making the video the only thing that is seen. The camera far plane renders the video after 3d environment, which makes it seem that the video is behind all the 3D objects.
- Render Texture mode renders the video on a render texture. This render texture can be used in materials like any other texture

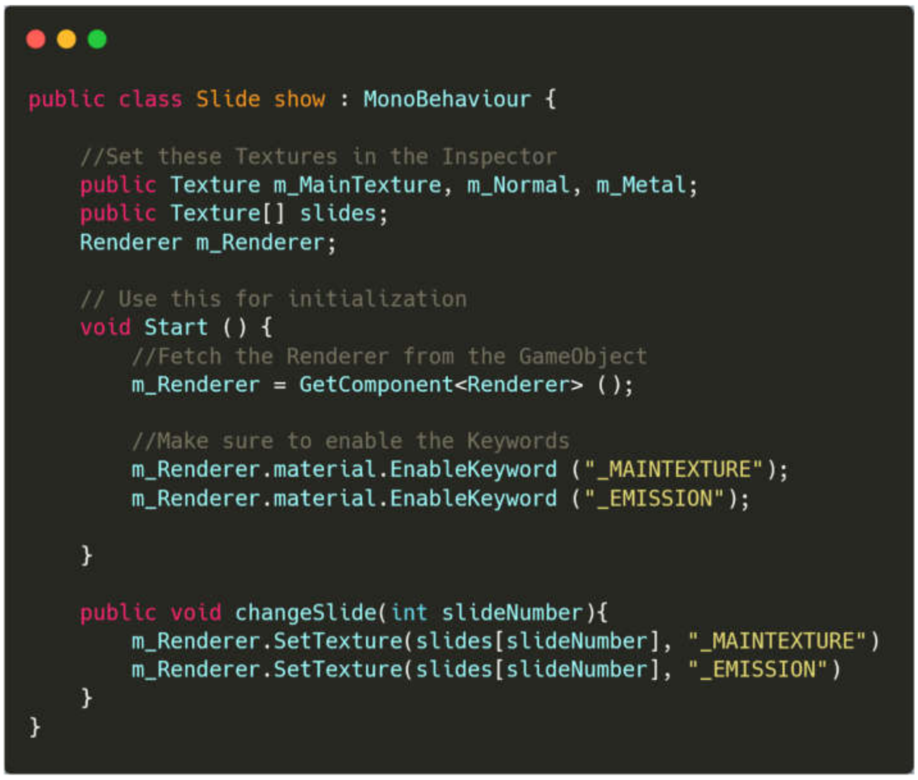
The video source and the playback state can both be altered from a script, making the control with UI buttons easy. (Unity Technologies 2019n.)

In the real world, a television screen does not only change the color of its screen, but it also emits light. Therefore, the video must be plugged to both, the albedo channel and the emission channel of the screen objects material. This is achieved using a render texture (Unity Technologies 2019o).

6.5.2 Slideshow

The slideshow works very similarly to the video player in that it also has a plane that is used as a screen and UI buttons that control the shown media. Switching the image on the screen is done by using Material.SetTexture function. It is used to change the textures of a specified material. It takes in two parameters: the name or the nameID string that is used to determine which texture of the material is changed and the texture class variable that is applied. (Unity Technologies 2019p.)

