Självständigt arbete på grundnivå

*Independent degree project - first cycle*

Datateknik
*Computer Engineering*

**Forest thinning in VR**
A VR application with the theme of forest thinning

**Thomas Astner**
Abstract

The purpose with this project was to create a virtual reality game were the users should be able to carry out a thinning. The main goals are to use real forests terrains as terrain models in the game, the GameObjects and the teleportation system in the application should be able to handle changing terrains and the application should not cause virtual reality sickness. The application has been developed with the help of the game engine Unity and plugins from Unitys own asset store. User tests and measurements will be carried out in order to evaluate if the game causes virtual reality sickness or not. The results shows that it is possible to use real forests terrains and that the solution is suited for this application. The downside is that in order to use real life terrains several steps has to be taken and that the terrain object has to be designed manually. It also shows that the GameObjects and the Teleportation system has been implemented in a way so they can handle changing terrains. Furthermore it shows that some of the functionalities of the application could be improved, especially the scoring system. The users tests and the measurements showed that the application isn't causing virtual reality sickness but it also showed that the users feels like there are things missing in the application.

Keywords: Virtual reality, VR, C#, Head mounted display, HMD, GameObjects, SCA
Foreword

I would like to thank everyone at SCA that has help me during this project and a special thank to my supervisor Magnus Viström.

I would also also like to thank my supervisor at school, Patrik Österberg that has helped me with the report.
Table of Contents

Abstract.......................................................................................................................... iii
Foreword......................................................................................................................... iv
Terminology..................................................................................................................... vii

1 Introduction................................................................................................................. 1
  1.1 Background and problem motivation................................................................. 1
  1.2 Overall aim ........................................................................................................... 1
  1.3 Concrete and verifiable goals ............................................................................. 2
  1.4 Contributions ....................................................................................................... 2

2 Theory ......................................................................................................................... 4
  2.1 Game engine......................................................................................................... 4
    2.1.1 Unity Personal................................................................................................ 4
    2.1.1.1 GameObject............................................................................................. 5
    2.1.1.2 MonoBehaviour......................................................................................... 5
  2.2 Virtual reality....................................................................................................... 5
    2.2.1 Head mounted displays................................................................................ 5
    2.2.1.1 Lenovo Explorer...................................................................................... 5
    2.2.1.2 HTC Vive................................................................................................. 6
    2.2.1.3 Oculus rift............................................................................................... 6
    2.2.2 OpenVR........................................................................................................ 7
    2.2.3 Virtual reality sickness................................................................................ 7
  2.3 Frame rate capturing software............................................................................ 7
  2.4 Screen recording software................................................................................... 7
  2.5 Photo editing software....................................................................................... 7
  2.6 Heightmap........................................................................................................... 7
  2.7 Refactoring........................................................................................................... 8

3 Methodology................................................................................................................ 9
  3.1 Prestudy................................................................................................................. 9
    3.1.1 Code guidelines............................................................................................. 9
    3.1.2 Priority list.................................................................................................... 9
  3.2 Refactoring........................................................................................................... 9
  3.3 Development tools............................................................................................... 9
    3.3.1 Unity............................................................................................................. 9
    3.3.2 SteamVR plugin........................................................................................... 10
    3.3.3 Microsoft Visual Studio.............................................................................. 11
  3.4 Product evaluation.............................................................................................. 11
    3.4.1 User tests.................................................................................................... 11
    3.4.2 FPS measurements...................................................................................... 11

4 Prestudy....................................................................................................................... 12
  4.1 Code guidelines................................................................................................... 12
  4.2 Priority list.......................................................................................................... 14
5 Implementation
5.1 Scripts ................................................................................. 16
5.2 Components/objects .......................................................... 17
5.2.1 The terrain ..................................................................... 18
5.2.2 SpawnController .......................................................... 19
5.2.3 The trees ........................................................................ 20
5.2.4 The player ...................................................................... 22
5.2.4.1 The controllers .......................................................... 23
5.2.5 The chainsaw ................................................................. 25
5.3 Interaction system ............................................................. 26
5.4 Teleportation system ......................................................... 27
5.5 Scoring system ................................................................. 28
5.6 Importing real life terrain .................................................. 30

6 Results ...................................................................................... 32
6.1 Prevention of virtual reality sickness .................................... 32
6.2 The terrain .......................................................................... 32
6.3 Teleportation system ......................................................... 33
6.4 Priority list ........................................................................... 34

7 Conclusion and discussion ................................................... 35
7.1 Results ............................................................................... 35
7.2 Suggestions for future improvements ................................. 37
7.3 Ethical aspects .................................................................... 37
7.4 Project conclusion ............................................................. 38

References .................................................................................. 39

Appendix A: The user tests ......................................................... 42
Terminology

Acronyms

VR  Virtual reality
IDE  Integrated development environment
HMD  Head mounted display
FPS  Frames per second
1 Introduction

In the 21st century, rapid advancements has been seen in the development of virtual reality (VR). As of the introduction of the modern head mounted displays (HMD) in 2016, VR has risen to be a trendy technology. SCA, together with many others, think that it will keep growing bigger and bigger. 2014 the sales for VR hardware and VR content were only US$108 million, 2020 the worldwide revenue is expected to grow to US$21.8 billion with a compound annual growth of 142%[1]. SCA took their first steps into the world of VR last year when four student (me included), developed an VR application that showed parts from SCA's responsible forestry. The project was a part of the SIMS course held by Mid University in Sundsvall.

Now that SCA had taken the first step into the VR world, the questions that followed was “what is the next step?” and “how do we take this application further?”. VR can be used in many different areas but the choice fell on thinning.

1.1 Background and problem motivation

The problem with thinning is that most people think it's just an extra cost. In reality it is the opposite. It's both good for the forest's well-being and it will give an economical return. Once a thinning has been completed, the remaining trees will get more space and better opportunities to grow undisturbed. This will result in more economical return when it is time for the final felling.

So, how can SCA show people that thinning is a good thing? That is an hard question to answer. Hopefully it can be done through VR, where the immersive experience could make an impact on people.

1.2 Overall aim

The purpose with this project is to create a game where the users, mostly forest owners, should be able to carry out a thinning. The game will be in the form of a virtual reality application which will increase the reality feeling for the users. To increase the reality feeling even more, there is an aim to use forest owner's own forests as model for the terrain in the application. Considering this, every object such as trees, the player, chainsaw and so on, must be placed in the game scene based on the chosen terrain model. To achieve that a priority list will be set up that contains functions/scripts that deems necessary to be implemented in order to make that work.

The aim is that this is going to be a standalone application but at the same time it should be possible to add it as an experience to the existing application created during the SIMS course. Therefore, the developing of this application
will be done in the same Unity project as the existing application. Another benefit with this is that 3D models and prefabs can be reused.

The user movement in the existing application is by teleportation. Teleportation is when the user instantly gets taken to another position. Almost all Teleportation system has a pointer that activates by a button on the controller. The user press down the button, aim where they want to go and when releasing the button they gets taken to where the controller is pointing. The benefits with that solution is that the user doesn't actually have to accelerate to move around in the application. User acceleration is one of the biggest causes for virtual reality sickness (see chapter 2.2.3). However, the existing teleportation system only allows teleportation on a flat plane. That solution has it's problems when the terrain is uneven. So, there is a aim to implement a teleportation system that can handle uneven terrains.

