Universitetet i Stavanger

DET TEKNISK-NATURVITENSKAPELIGE FAKULTET

# BACHELOROPPGAVE

| | |
|---|---|
| Studieprogram/spesialisering:<br><br>Bachelor i ingeniørfag /<br><br>Data | Vårsemesteret 2021<br><br>Åpen |
| Forfatter(e): Mads Henrik Tonheim, Alexander Pedersen Halvorsen | |
| Fagansvarlig: Erlend Tøssebro<br><br>Veileder(e): Erlend Tøssebro | |
| Tittel på bacheloroppgaven: Edutainment Escape Room<br><br>Engelsk tittel: Edutainment Escape Room | |
| Studiepoeng: 20 | |
| Emneord:<br><br>VR, Edutainment, Escape room,<br><br>Unity, C# | Sidetall: 64<br><br>+ vedlegg: Appendix A<br><br>Stavanger 15. mai 2022 |

# Contents

# Acknowledgements

We would like to thank our supervisor Associate Professor Dr. Erlend Tøssebro at the Department of Electrical Engineering and Computer Science (University of Stavanger) for his guidance and support throughout this thesis.

# Chapter 1

# Background

## 1.1 Description

This thesis is based on previous assignments where the goal was to create educational games based around elements of different institutes at the University of Stavanger (UiS). The assignment was suggested by the Department of Electrical Engineering and Computer Science. The main objective is to create an entertaining game while also focusing on introducing players to subjects the University of Stavanger has to offer. VR (Virtual Reality) will be used to make the game both entertaining as well as educational.

Games have become more and more widespread over the years, which makes this thesis interesting and relevant.The motivation behind the development of the game is to create something new and interesting to inform and motivate students to study at UiS. The idea and mindset that kept the project going was the wish to create a virtual environment filled with engaging educational challenges which were both entertaining and fun to explore.

The main focus for the game is to overcome challenges in a virtually created

escape room primarily based around coding and programming. One of the reasons for this choice is to create something different and new compared to previous groups with a similar assignment. Instead of creating multiple mini-games this thesis focuses on one Escape Room based around different elements of Computer Science filled with different creative tasks to learn more about Computer programming.

## 1.2 Introduction to Virtual Reality

Virtual Reality, often referred to as VR, is an artificial simulation of reality. The player gets to experience real-life physics and movement in a 3D-world. This allows the user to interact with the environment, hear sounds around them and being able to move around in a simulated environment. This makes the experience both realistic and engaging.

Some VR systems also comes with their own set of controllers to interact with the simulated environment. These are called motion controllers and allows the player to simulate real-life hand movement and behaviour inside the game. We often categorize VR in three categories. Fully Immersive Virtual Reality, Semi-immersive Virtual Reality and Non-immersive Virtual Reality. Fully Immersive Virtual Reality is the one used for this project where the goal is to make a realistic and interactive environment. It also provides the player with the feeling of being present in the virtual world. To accomplish an immersive experience like this, VR headsets are commonly used. Semi-immersive Virtual Reality is an in between type of virtual reality where the player is still connected to the real world environment, but is able to control the 3D virtual world with the help of physical objects. To give an example of a semi immersive experience, racing simulation could be mentioned. Racing games can sometimes be further immersed with the right equipment like cockpits, steering wheels or haptic feedback from controllers which makes it semi immersive. Non-immersive Virtual Reality is the type most common today where the 3D environment is controlled through a computer. This type of virtual environment does not interact directly with the player. Most computer games today still consists of the non-immersive type.

## 1.3 Game Information

The objective is to create an interesting and unique experience in the form of an escape room. The escape room itself consists of different challenges the player will have to overcome in order to escape. The challenges will require logical thinking and basic understanding of the represented study. Some of the tasks will require set pieces in order to fully understand the meaning. These will be hidden around in the room and make the player actively search for the missing clues during the game.

The tests are based on programming where one of them is to fill out an if-statement so the correct lines of code run. Once completed the player will receive one of the keys to open the final door. There will be multiple rooms with tasks, as having all the tasks in one room proved to be confusing to some players. There are also tasks for figuring out what value a variable has after some code has ran, where the player types the value in on a keypad. One of the other tasks features a while loop used for revealing a key. The key can become visible without the loop, but only for a short burst of time. If statements such as "if(false)" are used the key will not show at all.

Multiple objects and set pieces are present in the room during the trial. Some of them are possible to interact with and could prove useful for one of the upcoming challenges. There are also visual clues on the walls or hidden messages in the room for the player to find. These are not necessary for completing the challenges, but can be helpful if the player has no previous coding experience.

Once the player has gone trough all the rooms, completing all the challenges and put the keys in the correct spot the game will progress to the final puzzle. This one will be based upon some of the previous puzzles and also runs on a timer.

## 1.4 Why Escape Room and Programming

Escape rooms are something most people are familiar with. You solve puzzles in order to get closer to your goal of escaping. The tasks themselves can be interactive and require you to find clues and study the room you are in, which makes things interesting for the participants. It became clear early on that combining this experience with programming based puzzles could enlighten and motivate players to learn more about the subject. Programming has become increasingly more relevant every year and quickly became the natural choice for the room puzzles. Doing research beforehand also revealed there were few similar games with both escape room and programming as the main theme. This also made the project more exciting to work with.

# Chapter 2

# Equipment and software

## 2.1 VR Headset

When it comes to Virtual Reality Headsets there are different variants that work for various situations. Headsets such as the Oculus Quest 2 and HTC Vive works well for most situations. The HTC Vive headset comes with a wide variety of additional tools compared to the Oculus Quest 2, like the possibility to connect directly to the PC. It comes with two sensors that makes it possible for the player to move with a 6 degrees of freedom also called 6DOF. 6DOF refers to the movement freedom, in this case that the player can freely change position in any direction. The Oculus has four built in sensors to track movement and also supports the 6DOF tracking technique. Both of these headsets have been used during development. They also come with two controllers and the HTC Vive headset also contains two sensors. They both have HMD displays (Head-mounted display). The HTC Vive and Oculus Quest 2 uses different methods for tracking the player movement. HTC Vive uses two infrared lighthouses. The infrared light makes it possible for the system to track movement. The Oculus Quest 2 uses markerless tracking and does not need additional sensors to be mounted. Instead it uses the cameras on the headset for a process called SLAM (Simultaneous localization and mapping). This process makes it possible to generate a 3D map of the environment in real time. The tracking of the player works well on both devices, but the HTC Vive requires

manual setup for the tracking area and ground level. The Oculus headset automatically configures this through the cameras, but it is required to create your own play area through the headset configuration as well as ground level. Since the Oculus only supports cameras at the front of the headset the tracking can be lost if the controllers are hidden away from behind the player or objects that blocks the view of the cameras. Similar problems can be seen with the Vive if the sensors are badly configured which causes worse tracking. Both headsets uses controllers with a touchpad and buttons like grip and trigger. Similar button layout on the controllers makes it easier to work with two headsets. The buttons can be programmed to interact with objects in the virtual world.