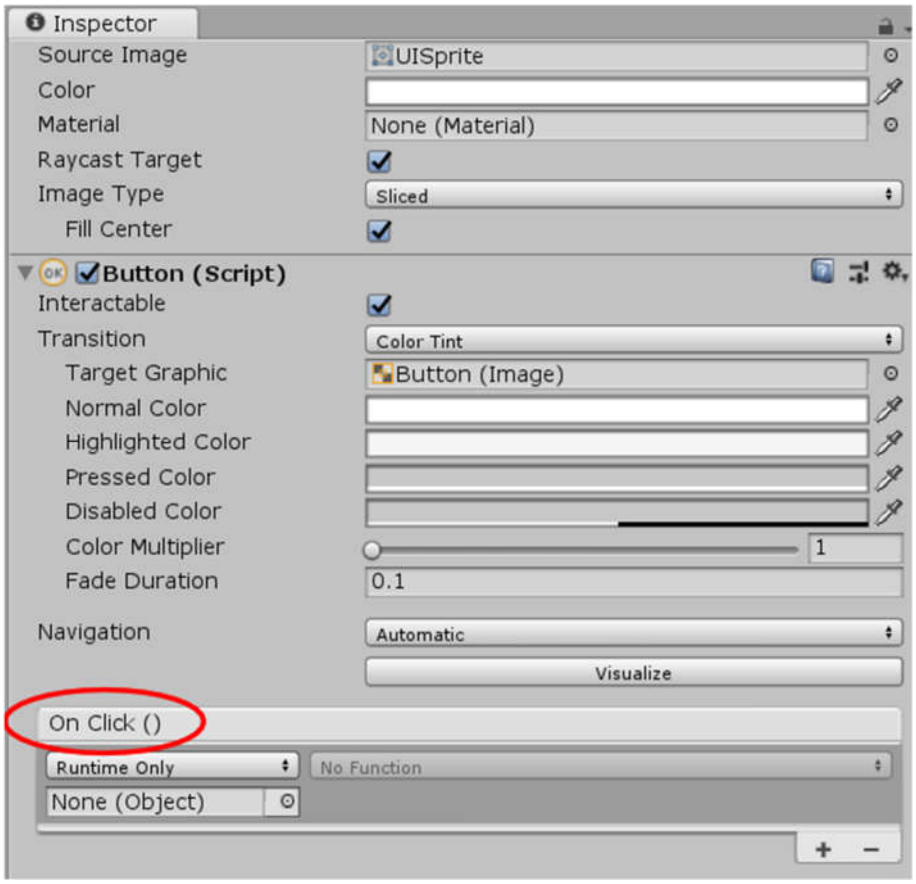
Figure 12 shows an example where the texture is applied to both the albedo (or main texture) and emission channels to make the material act like a screen.



```
public class Slide show : MonoBehaviour {

    //Set these Textures in the Inspector
    public Texture m_MainTexture, m_Normal, m_Metal;
    public Texture[] slides;
    Renderer m_Renderer;

    // Use this for initialization
    void Start () {
        //Fetch the Renderer from the GameObject
        m_Renderer = GetComponent<Renderer> ();

        //Make sure to enable the Keywords
        m_Renderer.material.EnableKeyword ("_MAINTEXTURE");
        m_Renderer.material.EnableKeyword ("_EMISSION");

    }

    public void changeSlide(int slideNumber){
        m_Renderer.SetTexture(slides[slideNumber], "_MAINTEXTURE")
        m_Renderer.SetTexture(slides[slideNumber], "_EMISSION")
    }
}
```

Figure 12. Example of the SetTexture function

The slide show has a script applied to it that has public functions that are used to change the images back and forth. These can be accessed by UI buttons using the Button.On-Click function. Figure 13 shows where to access the onClick function in Unity's UI.



Inspector

Source Image	UISprite
Color	
Material	None (Material)
Raycast Target	✓
Image Type	Sliced
Fill Center	✓

✓ Button (Script)

Interactable	✓
Transition	Color Tint
Target Graphic	Button (Image)
Normal Color	
Highlighted Color	
Pressed Color	
Disabled Color	
Color Multiplier	1
Fade Duration	0.1
Navigation	Automatic

Visualize

On Click ()

Runtime Only | No Function
None (Object)

Figure 13. On-click function in the inspector window

6.6 Drawing Board

The drawing board shown in Figure 14 has two parts: the board and the pen. If the player grabs the pen with the controller of the HTC Vive and collides the tip of the pen to the board, it will draw with the color of the pencil.

The drawing is done in two stages. The first stage is the collision detection which is done by using Unity's OnCollisionStay function. It updates every frame when another collider collides with the object (Unity Technologies 2019q). Then, if the colliding object is a pen, the local coordinates of the collision will be recorded by using the ContactPoint struct and the InverseTransformPoint function. The InverseTranformPoint will convert the global co-ordinates given by the ContactPoint struct to local coordinates (Unity Technologies 2019r).

The second stage is to apply the color of the pen to the drawing board's texture. This is done by using the SetPixel function. It takes in local coordinates and the color as parameters and sets the color of the pixel to the given color (Unity Technologies 2019s). All these parameters are gotten from the first stage.