At the end of the project, the goal is to have a working application that can run smoothly. Due to the time limit of the project, the main focus is on functionality and not on the visual.

1.3 Concrete and verifiable goals

Together with SCA and the existing application in mind, four goals were established.

1) The application should not cause virtual reality sickness.

2) Find a solution on how different forest's real terrain can be used as terrain model in the game.

3) Implement a teleportation system that is better suited for this application then the existing system.

4) Establish a priority list with functions needed to carry out a thinning and implement them.

5) A user test should be conducted with the goal that none of the test participants should experince any symptoms that could be realted to virtual reality sickness.

1.4 Contributions

This project has been done individually but in the same project folder as the application created in the SIMS course. Therefore to separate them a folder named Thinning was created. That folder then has sub folders containing all the scripts, materials, prefabs, etc. Even everything that has been downloaded from the asset store has been put in the folder called imports.
The only thing used from the existing project has been 3D-models and the scripts that comes with the SteamVR plugin. No scripts created by students during the SIMS course has been used in this project. There are also nothing that overlaps regarding the reports.

The scene created has been build as a standalone application. It has also been added as an experience to the existing application. Figure 1.2 shows how it's connected to the other experiences.
2 Theory

2.1 Game engine

To understand what a game engine is it is good to get a general idea of what a game library and what a game framework is.

A game library is simply a collection of code that performs tasks from specific domains. Some examples of these tasks are, playing audio, performing physics and handling input. To create a game, you can't use just one game library, several are needed.

So, here is where a game framework comes in. Basically, a game framework is a collection of libraries and tools which together have one task, to create a game.

The next step after a game framework is a game engine. In addition to the libraries and tools needed to create a game, a game engine also contains its own integrated development environment (IDE). The IDE usually contains a scene graph and a level editor. The scene graph is essentially the data structure that holds a games world. A level editor essentially allows the user to design a games world without having to use any code, for example by taking the desired object and drop it into the scene graph. The level editor requirement is probably the biggest difference between a game framework and a game engine.[2]

2.1.1 Unity Personal

Unity is a game engine that has been developed by Unity Technologies and Unity Personal is their free version. Unity Personal offers almost the same features as the other paid versions, one of the biggest differences is that a splash screen with Unity's logo is required when launching a game. Unity is suitable for beginners, students and hobbyists and is a great tool for game creation. Unity is a component-based engine and its main scripting language is C#, but it also has support for UnityScript (a form of JavaScript) and Boo. Unity support over 28 platforms including mobile, web, PC and console.

Unity offers many official tutorials and has one of the biggest game engine communities when it comes to both designers and programmers. The build in Asset Store is also a great tool to find 3D models, plugins, solutions, examples and much more.[3][4]
2.1.1 GameObject

GameObject is the base class for all entities in a Unity scene. GameObject is basically the collection name for all objects. When GameObject is referenced in the text it means that it could be any object like a cube, sphere, terrain, etc. [5]

2.1.2MonoBehaviour

MonoBehaviour is the base class that every Unity script derives from. This class contains many functions that makes it easier to handle GameObjects and events. The two most common and most used functions are Start() and Update(). The Start() function is called only once and that is when the script is enabled, it will always run before the first Update() function call. The Update() function is called every frame as long as the script isn't disabled. As all script are attached to GameObjects you can basically say that Start() runs when the GameObject is created and Update() will run until the GameObject is destroyed. [6]

2.2 Virtual reality

Virtual reality is a three-dimensional, computer generated environment which a user can experience with the help of a HMD. The HMD combined with some sort of movement, often through hand held controllers, makes the experience so immersive that the users get the feeling that they are there both mentally and physically. To make the user get that feeling many of their senses have to be tricked, some examples are sight, hearing, touch, smell and the sense of balance. The more senses a VR application can trick the more real it feels for the user. [7][8]

2.2.1 Head mounted displays

Chapter 2.2.1.1 is going to introduce the HMD used in the developing of this project. Chapter 2.2.1.2 and 2.2.1.3 is going the give some information about the two HMD's that are the foundation of all the modern HMD.

2.2.1.1 Lenovo Explorer

Lenovo Explorer was the HMD used in this project. It was released in October 2017 by the computer giant Lenovo. Lenovo Explorer is a part of the Microsoft Mixed Reality family together with companies like Dell, Samsung and much more. Microsoft is still pretty new in the VR market which puts them a little bit behind HTC-vive and Oculus rift when it comes to the quality and amount of platform specific applications. The good thing is that all headsets in the Microsoft Mixed Reality family has support for SteamVR which makes it possible to use applications found on Steam. [9]
2.2.1.2 HTC Vive

HTC Vive was released 2016-04-05 and was developed by HTC and Valve. It was the first headset that offered a room scaled experience. HTC Vive together with Oculus rift are the two HMD's that are the foundation of all modern HMD's. HTC Vive comes with two sensors, the headset and two motion controllers.[11]

2.2.1.3 Oculus rift

Oculus rift was released 2016-03-28 and was developed by Oculus VR but since Facebook bought them 2014 the owner is Facebook. Oculus rift is shipped with the headset and two motion controllers as standard.[13]
2.2.2 OpenVR

OpenVR is an application programming interface (API) that provides a game with a way to interact with HMD’s without specifically knowing anything about the hardware of the device. Game engines such as Unity, Unreal and Godot have the OpenVR API built in with makes it easier to develop VR applications that works for all HMD’s.[15]

2.2.3 Virtual reality sickness

VR sickness is one of the biggest, if not the biggest problem when it comes to VR. VR sickness can be compared to both motion sickness and seasickness. They all have the same cause, the mismatch between the visual and the vestibular systems and it can give symptoms like nausea, headache, stomach awareness, discomfort, vomiting, etc.[16]

VR sickness are often occurring when the user moves or accelerates the player in the game when standing still in reality. The eyes send signals to the brain that the user is moving. The balance nerves in the ears and the muscles send signals that the user isn’t moving. This mismatching information can sometimes confuse the brain and result in VR sickness.[17]

Low frame rate is also a cause of VR sickness. To prevent this the frame rate should never drop below 90 fps for more then a few seconds. This is a factor that all developers must account for when developing for VR.[18]
2.3 Frame rate capturing software

A frame rate capturing software can measures and displays the FPS of a software at run-time. This is a good tool for VR developing where high frame rate is important.

2.4 Screen recording software

This is a software that makes it possible to take pictures and recordings of a screen. It is commonly used in the game industry to record gameplay. This is a good tool to combine with some frame rate capturing software as it makes it possible for the developer to test the application and later look at the recording to see if the frame rate dropped at some point.

2.5 Photo editing software

This is a software that makes it possible to manipulate digital images. Commonly features in photo editing software’s are image size alteration, cropping, cutting, removal of elements, color changing, etc.