## 2.2 Unity Software and Description

It was decided to use unity when creating the Escape Room mostly because it looked suitable for the project, while it also felt familiar due to the fact that the scripts support C# which both group members had experience with from before. Unity is a popular choice among developers and is used by both big and small game companies. It offers different licences depending on the features you need for development. The free licence targets beginners and the PRO licence provides additional services and features to assist when producing a higher quality game. The PRO licence was provided for free and used on this project through the education program unity offers for students. An important reason to why this game engine was chosen over other options was the availability, beginner friendly layout and UI (User Interface). Unity has its own asset store website that can be used to import models among other things to make it less tedious and time consuming to design every game object by yourself.

## 2.3 Unity Hub and Editor

Opening Unity will boot up the Unity Hub which presents you with a list of the projects created on the computer or an option to create a new one. Account information and licence will also be presented here. It will also display the Unity Editor version used on each project. Not only can the

## 2.3 Unity Hub and Editor

Unity Hub be used to choose version and project to work on, it can also provide tutorials made by the community to get a better understanding of how Unity works.

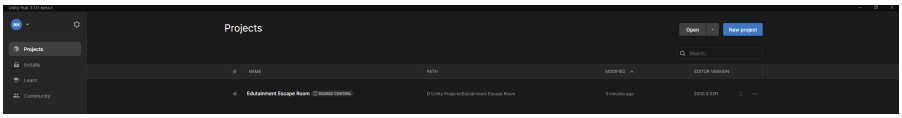


**Figure 2.1:** Unity Hub project list

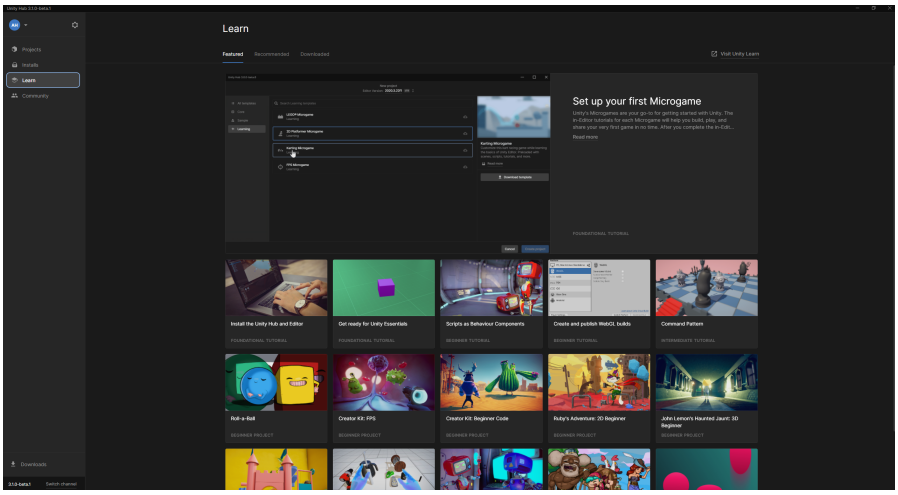


**Figure 2.2:** Unity Hub learning/tutorial tab

Creating and opening a new project generates an empty scene in the Unity Editor. The scene is a 3D environment where all of the models and environment are designed. Main camera and directional light is already deployed in the scene and can be modified by the developers liking. The camera component is responsible for rendering the game and makes it possible for the player to see the scene. Everything from changing the camera perspective and the field of view can be done on this component. The directional light determines where the lighting should come from and at what angle. Shadows can occur depending on the position of this light. Creating dark rooms or day/night scenarios in the game is completely possible when working with directional light in Unity.

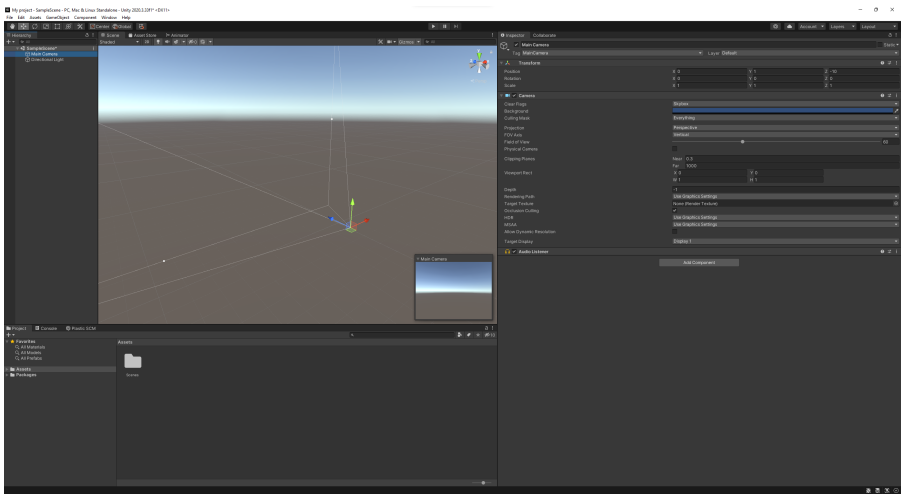

**Figure 2.3:** Unity Editor when opening a new project

### 2.3.1 Game Objects and Components

Game Objects are one of the most important things in a game. It serves as the objects you put in your game, but doesn't necessarily do anything by itself. Every game object can be modified with components to make it behave the way you want it to with scripts for example. The game objects gets placed to the left of the scene as shown in the picture below. When creating multiple game objects it is important to sort the objects into categories to create better accessibility in the future. Another important note to remember about game objects is that you can create prefabs to store the object that has been created. This will save components, position and everything stored inside. This can then be used later without the need to replicate the previous game object by hand.