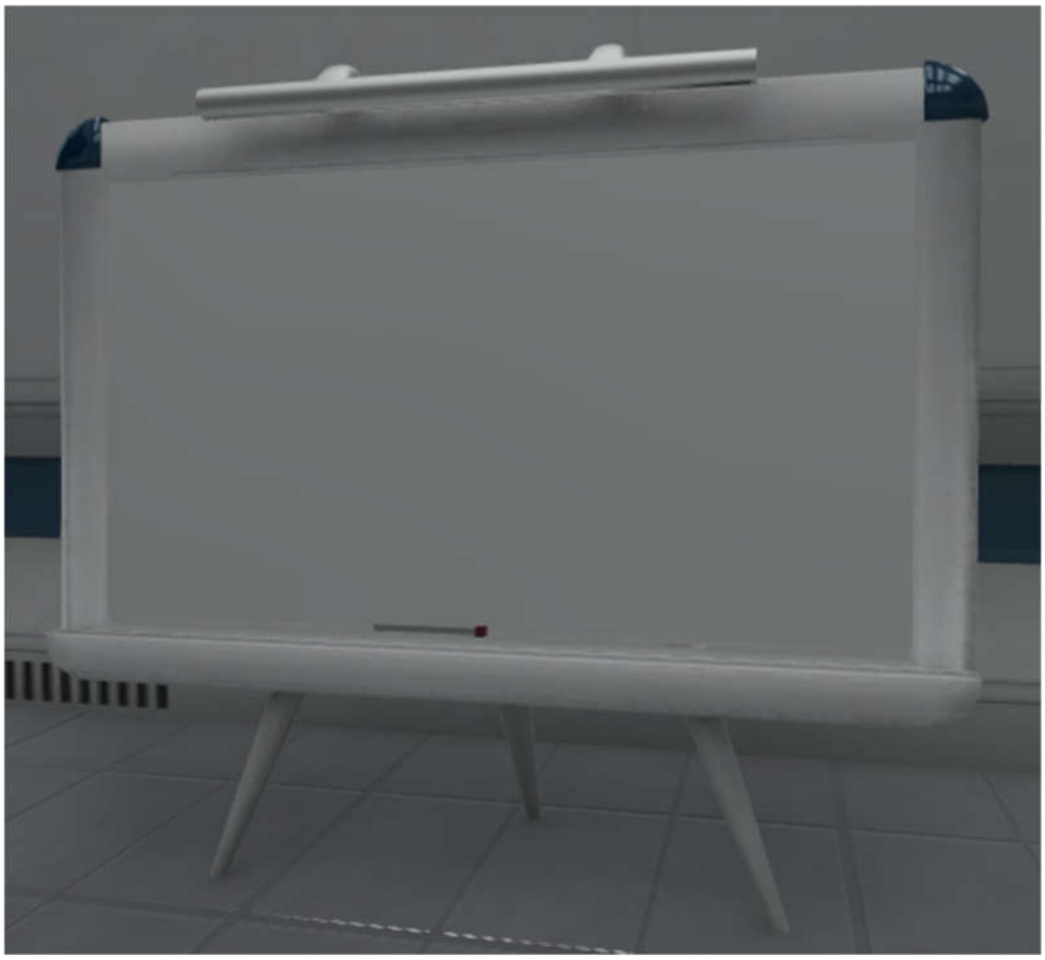


Figure 14 Drawing board

6.7 Grabbing Objects

Grabbing objects is different between 6DoF and 3DoF controllers. With 6DoF controller, the user can move the controller around the 3D environment, and this enables them to reach for the objects and makes it easy to implement grabbing functionality. The objects that can be grabbed must have physics enabled; this means that the game object must have a RigidBody component applied. When the user presses down the button that is mapped to the grabbing function, the script checks for collisions with objects that have a RigidBody component. If one is found, the script will parent this object to the controller and make it kinematic. The parenting "locks" the game object to the controller, but if it still has gravity applied, it will just drop to the ground. Making the object kinematic will essentially disable physics from affecting the object. This is needed to have the object follow the controller while it is being grabbed. When the button is released, the script will reverse these operations, releasing the object from the controller.

Because the 3DoF controller does not track the position of the controller, reaching for the objects is not possible, and the grabbing needs to be implemented differently. The way this is done in the SuperApp VR game is by using the bézier curve that is used by the teleportation system. The script has a button that first draws the bézier when it is pressed down. If the drawn bézier collides with a grabbable object and a button that is bound to the grabbing is pressed down, the object is moved to the controller's position along the bézier curve. Releasing the button will drop the object like with the 6DoF controller.