2.6 Heightmap

A heightmap, when talking about computer graphics, is a raster image used to store surface elevation data. The image is often in grayscale where white represents the maximum height and black the minimum height. Heightmaps are often used in terrain rendering softwares and modern video games. By using the elevation data, the image can be converted into a 3D mesh. Figure 2.4 shows how a grayscale heightmap looks like and figure 2.5 shows how the same heightmap has been converted into a 3D mesh.[19]

Figure 2.4: Example of a heightmap in the png format  [20]
2.7 Refactoring

Refactoring is a technique for reconstructing the internal code without changing it's external behavior. The developers goes through the code over multiple iterations. Because every refactoring is small, it's not likely to mess up the code completely. This technique makes the code simpler and easier to understand. [22]
3 Methodology

This section aims to describe the methods and tools that are planned to be used in order to reach the goals in chapter 1.3. Chapter 3.1 will describe the two parts that the prestudy will cover and also why they are in the prestudy. Chapter 3.2 will describe why and how refactoring is used in order to write better code. Chapter 3.3 will present which tools that are going to be used for the development of the application and why they have been chosen. Chapter 3.4 are going to cover the user test and the methods/tools for FPS measurements.

3.1 Prestudy

Chapter 3.1.1 and 3.1.2 will present the two topics that will be covered in the prestudy.

3.1.1 Code guidelines

It is always good to follow some code guidelines. That will make the flow in the code better and make it more consistent. The idea for this project is to carry out a prestudy where some code guidelines will be defined.

3.1.2 Priority list

The plan is to determine some functionalities that deems necessary to be implemented in the application. This will be done in a prestudy and the result will be put together into a priority list. The priority will be related to goal number 4 in chapter 1.3 Concrete and verifiable goals.

4. Establish a priority list with functions needed to carry out a thinning and implement them.

3.2 Refactoring

Refactoring is going to be used to make the code simpler which will make the code more understandable in the end. The thought is that directly when a script is working, instead of being satisfied, the code should be reviewed a couple of times.

3.3 Development tools

Chapter 3.3.1 – 3.3.3 will present which tools that are going to be used for the developing of the application.

3.3.1 Unity

Unity will be the game engine that is going to be used in the project. That became natural since Unity was used under the development for the application
made under the SIMS course. Even if this project wouldn't be in the same project folder, Unity would still be the choice. The reasons is that experience was gained in the SIMS course about Unity, so no unnecessary time is going to be spent on learning a new game engine.

One of Unity's strong points is their asset store where many plugins and prefabs can be found. Unity is also the biggest game engine when it comes to available information. Unity themselves offers many great tutorials and there are many blogs and threads out on the internet which makes it easy to find the information needed. Unity Collaboration is a build-in version handler that Unity offers. This tool makes it possible to choose who the owner of the project is and which users that should have access to it. This will make the transition from the account used in this project to a account owned by SCA smooth.
3.3.2 SteamVR plugin

This plugin contains many prefabs and ready to use scripts which makes development easier. The main reason that the SteamVR plugin are going to be used in this project are for it's camera rig prefab. The camera rig prefab (often referenced to as the player) simply is the object that makes it possible to use the VR headset and the controllers. Another reason is it's connection to the OpenVR api that will make it so the application can be used by all HMDs that has support for SteamVR. Developing a player from scratch isn't an option due to its complexity, the time wouldn't be enough.

The SteamVR plugin also have other prefabs and scripts that will possibly be used in the application.

3.3.3 Microsoft Visual Studio

Microsoft Visual Studio is going to be used when it comes to the coding part of the project. It is the only fully featured IDE for Unity. When installing Unity it will also install Visual studios if the user not manually removes it. So the use of Microsoft Visual Studio for writing the scripts are a given.

3.4 Product evaluation

Chapter 3.4.1 and 3.4.2 will present the methods that are planned to be used for the product evaluation.
3.4.1 User tests

The fifth goal in chapter 1.3 states that a user test is going to be conducted. The user test is planned to be conducted in the end of the project when the application is in a stage where it is testable. The main purpose with the user test will be to see if goal number 1 in chapter 1.3 Concrete and verifiable goals are achieved:

1. The application should not cause virtual reality sickness.

There will be eight test participants that gets to use the application as long as they like. After they are satisfied they have to answer two questions. The first is if they experience any kind of symptoms that could be described as virtual reality sickness. The second question is if they feel like something is missing in the application or should be added to the game.

3.4.2 FPS measurements

A framerate lower than 90 fps can be a cause to VR sickness as described in chapter 2.2.3 Virtual reality sickness. Therefore, the plan is to develop a script that displays a FPS counter on the screen. The thought behind this is that the frame rate can be monitored throughout the whole project.

When the application has reached it's final stage, there is a plan to also use the frame rate capturing software FRAPS. FRAPS is one of the most popular frame rate capturing software and the thought is to see if it gives some different frame rate values then the script.
4 Prestudy

Before the development of the project began a prestudy was conducted. Chapter 4.1 and 4.2 will present what the prestudy resulted in.

4.1 Code guidelines

It is always good to write consistent and understandable code. Especially when someone else is going to take over the project. One way to achieve that is to set up some guidelines. Of course the guidelines must be followed when writing the code, else they would be useless.

Writing scripts in Unity with c# is a little different then just writing regular c# code. A guide specifically aimed at C# in Unity was found and was the basis of the guidelines that this project has followed.[23]

The guidelines that was set up for this project can be seen down below.

- **Bracing:** Open braces should be on the line below the statement and the content inside should be intended by one tab.

  ![Figure 4.1: Bracing and content example.](image)

- **Comment style:** Single line comment should use the two slashes(/) style. Also the placement of comment should be on the line above the code.

  ![Figure 4.2: Multiple line style example.](image)

- **Spacing:** Spacing is a good way to make code more readable. There should always be a single space after a comma between function arguments.
Figure 4.3: Spacing after comma example.

There should not be any space after a parenthesis and function argument.

Figure 4.4: Spacing after parenthesis example.

There should not be a space between a function name and parenthesis. When it comes to flow control statements there should be a space.

Figure 4.5: Spacing after functions and statements example

There should be a space before and after a comparison operator.

Figure 4.6: Spacing comparison operator example.

● Naming: Naming is an important part when it comes to writing understandable code. All class names, function names and variable names should be named so that they make sense, they should be describing. All kind of variables and parameters should follow the camelCasing style. That means that if it the name consists of two words, then the first letter in the first word should be lowercase and the first letter in the second word should be uppercase. If it just a single word then the first letter should be with lowercase.

Figure 4.7: Variable naming example.

Class and function names should use the PascalCasing style instead. That means that the first letter in both the first and second word should be uppercase. If the name is a single word then the first letter should be uppercase.
4.2 Priority list

Placing GameObjects in Unity is usually done by dragging them into the scene and place them on the terrain where it fits. This approach will not work in this project due to the second goal in chapter 1.3 Concrete and verifiable goals.

2. Find a solution on how different forest's real terrain can be used as terrain model in the game.

This will make it so that no terrain will exist in the scene until the user chooses one. So the placing of GameObjects will be depending on whether it exist a terrain object in the game. All placing must be generated by code and the approach will be through functions/scripts. Down below is a list of GameObject felt needed to make this a good experience.