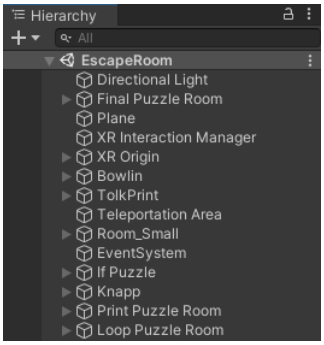


**Figure 2.4:** Some Game Objects from the project

Components are made to create behaviour for the Game object. What the components do for the objects varies, but defines what it does in the game. Some components can execute scripts to add behaviour while other components creates collision, position and size of the object. When inspecting a game object in unity the inspector shows up on the right side of the scene and gives an overview of what components the object consists of. Some of the most common components are listed below starting with the component that almost every object has as a default.

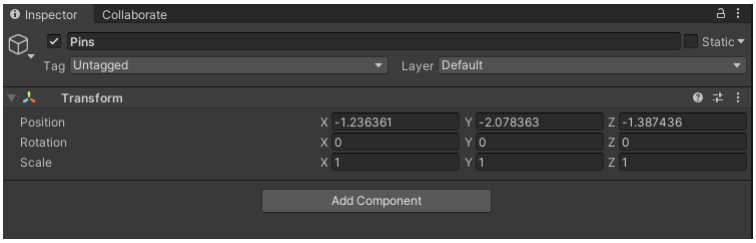


**Figure 2.5:** Inspector showing the components tab for an object

Some of the most common components used in the project will be listed below with a short description on each one:

- Transform - This is considered to be one of the most central components for Unity development and is therefore almost always present by default in the inspector when creating an empty game object. Trans-

form lets you control the size, position and rotation of the selected item

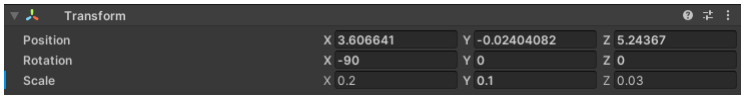


**Figure 2.6:** Transform Component

- Collider - The colliders makes it possible to establish physical interaction with the rest of the objects in the game. There are different types of components to use depending on the object you are creating a collider for, but one of the standard ones is called the box collider. There are different attributes to change like editing the collider itself (Position, rotation and angle). There is also an option to make the collider trigger on certain events using the "is Trigger" button.

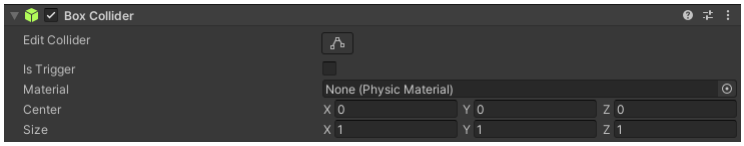


**Figure 2.7:** Box Collider Component

- Material - Materials shapes the color of the game objects and always comes with a default setting. With the material component it is possible to change color preset of the object and also change the shader. Shaders can help with making the object look more realistic by changing how the lighting affects the game object.

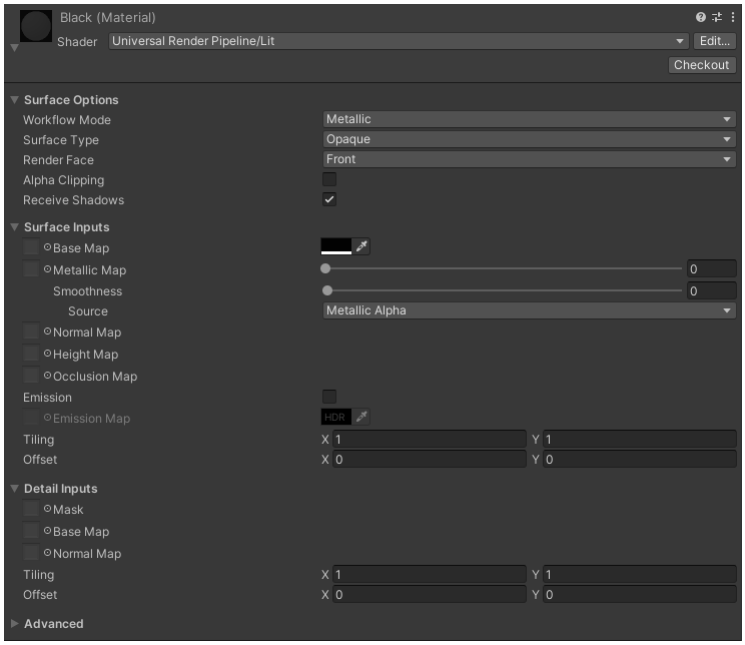


**Figure 2.8:** Material Component

- Mesh Renderer - A mesh can be seen as the graphic on an object and contains small triangles that all form a shape that can be seen in the scene. There are different types of mesh components, but one of the most important ones would be the mesh renderer. It renders the object to make it look like an object in the game. Without this component or disabling it completely would make the object invisible.

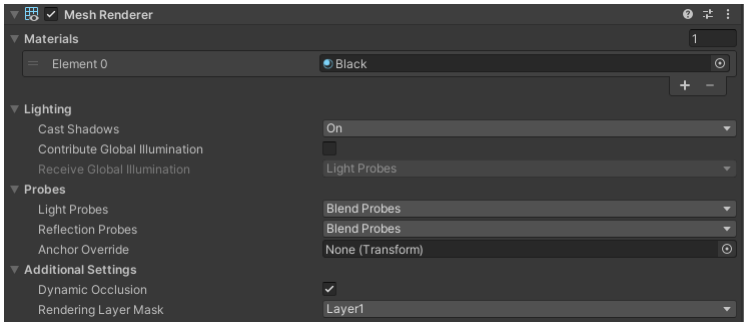


**Figure 2.9:** Mesh Renderer Component

- Rigidbody Components - Multiple Rigidbody components can be cho-

sen from the "add component" list in Unity. The Rigidbody component makes it possible for objects to be affected physically in the environment. This can be controlled by changing the attributes for mass, force, gravity, etc.