7 CONCLUSION

The objective of the thesis was to develop a virtual reality game using the Unity game engine. It was made for SuperApp to market their virtual reality development services. The developed application works with the HTC Vive, Oculus GO and Samsung Gear VR headsets.

The developed software can be developed further by adding more functionality on top of the basic functionality that was developed in the thesis. SuperApp’s vision is to add a tutorial event and multiplayer functionality for remote conferencing.

The future of virtual reality seems to be bright as it is forecasted that in 2022, 121 million virtual and augmented reality devices will be sold. It is also predicted that the majority (63%) of the sold virtual reality devices will be dedicated virtual reality devices like the Oculus GO. (CSS Insight 2019.)

SOURCES

Android Central. 2018. Oculus Go vs Samsung Gear VR [accessed 07.05.2019]. Android Central. Available at: https://www.androidcentral.com/oculus-go-vs-samsung-gear-vr

Blender. 2019a. About [accessed 28.04.2019]. Blender Foundation. Available at: https://www.blender.org/about/

Blender. 2019b. EEVEE [accessed 28.04.2019]. Blender Foundation. Available at: https://www.blender.org/2-8/

Chernikov, Y. 2019 Introduction to Rendering | Game Engine series. Video [accessed 07.05.2019]. Available at: https://youtu.be/Hoi-Gzk-How

CSS Insight. 2019. Virtual Reality and Augmented Reality Device Market Worth $1.8 Billion in 2018 [accessed 04.05.2019]. Available at: https://www.ccsinsight.com/press/company-news/3451-virtual-reality-and-augmented-reality-device-market-worth-18-billion-in-2018/

Dykhoff, K. 2012. Non-diegetic Sound Effects. New Soundtrack, Vol.2

Edwards, C. 2013. Valve Lines Up Console Partners in Challenge to Microsoft, Sony [accessed 27.04.2018]. Available at: https://www.bloomberg.com/news/articles/2013-11-04/valve-lines-up-console-partners-in-challenge-to-microsoft-sony

Ivanov, G. 2016. Introduction to Unity Scripting [accessed 28.04.2019]. raywenderlich.com Available at: https://www.raywenderlich.com/980-introduction-to-unity-scripting

Lewis, M. & Jacobson, J. 2002. Game Engines In Scientific Research. Commun. ACM

MedlinePlus. 2019. Motion Sickness [accessed 14.03.2019]. Available at: https://medlineplus.gov/motionsickness.html

Mortenson, M. 1999 Mathematics for Computer Graphics Applications. Industrial Press Inc

Pino, N. 2018. HTC Vive Pro review [accessed 14.03.2019]. techradar. Available at: https://www.techradar.com/reviews/htc-vive-pro

Ogre3D. 2019. About [accessed 26.04.2019] Available at: https://www.ogre3d.org/about

Plunkett, L. 2013. Steam Is 10 Today. Remember When It Sucked? [accessed 26.04.2019]. Available at: https://kotaku.com/steam-is-10-today-remember-when-it-sucked-1297594444

Rouse M. 2016. VR headset (virtual reality headset) [accessed 17.03.2019]. Available at: https://whatis.techtarget.com/definition/VR-headset-virtual-reality-headset

Samsung. 2019. What Is a Graphics API [accessed 02.05.2019]. Available at: https://developer.samsung.com/tech-insights/vulkan/what-is-a-graphics-api

Shiffman, D. 2017 5.0a: Introduction to Physics Engines Part 1 - The Nature of Code. Video [Accessed 07.05.2019]. Youtube. Available at: https://youtu.be/wB1pcXtEwIs

Simpublica staff. 2014. The Sword of Damocles and the birth of virtual reality [accessed 17.03.2018]. Simpublica magazine. Available at: http://simpublica.com/2014/03/19/the-sword-of-damocles-and-the-birth-of-virtual-reality/

Szauer, G. 2017a. Teleport Curves with the Gear VR Controller [accessed 27.04.2019]. Oculus. Available at: https://developer.oculus.com/blog/teleport-curves-with-the-gear-vr-controller/

Szauer, G. 2017b. Teleport Curves with the Gear VR Controller [accessed 27.04.2019]. Oculus. Available at: https://developer.oculus.com/blog/teleport-curves-with-the-gear-vr-controller/