- A Chainsaw
- Trees
- Hands
- The player

The first three are self-explained. The fourth object called the player is the object that makes it possible to use the VR headset and the controllers. It is a must have in a VR application/game.

This was taken into consideration when the priority list was established. Also other functions that deemed necessary to carry out a thinning was thought of. Even thought it is called function description in the table down below it doesn't mean that the solution have to be in the form of a function. It could be a script, several scripts or some other solution.

<table>
<thead>
<tr>
<th>Number</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The trees should be placed based on the terrains position. Meaning that no trees should be placed outside of the terrain and they should be following the height of the terrain, not be flying.</td>
</tr>
</tbody>
</table>
The player should be positioned somewhere inside the terrain. Not flying around outside.

The chainsaw should be positioned somewhere in the terrain.

The controller models seen in the game should be replaced with hand models.

There should be a scoring system implemented.

These items have been the starting point and the development has circled around them.

<table>
<thead>
<tr>
<th></th>
<th>The priority list</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>The player should be positioned somewhere inside the terrain. Not flying around outside.</td>
</tr>
<tr>
<td>3</td>
<td>The chainsaw should be positioned somewhere in the terrain.</td>
</tr>
<tr>
<td>4</td>
<td>The controller models seen in the game should be replaced with hand models.</td>
</tr>
<tr>
<td>5</td>
<td>There should be a scoring system implemented.</td>
</tr>
</tbody>
</table>
5 Implementation

This chapter will present six major parts.

- **Scripts**: This section will shortly describe all the scripts created during this project.
- **Components/objects**: This section will show the objects used in the project and how they are implemented into the scene.
- **Interaction system**: This section will cover how the interactions system has been implemented.
- **Teleportation system**: This section will describe how the teleportation system has been implemented.
- **Scoring system**: This section will present how the scoring system works.
- **Importing real life terrain**: This section will describe how a terrain object can be sculptured into looking like a real life terrain.

5.1 Scripts

The name of the script will be shown followed by a short explanation of what the script does. Chapter 5.2 – 5.5 will go in deeper on how the scripts are used and why they work like they do.

- **ActivateScripts**: This script checks if there exist a terrain object in the scene. If it does, then it activates the scripts “SetPlayer” and “RandomTreeSpawner”.

- **SetPlayer**: This script is responsible to positioning the player based on the Terrains position. It also activates the script “ChainsawSpawner”.

- **RandomTreeSpawner**: This script spawns a predefined number of trees in random locations on the terrain. There are three different kinds of trees and the distribution between them depends on the input the user gives.

- **ChainsawSpawner**: This script sets the position of the chainsaw to be in front of the player and also activates the chainsaw. The chainsaw is disabled from the beginning.
● FPSDisplayer: This script creates a text box in the upper left corner of the screen that displays FPS. It is only visible on the computer screen, not in the HMD.

● InteractableItem: This script makes it possible for GameObjects to be picked up. It also has the responsibility to make the picked up object follow the controller.

● ControllerThinning: To make this script work, it must be attached to either the right controller or the left controller. To describe this script shortly it makes it possible for the controller to pick up GameObjects.

● TriggerAnimationHandler: This script activates the hand animation when the grip button on the controller is pressed.

● ChainsawSound: This script controls the two audio sources that are attached to the chainsaw.

● TreeCollision: This script makes it possible to cut down trees with the help of the chainsaw.

● DisplayTerrain: This script makes it possible for the users to select which terrain model they want.

● Distribution: This script makes it possible for the users to choose between four different tree distributions.

● ResetLevel: This script will make a reset button visible on the computer screen if the space bar is pressed. Clicking on the reset button will restart the application.

● GetInput: This script saves the numbers of trees that the user types in into a global variable.

● DisplayScore: This script activates the information box that shows the score.

● GameEndSphereSpawner: This script positions a sphere in one of the terrain corners.

● EndGame: This script looks for a collision by the sphere and a controller. If a collision occurs then the game will end and the script ScoreCounter is activated.

● ScoreCounter: This script is responsible for counting the score.
5.2 Components/objects

Chapter 5.2.1 – 5.2.5 will go through how the different GameObjects are implemented into the application.

5.2.1 The terrain

When the game starts no terrain object exists in the game. The user is presented with the option to choose which terrain to use from an information box. The information box first consists of a button with the text “Välj önskad terräng” and an empty list. How that looks can be seen in figure 5.1.

![Screenshot of the terrain selector information box before clicking the button](image)

Figure 5.1: Screenshot of the terrain selector information box before clicking the button

Clicking on the button will invoke the function GetTerrains from the DisplayTerrain script. The function first store all the object of the type TerrainData from the folder “Resources/Terrains” into an array of objects. Having objects stored in a folder named Resources will make them accessible by using the Resources class[24]. The function then loops through the array and creates a button for each object. The text on the button will be the same as the objects name and they are placed into the empty list, see figure 5.2. The function will also add a “onClick” listener to the buttons.

![Screenshot of the terrain selector information box after clicking the button](image)

Figure 5.2: Screenshot of the terrain selector information box after clicking the button
Clicking on one of the buttons displayed in the list will invoke that button’s “onClick” function. The function will use the text on the button and store the TerrainData object from the Resource folder with matching name into a variable. The function will then create a terrain object and apply the TerrainData to it. The TerrainData object stores information of how a terrain is sculptured, for example, the heighmap used, the texture, the details, etc[25]. So, applying TerrainData to a terrain will make it render that information. To clarify it even more, if a terrain is created and a texture, some flower and some hills are added. That information will be stored in the terrains TerrainData. If the TerrainData then is applied to another terrain the two terrains will look identical.

5.2.2 SpawnController

The SpawnController is an empty GameObject that has several scripts attached to it. It is the SpawnController that handles the spawning and positioning of the chainsaw, the player, the trees and the sphere that ends the game in the scene.

![Figure 5.3: The SpawnController and the scripts attached to it](image)
There are four scripts that are disabled from start (see figure 5.3. These are the scripts that spawns and positions the GameObjects in the scene. They are disabled due to the fact that there doesn't exist any terrain object when the game starts and can't be placed before one exists.

The script Activate is the one that handles the activation of the other scripts. It checks if there exist a terrain object in the scene. If a terrain object is found, then it enables the RandomTreeSpawner script, the SetPlayer script and the GameEndSphereSpawner.

5.2.3 The trees

Three different kinds of tree types were used in the application: pine, spruce and birch (see figure 5.4). The prefabs are identically except for that they look different of course. The trees have colliders which makes it possible for object to collide with them and not just pass through them. The tree prefab also has a disabled child that is a replica of the tree itself.

As mentioned earlier the script RandomTreeSpawner is responsible for placing the trees in the scene. The script has two functions and that is Start() and Update(). The Start() function stores the terrains length, width, x position and z position into variables.
The Update() function then uses that information to find a random x position and a random z position within the terrain. Then it uses those two values to get the terrains y position for that spot. Then a tree is instantiated at that location.

The RandomTreeSpawner has three public variables that are for the tree prefabs. The number of trees that will be spawned depends on the number typed in by the user in the information box displayed when the game is started. Figure 5.5 shows how the information box for that looks like.