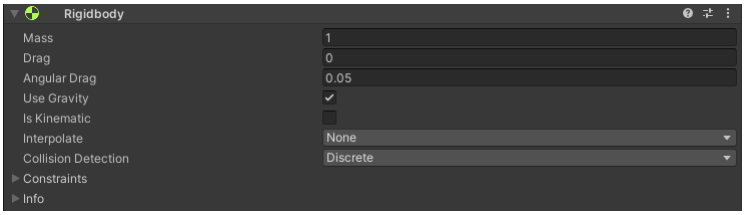


**Figure 2.10:** Rigidbody Component

- VR Components - One of the most common VR components in Unity is the XR Grab Interactable which makes it possible for the player to interact and grab object in a VR environment. Both the Socket Interactor and the XR Grab Interactable component have been used frequently throughout the project. The socket interactor allows for certain objects to be put into sockets, for example puzzle items. Both of these can be customized. The socket interactor can be modified to only allow certain objects among other things such as transform and Interactor events, as shown on the picture below. The XR grab component lets you change the movement type of the item grabbed by the player while also being able to smooth out the tracking and position of the game object.

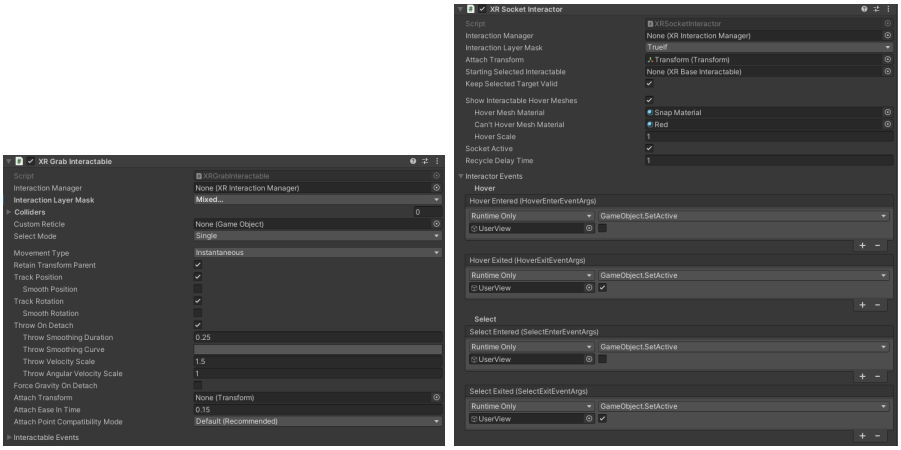


**Figure 2.11:** VR Grab and Socket Component

### 2.3.2 Position and Scaling

Positioning and scaling is essential in Unity development and is commonly changed or modified through the hierarchy on the right side with the objects. Creating game objects often presents you with a transform component, but if you create a set of objects, one parent object can make the transforming easier to deal with. The child objects can also be changed and transformed, but is relative to to the parent and centered around it.

### 2.3.3 VR Player

Every VR project requires a virtual reality player that can interact with the world. In VR the player is always shown through a first person perspective which means that you see the environment through the character's eyes. The only form of visibility you have on the character would be the player hands. Making a full body model is also possible, but won't matter in a single player escape room. The models used for the hands was imported using the Oculus Hands assets. The animations was added on top of the models using the same asset package.

### 2.3.4 Scripts and C#

Scripts are used on Objects as components to make the game function properly, and therefore makes it an essential building stone on how to make a proper game in Unity. Grabbing objects, pressing buttons, movement and logic behind puzzle are all perfect examples of features made with different scripts. The scripts themselves are all programmed using the object-oriented programming language C# as mentioned before. It is the only supported language in Unity and handles all the functionality of our program. When opening an empty script in Visual Studio Unity automatically adds some default packages. These packages are imported with the term "using" in Visual studio. From the very start there are already two functions/methods created called Start and Update. The default empty script can be viewed below. The first method Start() runs immediately on the first frame when the game starts while the Update() method runs constantly every frame to check for updates.

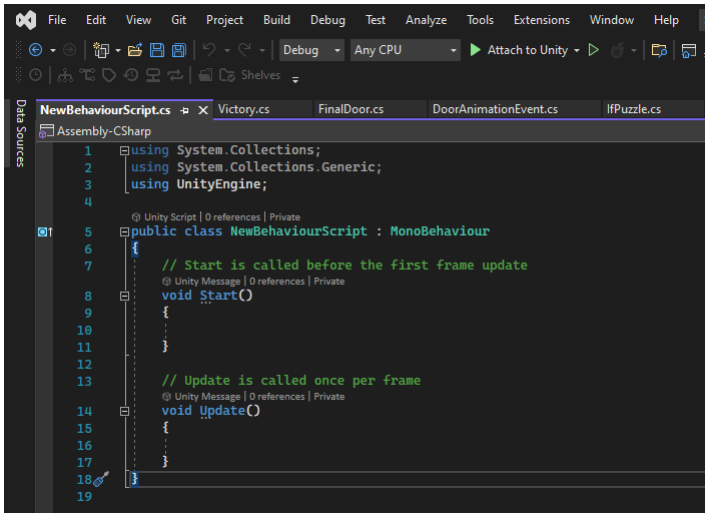


**Figure 2.12:** Image of a newly created script in Visual Studio

Creating variables in scripts and making them public makes it easy to test in the editor. The editor will show the variable as an input depending on how the variable was made. This will be shown to the right as a component for the object and can be changed live during testing and playing the game.

Some variables can be made to reference certain objects, others can demand certain values and some can just be true or false inputs.

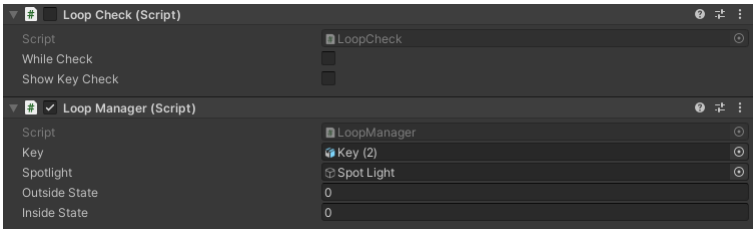


**Figure 2.13:** Some examples of changeable variables from two different scripts

### 2.3.5 Assets

Assets can be central when it comes to Unity development and represents anything that can be imported into your project that has already been made or developed. This can include 3D environments or models, controller support, images, scripts, etc. Assets can be imported using the Unity website called the asset store. Previous versions of Unity also supported an integrated asset store that could be reached from the editor, but as of Unity version 2020 that is no longer an option. Importing the assets from the website will put them in the package manager in the editor. There will be an option from the package manager to import the selected asset into the active project. You can then modify which files you want to import instead of the whole package. This project consists of assets for the 3D environment and some for specific controller integrations. A list of the external assets used in the Edutainment project will be listed below.