Unity 2019. Industry-leading multiplatform [accessed 14.03.2018]. Unity. Available at: https://unity3d.com/unity

Unity Technologies 2019. Physics [accessed 28.04.2019]. Unity Technologies. Available at:https://docs.unity3d.com/Manual/PhysicsSection.html?_ga=2.165688382.1674815596.1556467922-1047045567.1534768819

Unity Technologies 2019a. User Interfaces for VR. [accessed 26.04.2019]. Available at: https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr

Unity Technologies 2019b. Non-Diegetic. [accessed 26.04.2019]. Available at: https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-vr

Unity Technologies 2019c. Access to fast native graphics API [accessed 28.04.2019]. Available at: https://unity3d.com/unity/features/graphics-rendering

Unity Technologies 2019d. User Interfaces for VR [accessed 28.04.2019]. Available at: https://unity3d.com/learn/tutorials/topics/virtual-reality/user-interfaces-v.r

Unity Technologies 2019e. Audio Overview [accessed 28.04.2019] Unity Technologies. Available at: https://docs.unity3d.com/Manual/AudioOverview.html

Unity Technologies 2019f. About [accessed 28.04.2019]. Unity Technologies. Available at: https://unity3d.com/unity

Unity Technologies 2019g. The Hierarchy window [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/Manual/Hierarchy.html

Unity Technologies 2019h. The Inspector window [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/Manual/UsingTheInspector.html

Unity Technologies 2019i. The Inspector window [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/Manual/UsingTheInspector.html

Unity Technologies 2019j. The Project window [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/Manual/ProjectView.html

Unity Technologies 2019k. GameObject [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/ScriptReference/GameObject.html

Unity Technologies 2019m. Choosing a Lighting Technique [accessed 28.04.2019]. Unity Technologies. Available at: https://unity3d.com/learn/tutorials/topics/graphics/choosing-lighting-technique

Unity Technologies 2019n. VideoPlayer [accessed 07.05.2019]. Unity Technologies Available at: https://docs.unity3d.com/ScriptReference/Video.VideoPlayer.html

Unity Technologies 2019o. RenderTexture [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/ScriptReference/RenderTexture.html

Unity Technologies 2019p. Material.SetTexture [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/ScriptReference/Material.SetTexture.html

Unity Technologies 2019q. OnCollisionStay [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/ScriptReference/MonoBehaviour.OnCollision-Stay.html

Unity Technologies 2019r. InverseTransformPoint [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/ScriptReference/Transform.InverseTransform-Point.html

Unity Technologies 2019s. Texture2D.SetPixel [accessed 28.04.2019]. Unity Technologies. Available at: https://docs.unity3d.com/ScriptReference/Texture2D.SetPixel.html

Unreal Engine 2019. Frequently Asked Questions [accessed 07.05.2019]. Epic Games Available at: https://www.unrealengine.com/en-US/faq

Valve Corporation 2019. About us [accessed 27.04.2019]. Valve Corporation. Available at: https://store.steampowered.com/about/

Virtual Reality Society 2019a. Who Coined the Term "Virtual Reality"? [accessed 17.03.2018]. Virtual Reality Society. Available at: https://www.vrs.org.uk/virtual-reality/who-coined-the-term.html

Virtual Reality Society 2019b. The Unreleased Sega VR Headset – So Much Effort Squandered [accessed 17.03.2018]. Virtual Reality Society. Available at: https://www.vrs.org.uk/unreleased-sega-vr-headset-much-effort-squandered/

VR Bound 2019. Oculus GO [accessed 28.04.2019]. VR Bound. Available at: https://www.vrbound.com/headsets/oculus/go

VRFocus 2018. Oculus Go's Release Date has Been Confirmed [accessed 28.04.2019]. VRFocus. Available at: https://www.vrfocus.com/2018/05/oculus-gos-release-date-has-been-confirmed/

Weinbaum, S. 1935. Pygmalion's Spectacles. The Floating Press

Weis S. 2018. 3DOF, 6DOF, RoomScale VR, 360 Video and Everything In Between [accessed 14.03.2018]. Packet39. Available at: https://packet39.com/blog/2018/02/25/3dof-6dof-roomscale-vr-360-video-and-everything-in-between/