Typing in a number and clicking on the ok button will invoke the function SetTreeNumber from the script GetInput. The function will store the value from the input field and store it in a global variable that later are used in the RandomTreeSpawner script as number of trees to spawn.

How the distribution between the trees will be, depends on what the user chooses from the list that is displayed after the number of trees are selected. Figure 5.6 shows how that information box looks like.
Selecting a distribution will activate the script Distribution which have the task to store the distribution values into global variables that later are used in the RandomTreeSpawner script.

5.2.4 The player

The Player is definitely the most important object in the application. It is the player that makes it possible to use the headset and the controllers.
As seen in figure 5.7 there are a lot of components in the player prefab. Most of the components also have several scripts attached to them which makes the player a very complex prefab. Here is where the SteamVR plugin comes in, it comes with a ready to use player prefab called the camera rig. This camera rig was used in the application made in the SIMS course. The player for this application was created by using the camera rig prefab together with some components from another plugin called VRTK. Why components from the VRTK plugin was used will be further explained in chapter 5.4.

The player is active from the start but before any terrain exists in the scene, the only thing seen in the HMD is a background of a forest. As soon as the user chooses terrain, the script SetPlayer gets activated. The script works similar to the RandomTreeSpawner script. It stores the position and size of the terrain. It uses that information to find a spot on the terrain and gives that position to the player. It also activates the ChainsawSpawner script.

5.2.4.1 The controllers

To make the game feel more realistic the controller models where replaced with hand models. Figure 5.8 shows how the implemented hand models looks against the standard controller models.

![Figure 5.8: On the left is the hand models and on the right the controller models.](image)

The controller models are child's to the controller objects that handles the functions of the controllers. So removing the controller models doesn't break the actual functions of the controllers. So all that was needed to do was to download two hand prefabs, put them as a child to the controllers and deactivate the standard controller models. Now the models have switched from controllers into hands.
The next step was then to add a grip effect to the hands. For this the built-in animation system in Unity was used[26]. To be able to make a grip animation the hand prefab must have the right components. To explain it simply the hand prefab must be divided into parts which can be positioned individually. Take a look at figure 5.9 that is down below this text. The object with blue text is all a part of the hand prefab. The object marked by red are the objects representing the fingers of the hand. It isn't shown in the figure but all the fingers each consist of three or four parts. This allows us to bend a finger in three or four places.

![Figure 5.9: Screenshot of the hand prefab](image)

The Animation system was used by first creating two animation objects and one animation controller object. The standard hand was added to the first animation, which would represent a open fist. For the second animation all the joint in the fingers where repositioned so the animation would look like a slightly closed fist. Figure 5.10 shows all the joint that had to be repositioned.

![Figure 5.10: Screenshot of all the joint that had to be repositioned](image)
Then in the animation controller a flow on how to switch between the two animations where set up. Figure 5.11 shows how the flow was set up.

![Figure 5.11: Shows how the state flow was set up](image)

When the game starts the Entry state sends the hands into the state named LeftAnimation and that is the open fist animation. In the right corner in the figure, there is a Boolean variable named isGrabbing that is set to false. When that variable switches to true the any state will send the hand to the state named GrabAnimation and that is the closed fist animation. When the variable turns false it will be sent back to the LeftAnimation state again.

The object that controls the isGrabbing variable is the script TriggerAnimationHandler. That script checks if the grip button on the controller is being pressed down or pressed up. If the button is pressed down then it sets the isGrabbing variable to true. If the button is pressed up then it sets isGrabbing to false.

### 5.2.5 The chainsaw

The chainsaw object is in the scene from when the game starts but it is inactivated. It is the script ChainsawSpawner that activates the chainsaw and places it in front of the player. It also has a the script InteractableItem attached to it which allows the user to pick it up, more on how that works can be read in chapter 5.3.

The chainsaw also has two Audio Source components attached to it. One has a sound of a chainsaw running idle and the other has the sound of a chainsaw when the accelerator is being pressed. These are being managed by the script.
ChainsawSound. When the chainsaw is being held the idle sound is activated. The accelerator pressed sound will be activated when the user press the trigger button.

The chainsaw also has the script TreeCollision attached to it. That script makes it possible to cut down trees. When a collision with a tree is found and the trigger button is being pressed, a function is invoked. That function activates the child attached to the tree. The child is a replica of the tree but it has a rigidbody component. The rigidbody makes it so the tree is effected by gravity. The child is then released from the parent and the parent is destroyed. A force is also applied to the child which will create the falling effect. To make the feeling of cutting down a tree even more realistic, the function first waits for two seconds before it is executed. This will make it so that the tree doesn't fall down directly when the chainsaw touches it. Removing the chainsaw from the tree or releasing the trigger button before the two seconds have passed will stop the script. So, to cut down trees, the user have to hold the chainsaw against a tree and press the trigger button for two seconds.

5.3 Interaction system

The SteamVR has an interaction system consisting of two scripts. One that is placed on the controller and the other is placed on the object. That makes it so that the object can be picked up with the help of the controllers. This solution was used in the existing application but unfortunately it isn't suited to have in this application. The reason is that the picked up object becomes a child to the controller, which is good when it comes to the positioning, it will follow every move the user makes with the hand. The negative is that it will follow the hand everywhere, meaning that it can pass through object even though they have a collider. Having a chainsaw that passes through the trees wouldn't make a realistic feeling.

So instead of the using the SteamVR scripts, two custom scripts were created. The one for the controller is named ControllerThinning and the one for objects is called InteractableItem.

The ControllerThinning script checks if the controller collides with any object. If it find a collision and that object has the script InteractableItem attached, then it is placed in a hash set. In this application there is only one object, the chainsaw, that has the InteractableItem script attached to it. If it would exist more and the controller would collide with two at the same time then both of them would be placed in the hash set. When the grip button is pressed the script checks the hash set for the closest object and invokes the InteractableItem scripts function BeginInteraction. Releasing the grip button will invoke the InteractableItems function EndInteraction.

The BeginInteraction function will give the object the same postion and rotation as the controller. There are a public vector3 in the script that makes it possible to rotate the object as wanted. As the object aren't a child of the controller it
won't follow it, that is where the Update function comes in. It will check if the object is attached to a controller. If that is true the it will add force to the object to make it follow the controller. Using forces instead of having the object as a child will make it so that the object won't go through other object. In other words the chainsaw won't pass through the trees. The EndInteration function simply breaks the connection between the controller and the object. Figure 5.12 shows how it looks like when the chainsaw is attached to the controller.

Figure 5.12: The chainsaw when it is being held by the right controller

5.4 Teleportation system

The teleportation system used in the existing application was from the SteamVR plugin. As mentioned in chapter 1.2 Overall aim it isn't good at handling uneven terrains.

To solve this the plugin VRTK was used. VRTK can be downloaded from the asset store for free. It is similar to the SteamVR plugin in the sense that it offers many prefabs and ready to use script. The difference is that it doesn't have a direct link to the OpenVR API. Instead it uses SteamVR, or something similar, for that connection. To make VRTK work together with SteamVR, the SteamVR camera rig will be used together with some scripts from VRTK [27].
VRTK offers a teleportation system that allows teleportation on the terrain object and that is the reason it was chosen. To implement it, basically it is just to follow one of their tutorials[28]. The way the teleportation system works is that when the user presses the touchpad button, a pointer is coming out from the controller. When the touchpad button is released, the user will be taken to the position where the pointer is touching the terrain. If the pointer isn't touching any terrain the pointer will turn red and the user won't move. Figure 5.13 shows how the teleportation pointer looks like.