- XR Interaction Toolkit - This toolkit provides a component-based interaction system for VR creation and a framework that makes 3D and UI interactions available from Unity input events. It also contains helper components for visuals and hooking in your own interaction events.
- Oculus Integration - The Oculus integration brings extended support for Oculus VR devices and some Open VR supported devices. The asset itself contains scripts and prefabs among other resources. This

project makes use of both the HTC Vive and Oculus Quest 2 headset which makes the package useful for this exact situation.

- VR Beginner Escape Room - This asset package introduces basic 3D environment models and props. It also contains a prototype scene to demonstrate how the assets can be used in other projects. Some of the Escape room based models were used in the final product, but the package itself was mostly used during the early learning stages of development.

- Snaps Prototype | Office - A package that comes with multiple office like model designs. Contains models of walls, floors and ceilings along with multiple props like computers, desks cabinets, etc. This asset package was used mainly for the room design and some of the props in the room.

- FREE Music For Puzzle Games - This one comes with 10 different music tracks fit for puzzle games. Only two of the tracks were used in the project and they are all license free. It was decided to add background music for the game to create an atmosphere as opposed to complete silence.

- Terrain Sample Asset Pack - Package contains a collection of terrain assets used to create realistic outdoor environments. The assets were for raising the terrain to create mountains around the escape room. The grass textures from the asset pack were also used to make the outside area of the game.

- Nature Starter Kit 2 - To further improve the visuals of the outside the nature starter kit were needed. It comes with bushes and grass textures to enhance the nature aspect of the game.

One of the most common VR tool kits in the Unity asset store is called VRTK (Virtual Reality Tool kit). It comes with a lot of helpful implementations like compatibility with other VR SDKs (Software development kit). It also makes it possible to simulate VR games in Unity without using a VR headset to test. The package also comes with a lot of other useful tools like UI tools and teleportation scripts. The 2020 version Unity makes this asset non compatible and was therefore not used in this project. Many of the perks that came with VRTK are now integrated by default in the Unity

editor like VR SDK compatibility and VR simulation. There are also other alternatives for teleportation with the use of certain scripts.

## 2.4 SteamVR and Oculus

Both SteamVR and Oculus are SDKs used in the project to make the game run properly with the selected VR headset. SteamVR is developed by Valve Corporation which is used with the HTC Vive and Oculus by Facebook Technologies which is used with Oculus Quest 2. These SDKs assists with the incoming inputs from the specific controllers and headsets when running the Unity game. Without the use of these external programs Unity would have a difficult time managing the inputs and detecting what configuration is right for the headset in use. There are no major difference from the two SDKs besides the fact that they are developed for different headsets. One of the few differences however is that SteamVR comes with a small preview of connections and a game view from the VR headset which the Oculus SDK does not support.

### 2.4.1 Unity Teams and Plastic SCM

GitHub is one of the most common development hubs out there and was highly considered for this project. Unity on the other hand has in recent years developed their own form of GitHub called Unity Teams. Unity Teams allows multiple people to work on the same project and make changes on the collaborate tab inside the project editor. Changes made to the project can then be published and picked up by other group members to pull into their own version. All the changes made will be stored which can be accessed if something goes wrong with the newer versions. Information about how many files and which ones that are modified are also displayed. Merge conflicts can occur if someone in the group have been modifying the same files. Unity will not overwrite any files if there is a merge conflict and it will need a manual approval by the developer. From there you can manually decide whether or not to merge your modified file with the remote branch. An integrated solution on the editor like this makes the cooperation between the group members less problematic which makes it an excellent choice for

the project.

During development of the game, Unity changed their policy and merged the collaborate tab in the editor with a new plugin called Plastic SCM. Since the newer version of Unity Teams made it impossible to work with the older collaboration method, the PRO licenses were upgraded for free with the Plastic SCM plugins. The new plugin works the same way as the previous, but gives more control and an even better overview of the changes and lines changed in the editor and scripts. While the plugin tab itself looks more advance and complex it features a lot more options inside the editor as opposed to the collaborate tab where some of the branching had to be done outside of the editor. Plastic SCM was thought to be troublesome to implement midway through the development period, but it was an easy process and proved to be an even better alternative after some practical use.

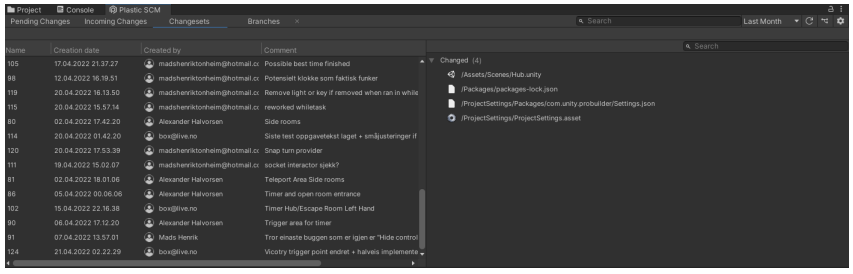


**Figure 2.14:** Plastic SCM plugin showing some of the changes and commits

## 2.5 Building the Game

To test the game or for finalizing the game it is required to create a game executable file. To do this there are several settings that must be adjusted to determine the appropriate build for the game. One of the first things that shows up when building the game is the scene selection. Unity needs to know what scenes to include and gives a list of available scenes to choose from. The order of selected scenes matter to build the game as intended. Platform must be selected and in our case either PC or Android depending on the headset. The Oculus Quest 2 is powered by a mobile unit and requires

Android as the selected platform when building the game. HTC Vive on the other hand does not support the Android mobile solution because a PC is required to even power up the HTC headset. Player settings can also be changed before publishing like resolution output and rendering settings to mention a few. Depending on your selected platform the build settings on the right side will change. As shown below Architecture, different builds and compression methods can be chosen here. The building process can vary depending on the size of the project. Upon build completion there will be an executable file on your selected storage location. All the game files like scripts, objects, etc can be accessed and launched by this .exe file. The game file itself will be considerably more compressed than the project file from Unity.