![The teleportation pointer](image)

5.5 Scoring system

To make this a game a scoring system is needed. The scoring system is based on two things, trees cut down and the distance between the trees. When carrying out a thinning approximately 1/3 of the trees are removed and the distance between the remaining trees should be at least four meters in order for them to grow freely.
The way the game works is that the user gets to cut down as many trees he likes. When feeling satisfied there is a sphere located in one of the corners. Touching the sphere with the controller will end the game and the result will be displayed on a canvas that will appear behind the sphere.

First a total tree score is counted. The way it works is that every tree has the value of two from the start, for every tree found that are inside the four meter range, points will be subtracted from that trees score. The ranges and points removed is presented down below.

- Distance < 1, -2 points.
- Distance < 2, -1.5 points.
- Distance < 3, -1 point.
- Distance < 4, -0.5 points.
- Distance > 4, no points removed.

The total tree value will then be multiplied with a value that depends on how many trees the user cut down. The percentage and the multiply value is presented down below. Trees cut down will represent the number of trees cut down and multiply will represent the value that the total tree score will be multiplied to.

- Trees cut down < 15%, multiply = 0.6
- Trees cut down < 30%, multiply = 0.8
- Trees cut down < 45%, multiply = 1
- Trees cut down < 60%, multiply = 0.8
- Trees cut down < 75%, multiply = 0.6
- Trees cut down > 75%, multiply = 0.4

So the result displayed to the user are the total tree score multiplied with the value given depending on how many trees cut down. In order for the scoring system to work optimal 100 trees are recommended for the numbers of trees in the scene. Selecting more or less then 100 trees will make the balance between the two values that are multiplied unbalanced. For example selecting 200 trees and cutting down 33% will make it so that it is impossible to have all the remaining tree outside of the four meter range from each other, the area is to small for that.
5.6 Importing real life terrain

Chapter 5.2.1 describes how the user can select a terrain and how it is created in the game. To be able to use the terrain from a real forest as terrain object in the game, some method of importing data into Unity had to be used.

The terrain object in Unity has an option where an image file in a RAW grayscale format can be imported and used to set the terrain's heightmap. More information on heightmaps can be found in chapter 2.6. Here is a quote from Unity's documentation site regarding the import and export button in the terrain option.

The Import Raw and Export Raw buttons allow you to set or save the terrain’s heightmap to an image file in the RAW grayscale format. RAW format can be generated by third party terrain editing tools (such as Bryce) and can also be opened, edited and saved by Photoshop. This allows for sophisticated generation and editing of terrains outside Unity.[29]

The only information given regarding the image file is that it should be in the RAW grayscale format. SCA has stored cloud point data over different forest areas. From the cloud point data they created a grayscale image file in the tiff format. The image was then imported into Photoshop where it was converted into a RAW file. So, according to Unity's documentation the image should now be able to be imported and be used as the terrain's heightmap, but that wasn't the case. Unity didn't give any warnings but nothing happened to the terrain. Through searching, reading forums and testing, some requirement regarding the image was found. Table 5.1 presents the requirements found.

<table>
<thead>
<tr>
<th>Format</th>
<th>The image must be a 8 or 16 bit grayscale RAW (32 bits doesn't work)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size</td>
<td>The height and the width must be the same. For example 100<em>100 would work but 100</em>90 wouldn't.</td>
</tr>
<tr>
<td>Picture resolution</td>
<td>It must be $2^x + 1$. For example 65, 129, 257, 513, 1025, etc.</td>
</tr>
</tbody>
</table>

Applying these things to the image SCA created made the import work and the image could be used to set the terrain's heightmap. The terrain only receives information about altitude differences from the image. To create a 3D mesh texture, details and so on has to be applied manually by the developer. Figure 5.14 shows how a terrain object looks like when it first is created and figure 5.15 shows how it looks like after a image has been imported.
Figure 5.14: Picture of a newly created terrain object.

Figure 5.15: Picture of a terrain that has imported a RAW image as heightmap
6 Results

6.1 Prevention of virtual reality sickness

Regarding the first goal in chapter 1.3

1. The application should not cause virtual reality sickness.

Frame rate measurements was done both with the script FPSDisplayer and the frame rate capturing software FRAPS. 90 FPS is the optimal and maximum FPS value that can be reached and as seen in Table 6.1, the frame rate was almost 90 FPS all the time with one exception.

Table 6.1: Fps measurements

<table>
<thead>
<tr>
<th>Moment</th>
<th>Frame rate (FRAPS)</th>
<th>Frame rate (Script)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Standing still</td>
<td>90 FPS</td>
<td>90 FPS</td>
</tr>
<tr>
<td>Moving controllers and head (waving the controllers and head)</td>
<td>90 FPS</td>
<td>90 FPS</td>
</tr>
<tr>
<td>Cutting down trees with the chainsaw</td>
<td>90 FPS</td>
<td>90 FPS</td>
</tr>
<tr>
<td>Teleportation</td>
<td>86 – 90 FPS (Dropped down to between 86 – 88 FPS for approximately one second after a teleportation was made, then back to 90 FPS)</td>
<td>84 – 90 FPS (Dropped down to between 84 – 87 FPS for approximately one second after a teleportation was made, then back to 90 FPS)</td>
</tr>
</tbody>
</table>

It was tested with 100, 200, 300 and 400 trees in the scene, they all gave the same result. The terrain object was also the same in all the tests.

The results of the user tests shows that none of the users experienced any symptoms of virtual reality sickness, see appendix A: The user tests.

6.2 The terrain

Regarding the second goal in chapter 1.3 Concrete and verifiable goals

2. Find a solution on how different forest's real terrain can be used as terrain model in the game.
The steps that is needed to be taken in order to use a forest's real terrain as terrain model in the game will be presented down below.

Step 1: First a image is needed that meets the requirements in table 5.1 in chapter 5.7 importing real life terrain.

Step 2: Create a new terrain object in a Unity project.

Step 3: Import the image by using the import RAW option in the terrain settings.

Step 4 (Optional): Design the terrain so it looks more like a real forest. For example by adding a texture, flowers, blueberries, rocks, a cottage, etc.

Step 5: Take the TerrainData of the created terrain, give it a proper name and add it to the project folder “Resources/Terrains”.

Step 6: Build and run the game, the terrain can now be chosen from the terrain list (see figure 5.x in chapter 5.3.1 The terrain).

6.3 Teleportation system

The teleportation system that is used in the existing application comes from the SteamVR plugin. The teleportation system implemented into this application comes from the VRTK plugin. Table 6.2 shows the functionalities of the two systems.