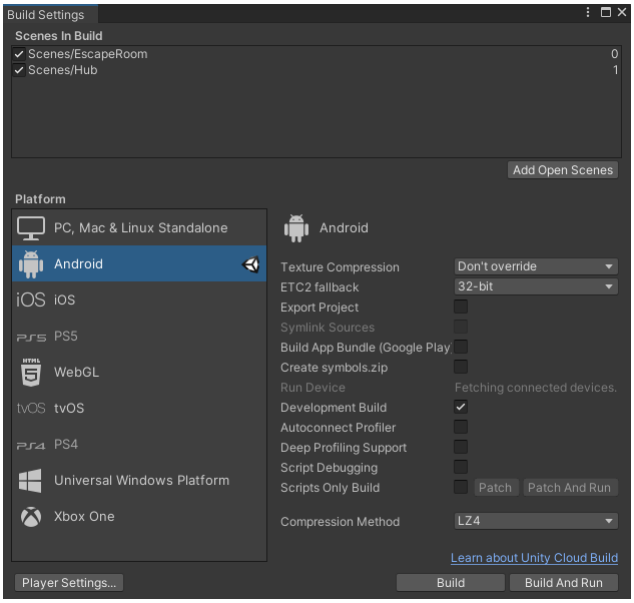


**Figure 2.15:** An overview of the various build settings before creating the game

# Chapter 3

# Game structure

The main goal of this bachelor thesis was to create an Edutainment game based on an Escape room experience with programming elements to it. The choice fell on a virtual reality experience to further immerse the player. The game itself contains one main hub area to practice and get used to the VR controls as well as a button to start the game. There is one main Escape Room experience with different puzzles to solve all based on programming. The chapter ahead will give a detailed developer description of the game layout and an explanation of the main components in the game.

## 3.1 Hub Area

The hub area is the central starting area for the player. This is placed in a different scene than the escape room, however it functions similarly and has the same controls. Different objects are available in this room to practice the controls and movement. A tutorial can be found in the area where the player can read and view the controls. There are interactable objects in the room to test how grabbing works. To get a feel for the grabbing a bowling lane with pins and a bowling ball is available in the room. This makes it more interesting for the player to get used to the controls.

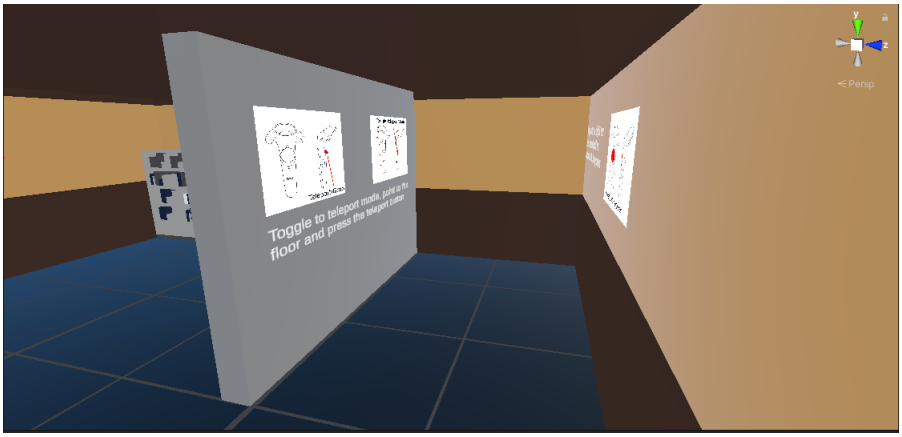


**Figure 3.1:** Controls information

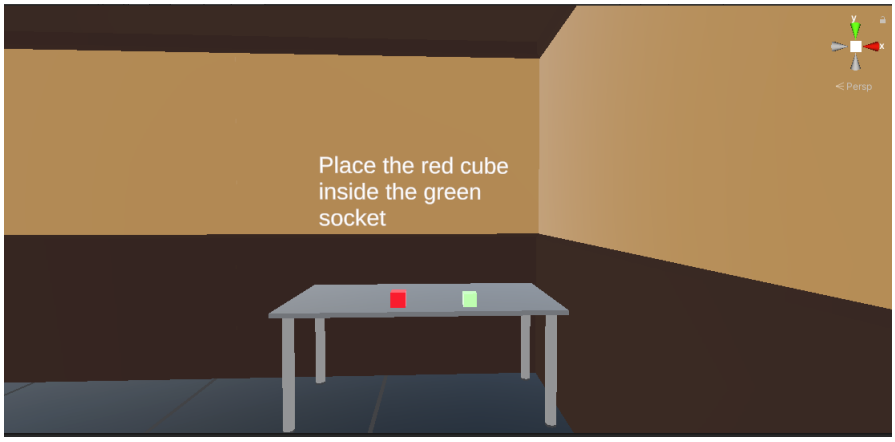


**Figure 3.2:** Socket tutorial

To move around in the game teleportation is used, as the set area for the VR-headset is not infinitely big. The same functionality is available in the hub to make sure that the controls gets represented the same way as in the actual game. On the wall to the right of the bowling lane, the best time for completing the escape room is displayed. This is a fun way to see how the users stack up against previous players. This can also be used as some sort of competition if the game is displayed on events like Open day. There is also a button for starting the game. Interaction with the start button can be done with both the right and left hand, and it does not require any but-

ton press on the controllers. The user can simply touch the desired button with their VR hand.