This is regarding to the third goal in chapter 1.3 Concrete and verifiable goals

3. Implement a teleportation system that is better suited for this application then the existing system.

Table 6.2: Functionalities of the two teleportation systems.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>SteamVR</th>
<th>VRTK</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activating the teleportation pointer</td>
<td>By pressing down the touchpad button</td>
<td>By pressing down the touchpad button</td>
</tr>
<tr>
<td>Activating the actual teleportation</td>
<td>By releasing the touchpad button</td>
<td>By releasing the touchpad button</td>
</tr>
<tr>
<td>Valid teleportation object/objects</td>
<td>A flat plane called Teleportation area. All areas covered by other objects (Such as a rock, the terrain, etc) makes those area invalid for teleportation</td>
<td>The terrain object.</td>
</tr>
</tbody>
</table>
How is the teleportation carried out

<table>
<thead>
<tr>
<th>Number</th>
<th>Function description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The trees should be placed based on the terrains position. Meaning that no trees should be placed outside of the terrain and they should be following the height of the terrain, not be flying.</td>
</tr>
<tr>
<td>2</td>
<td>The player should be positioned somewhere inside the terrain. Not flying around outside.</td>
</tr>
<tr>
<td>3</td>
<td>The chainsaw should be positioned somewhere in the terrain.</td>
</tr>
<tr>
<td>4</td>
<td>The controller models seen in the game should be replaced with hand models.</td>
</tr>
<tr>
<td>5</td>
<td>There should be a scoring system implemented.</td>
</tr>
</tbody>
</table>

Figure 6.1: The priority list and the parts implemented

100% of the items on the priority list have been implemented into the application.

### 6.4 Priority list

Figure 6.1 shows the results of the prestudy that were carried out in order to establish a priority list. The green marked rows in the figure shows the results of the implementation of the priority list.

This is regarding to the fourth goal in chapter 1.3 *Concrete and verifiable goals.*

4. Establish a priority list with functions needed to carry out a thinning and implement them.
7 Conclusion and discussion

This section will first cover the conclusions and discussions regarding the results. Then it will move on to some improvement suggestions followed by ethical aspects and lastly a conclusion around the whole project.

7.1 Results

Table 6.1 in chapter 6.1 Prevention of virtual reality sickness shows that the frame rate almost where at the maximum 90 FPS all the time. The only time it went below 90 FPS was right after a teleportation had been done and it only lasted for about a second before going back up again. Noteworthy was that the drop only was down to 84 FPS which still is pretty high. The high fps values is in line with what I expected due to the facts that the application is pretty small and doesn't contain a large amount of GameObjects. Another thing to take into account is that the computer provided by SCA is powerful, using the application on a less powerful computer could result in lower fps numbers. If we look at the user tests, they showed that no participant experienced any kind of symptoms that could be described as virtual reality sickness. I believe that the main reason to why no one experienced any symptoms is that teleportation is used as movement in the application. This removes the biggest cause to virtual reality sickness, accelerating in the game when standing still in the reality. Due to these factors I believe that goal number one and five in chapter 1.3 Concrete and verifiable goals has been fulfilled.

1. The application should not cause virtual reality sickness.

5. A user test should be conducted with the goal that none of the test participants should experience any symptoms that could be related to virtual reality sickness.

Chapter 6.2 describes all the steps that is needed to be taken in order to use a real life terrain as terrain model in the game. So, I would say that the second goal in chapter 1.3 Concrete and verifiable goals are reached due to the fact that it is possible to use a real forest's terrain as terrain model in the game.

2. Find a solution on how different forest's real terrain can be used as terrain model in the game.

So, this solution makes it possible to use real life terrains in the game, but how good is the solution and is it suited for this application? Well, I have interpreted that SCA intends to use the application in two ways. The first way is that it could be release on Steam for free. That would be in a marketing purpose where everyone with a HMD could download the application. The idea here is to create some example terrains that the users could choose from, the intention is
not to let the users create their own terrains. The other way is that SCA lets forest owners carry out a thinning with the forest owners terrains as terrain model in the game. The thought here is that SCA collects data of the forest owners forest's and imports it into Unity. Both ways doesn't require that the users imports the real life terrain, instead it will be SCA that manages that. So, in that sense the fact that importing terrain can't be done in runtime wouldn't really matter. One thing that is required is someone that can managed all this. To learn someone how the importing works is really not that hard, even someone without a programming background should be able to learn it pretty quickly. But, the designing of the terrain also has to be done manually and for that someone with experience is needed. Creating realistic terrains is not necessary but it will increase the feeling of reality for the users. So to answer the question, yes I think this is a good solution and that it was the best suited solution for this project. But, there is potentially another technique that can be used and that is to directly import point cloud data. I call it a technique and not a solution due to the fact that I don't really know if the point cloud data can be used as terrain but I think it's possible. The time frame has stopped me from looking into it more in this project.

Table 6.2 in chapter 6.3 Teleportation system shows how the two teleportation systems works. They work in the same way when it comes to activating the pointer and the teleportation. The biggest difference is on where the systems allows teleportation. The SteamVR system comes with a teleportation area that has to be place above the terrain in order for it to work. This makes it hard to handle when there is a terrain object that is uneven. If the teleportation area is placed to high it will make the user feel like he is flying, to low and the user will not be able to teleport to that area. The VRTK system on the other hand allows teleportation on the terrain so it doesn't matter how the terrain looks like. I feel that the VRTK teleportation system is the optimal choice for this application. I can't really see or think of any teleportation system that would be better suited to handle terrains with different heights. The SteamVR teleportation system isn't even a option in my opinion. If the application only would have one fixed terrain, then I could have made it work by placing many small teleportation areas to make up for the height differences. But, I can't really see a solution involving the SteamVR system to handle different terrains. With this in mind, I think the third goal in chapter 1.3 Concrete and verifiable goals has been achieved.

3. Implement a teleportation system that is better suited for this application then the existing system.

Figure 6.1 in chapter 6.4 Priority list shows the priority list and that everything on it was implemented into the game. The method used for point 1 – 3 was pretty similar, check if a terrain object exist and if it does then use that terrains information to position or spawn the GameObjects. To position the GameObjects based on the terrain feels like the best solution when working with changing terrains. The third point on the list could have been solved in a simpler way, by purchasing hand prefabs from the asset store that has the
animation part included. However, there is always a risk that the assets don't work as you want, so it could be wasted money. I feel like the knowledge gained by doing the animations myself was worth the time it took. The last point however there should be a scoring system implemented could be improved significantly. One factor to this is that I don't have any kind of education in forestry or thinning and another factor is that the scoring system was implemented in the last minute, so not much time was spent on it. Carrying out a thinning takes many more things into account then just the number of trees cut down and the distance between the remaining trees. So I believe that goal number four in chapter 1.3 Concrete and verifiable goals have been achieved but that the scoring system could be improved significantly.

4. Establish a priority list with functions needed to carry out a thinning and implement them.

7.2 Suggestions for future improvements

To improve this application there are a lot of thing that can be done. But really, the way the application is created I feel that there are two things that really would take the application to the next level.

The first thing is the scoring system. As I mention earlier it could be improved significantly. More factors could be added, for example some damaged trees that would decrease the score if they not are cut down. I have to little knowledge about how a thinning really is carried out to give concrete examples of how the scoring system could be improved.