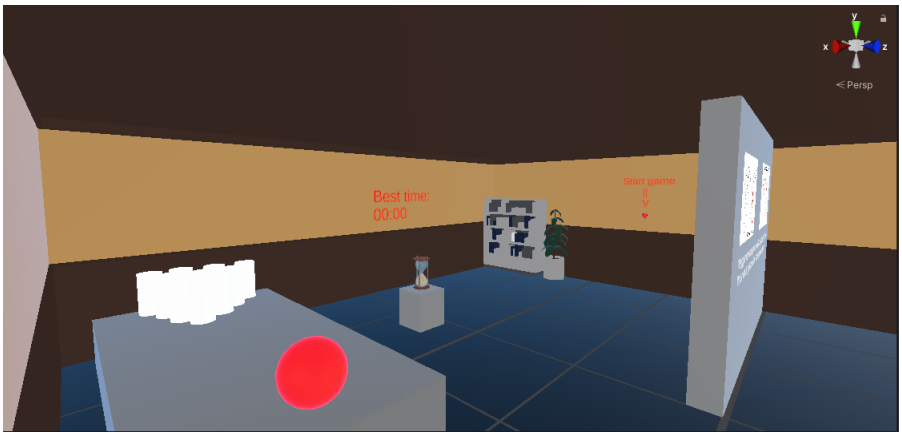


**Figure 3.3:** Bowling to test grabbing, best time and start button

If the player completes a game or chooses to leave the game through the hub button in the room, they will be sent back to the hub area. The best time can as mentioned be seen from the hub. During the game the user is timed, and when the game is done a check will be done to see whether or not the user was faster than the previous fastest time. If the player chooses to restart mid-game, the result will be invalid and the time will not be compared. Leaving sends the user back to the hub where they can choose to try again. The game does not have to be shut down for another person to play as the only thing saved in the game upon completion is the best time, which is saved as a static variable, and only changed if another player completes the escape room faster. Not having to shut down the game is important as this makes it easier to have multiple people try the game without having to spend time restarting. Considerations were also done for creating a simpler version of the escape room that could also be accessed via the hub. This version would function in the same way, but the text would be in pseudo code and easier for someone with no coding experience to read. This was ultimately not implemented because of multiple redesigns that were made in order to make a more balanced experience for everyone.

## 3.2 Escape Room

The escape room area is where the game starts. The player will need to explore the room in order to see the different tasks, and ultimately solve them to finish the game. The area is divided into multiple rooms each containing one puzzle which needs to be completed in order to proceed into the final part of the game. Each puzzle completed will reward the player with a key to use for the key sockets in the main room to open the final door. When all three keys are placed within the sockets, a door open with the final task which needs to be solved for the game to finish. A layout of the entire area will be displayed below.


**Figure 3.4:** Escape Room overview with all rooms

### 3.2.1 If Puzzle

This section of the game takes place in the main room. It contains multiple props and interactable objects. The puzzle can be found on the wall in the middle of the room, along with hints that indicates some puzzle pieces

can be found elsewhere in the room. The user needs to examine the wall to understand that a code snippet has to be inserted in the if statement to make the condition true. There are four code snippets available on the table next to the puzzle. These are interactable and can be placed within the condition of the if statement on the wall to check if it is true. None of the code snippets next to the wall are true, but there is a note indicating more code snippets are hidden in the room.

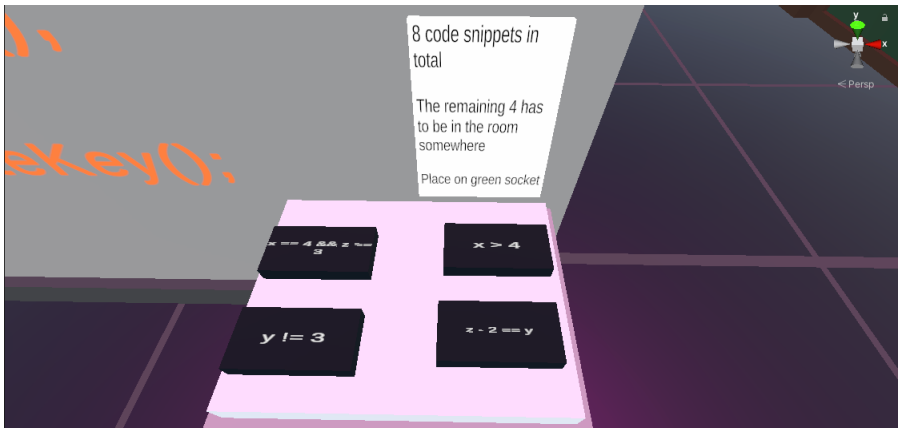


**Figure 3.5:** Some code snippets and a note

The puzzle room is filled with office like props that can be searched. A bookshelf can be searched and contains one of the code snippets which blends in with the environment. One of the other code snippets can be found in one of the corners along with some unused computer equipment that the player will need to look through. An office desk in the middle of the room will provide another piece. Upon finding the pieces for the puzzle the player will have to make a decision and either try to understand the provided code right away and check, or keep searching to gather all of them before trying to solve the challenge. The code snippet that makes the if true can be found in a pile of books lying on a table. In order to reveal the piece the user has to move/throw the books out of the away to find the piece. A picture of the puzzle is provided below.


**Figure 3.6:** The If puzzle

The idea of this puzzle is to make the players see how an if statement functions, and figure out what makes a condition evaluate to true. Looking at the first four snippets provided on the table can help the players to get a basic understanding of where to put them, as well as seeing some logical operators for comparing values. While the first snippets are quite easy to detect as false, the code pieces found in the room can be a bit trickier to understand and find. The item finding in the room was designed to create an entertaining and interactive experience during the game so that the puzzles themselves don't get stale while trying to solve them. Forcing the player to move through the room when searching will also require them to remember parts of the puzzle so that they can prevent time consuming trips back and forth to the task, if the newly found code snippets proves to be wrong.

### 3.2.2 Code interpretation puzzle

In the leftmost room a code interpretation puzzle can be found. Code is written on the wall to the left of the entry, and a table can be seen underneath. The user is supposed to figure out the value of a variable x. The value of this variable is then entered into a keypad on the front wall. After entering the four digit code on the keypad the user will press the green button to confirm. Wrong answers will display a "retry" message. If the user enters a number on accident, clicking the orange button will remove

the last digit. When the correct answer is given, "correct" is displayed on the keypad, and a key appears on the table underneath the code. This key is then supposed to be taken back to the main room. This is one of the three keys used to open the door for the final puzzle.

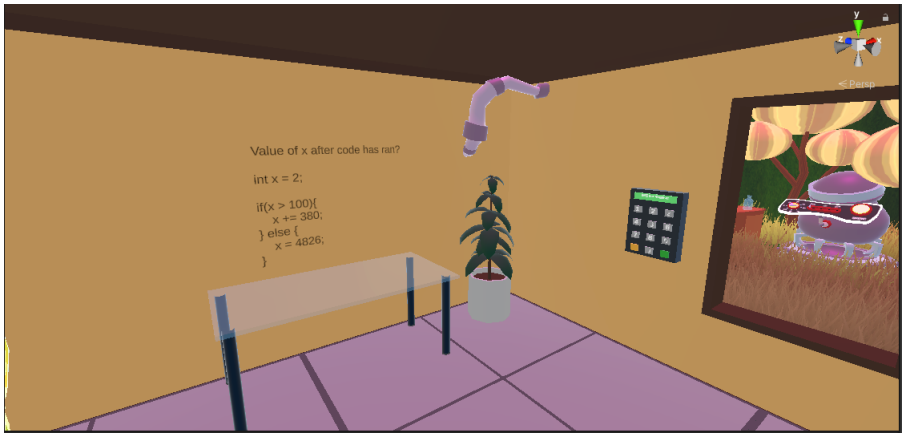


**Figure 3.7:** Picture of the code interpretation puzzle

The code starts of by creating a variable x. An if/else statement runs after this checking if x is greater than some value. Within both the if and the else portion of the statement the value of x is changed. Basing the code on a simple if/else statement should make the task somewhat easy for users with previous coding experience, while still making lesser experienced users able to solve the puzzle. Since there are only three times the value for x is set within the code, the user can also solve the puzzle through trial and error. This task is used to give the user an introduction to if/else statements as well as seeing how simple conditions work. If the user did the if-puzzle before starting this task, they will already have been introduced to conditions, making this task simpler to complete.