The second thing is how the terrains are designed after they are imported. The area in which the trees are spawned are not the full size of the terrain, it is in the middle. What I mean is that if the terrain is 100m x 100m then the trees spawn in a 50m x 50m area centered in the middle. This leaves a lot of “empty” space on every side. There are so many GameObject that could be added to the terrain, becoming a part of the terrain in other words. For example rocks, trees, cottages, flowers, blueberries and water could all be added to the terrain object. So if someone with the knowledge in 3D modeling, or at least is good in that type of work would design the terrain to work around the trees, then I think the feeling of reality in the application would significantly increase. The user tests showed that the users missed stuffs like rocks, small trees, maybe a cottage or water. All of this can be added in the actual terrain object and they will not interfere with the actual game if the terrain is designed properly.

7.3 Ethical aspects

Virtual reality in general have raised concerns in many different areas. Some examples are desensitization, social isolation, overestimation of abilities, psychiatric, manipulation, etc.
Looking at this application it is hard to really think of some ethical aspects. The aspect I can think of is the fact that the users are carrying out a thinning. I think you have to be careful when you are trying to mimic a real life thing, especially in virtual reality and make it clear for the users that getting high points in the game doesn't mean that doing the same thing in reality will be optimal for your forest. As I mentioned earlier there are many aspect to take into account when carrying out a thinning and those aspects are different for every individual forest. So I think that it is important to make the users understand that this is just a simulation, to claim that this is the way a thinning is carried out would be ethically wrong.

7.4 Project conclusion

Going into this project I didn't really have that many methods more then Unity and the SteamVR plugin that I had worked with in the SIMS course. Of course methods for writing better code and for the evaluation where used but no predetermined methods on how to achieve the goals were used. I think that approach is needed when developing virtual reality applications. The technology in which VR can be experienced through a HMD is still relatively new and it is constantly evolving, so it's hard to determine that a certain method is the right approach.

I think that it has been really great to work with Unity again. The fact that Unity are object based makes it easy to write scripts and the plugins that I have used has worked really great. Especially VRTK, I am really glad that I found that plugin and could use it for the teleportation system. If I would have been forced to write the teleportation system from scratch, then I don't think that I would have been close to come as far as I have now. So VRTK helped me a lot and it contains so much functions so I defenitely think that I am going to use it more in the future.

Overall I must say that I am satisfied with the result. From the start I didn't think that I would be able to solve the importing of real life terrain, so I am really satisfied that I managed to solve it in the end. I think that the methods/solutions on the implementation part really suites this application and I don't think that I would change anything if I got the chance to start all over.

Also I would like to say that I love working with VR, the technology is really cool. I think like many others that the VR market will continue to grow and that the prices will drop so that everyone will be able to experince VR.
References

Retrieved 2018-06-20 Published 2016-12-06

Retrieved 2018-04-17

https://store.unity.com/products/unity-personal
2018-04-23

https://unity3d.com/unity/features/multiplatform
Retrieved 2018-04-23

https://docs.unity3d.com/ScriptReference/GameObject.html
Retrieved 2018-05-24

https://docs.unity3d.com/ScriptReference/MonoBehaviour.html
Retrieved 2018-04-23

https://www.vrs.org.uk/virtual-reality/what-is-virtual-reality.html
Retrieved 2018-05-03

Retrieved 2018-05-03

https://xinreality.com/wiki/Lenovo_Explorer
Retrieved 2018-05-27
Forest thinning in VR – A VR application with the theme of forest thinning
Thomas Astner
2018-06-21

[10] CashConverters,
Retrieved 2018-05-27

[11] The Verge, “Valve's VR headset is called the Vive and it's made by HTC”,
Retrieved 2018-05-27

[12] Inet, “HTC Vive”,
https://www.inet.se/produkt/1999813/htc-vive
Retrieved 2018-05-27

https://arstechnica.com/gaming/2014/03/facebook-purchases-vr-headset-maker-oculus-for-2-billion/
Retrieved 2018-05-27

Retrieved 2018-05-27

https://github.com/ValveSoftware/openvr/wiki/API-Documentation
Retrieved 2018-05-03


https://www.webmd.com/cold-and-flu/ear-infection/motion-sickness#1
Retrieved 2018-05-03

[18] IRIS VR, “The Importance of Frame Rates”,
Retrieved 2018-05-07

https://en.wikipedia.org/wiki/Heightmap
Retrieved 2018-06-02
Forest thinning in VR – A VR application with the theme of forest thinning
Thomas Astner

https://upload.wikimedia.org/wikipedia/commons/5/57/Heightmap.png
Retrieved 2018-06-02

https://upload.wikimedia.org/wikipedia/commons/2/2f/Heightmap_rendered.png
Retrieved 2018-06-02

[22] Refactoring, “Refactoring”,
https://refactoring.com/
Retrieved 2018-05-07

[23] Unify Community, “Csharp Coding Guidelines”,
Retrieved 2018-04-23

https://docs.unity3d.com/ScriptReference/Resources.html
Retrieved 2018-05-27

https://docs.unity3d.com/ScriptReference/TerrainData.html
Retrieved 2018-05-27

https://docs.unity3d.com/Manual/AnimationOverview.html
Retrieved 2018-05-31

[27] VRTK, “Getting started”,
https://vrtoolkit.readme.io/docs/getting-started
Retrieved 2018-05-31

[28] Youtube, “[Basics] Pointers, Teleporting and Object Interactions”,
https://www.youtube.com/watch?v=sW9JxEU5fe8
Retrieved 2018-05-31

[29] Unity Documentation, “Terrain settings”,
https://docs.unity3d.com/Manual/terrain-OtherSettings.html
Retrieved 2018-06-05
Appendix A: The user tests

The users got to test the application for how long as they liked. There was no restrictions and if they had some questions they were free to ask.

The scoring system hadn't been implemented at the time of the user tests and no information on how the game works. The purpose of the user tests was to see if anyone felt anything similar to virtual reality sickness.

After the test two questions were asked:

1. Did you feel anything that could be described as virtual reality sickness, headache, stomach sickness or anything like that?
2. Is there anything you think is missing or that should be added in the game (Except for the obvious scoring system).

Person 1

1. Didn't experience any symptoms.
2. Sawdust from the chainsaw when cutting trees. It would be nice if a stump appeared where the tree was cut down.

Person 2

1. Didn't experience any symptoms.
2. Some smaller trees to make it feel more like a real forest.

Person 3

1. Didn't experience any symptoms.
2. Add something around the tree area, maybe a cottage, some rocks, water or something like that.

Person 4

1. Didn't experience any symptoms.
2. Sawdust from the chainsaw. Would be nice to choose how many trees there is in the game.

Person 5

1. Didn't experience any symptoms.

2. Had nothing to comment

Person 6

1. Didn't experience any symptoms.

2. Smaller trees to higher the forest feeling. Would be nice to be able to choose the distribution.

The user test was a small test due to the fact that the application not was in the state of a game. Some parts of comments regarding questions two were implemented in the application after the user test. The thing that where implemented after the user test:

- A choice for the user to choose distribution of the trees.

- A choice for the user of how many trees to have in the game.