### 3.2.3 Loop Puzzle

In the rightmost room of the game the user will find themselves standing before a yellow cube with opening and closing curly brackets, and two sockets, one before the brackets and one within. There is also a button with the text "Run code:" next to it. A hint is also available on the wall beside

the device that highlights the importance of running the code repeatedly. Time can be saved by solving the challenge without searching the room. On the table and elsewhere in the room are blocks of code that the user can place within the sockets. To make it easier for the player the code that is supposed to be put before the curly brackets is green, and the code that is supposed to go within them is beige. Only correctly colored blocks can be placed within the sockets.

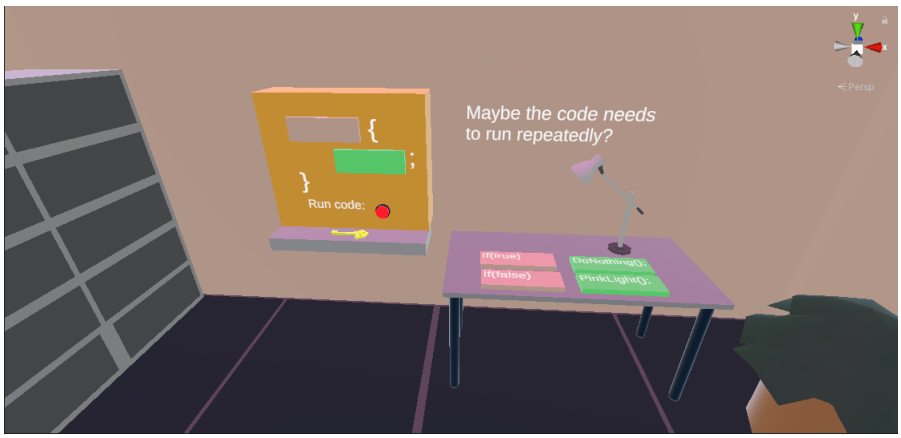


**Figure 3.8:** Picture of the loop task

This task is supposed to give the player an introduction to loops. Three code blocks can be placed before the curly brackets. These are "while(True)", "if(True)" and "if(False)". "if(False)" will do nothing when running the code. "if(True)" will run the code inside the curly brackets one time. If the correct code is placed within the brackets this will show the key for 1 second. When the correct piece, "while(True)" is placed in the socket the code will run infinitely. A common choice among players would be to use an if-statement, especially if the user has done the other tasks first. Realizing the key disappears after some time is then supposed to make the player try the while-loop block. This can teach a user that while-loops are conditional in the same way as if-statements, but that they run multiple times.

There are three blocks that can be placed within the curly brackets, same as for the outside. These are "ShowKey()", "DoNothing()" and "PinkLight()".

Using functions calls inside the brackets was decided, as this makes it easier to realize what the code is supposed to do. Seeing as the main objective of this task is to introduce the player to loops, a decision was made to make it so that the content within the loop could be easier for the player to understand. When both sockets on the yellow cube has been filled the user will click the button labeled "Run code: ". If the correct blocks are placed within the sockets the key will appear, without disappearing after one second, and can be taken back to the main room.

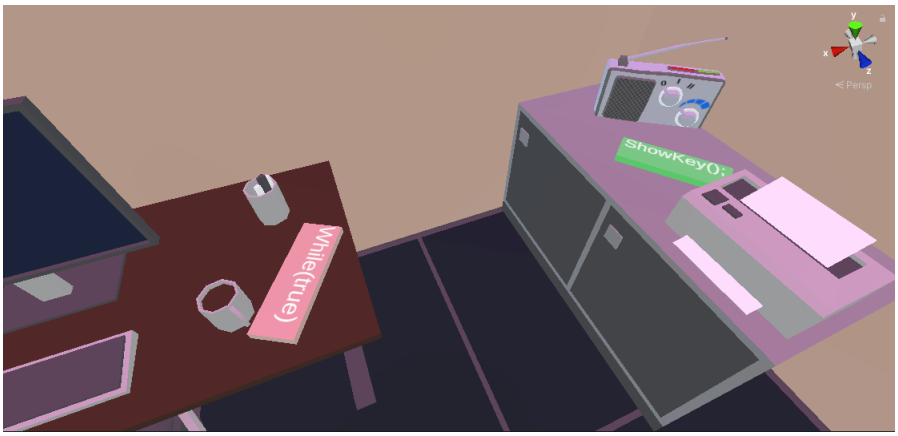


**Figure 3.9:** The correct pieces are placed somewhere else in the room

### 3.2.4 Final Puzzle

In the final part of the game the player should be able to place all three keys from the previous puzzle on the locked door in order to progress into the final test. Putting each key into the key sockets will make the doors in front possible to open. The puzzle will be based on the previous assignments in the room, but a timer will start counting down when the player enters the room. The doors will also close behind to make sure that it won't be possible to return to the previous tasks during this time. 5 minutes will be given to solve the remaining two-part task.

The first part consists of a while loop connected to the paintings on the wall on the other side of the room. Instead of making code snippets it

was decided to make this while task even more practical with the paintings being the main factor for solving it. The while loop is created to run a function that hides the second part required to progress. It runs while all the paintings are still attached to the wall and when the paintings are removed the while loop breaks and runs the last function, which reveals the second part. While the puzzle itself is easier than the previous while task it functions as a test to check if the player learned something during the other loop assignment. The most time consuming part of this puzzle would be to understand the connection between the paintings and the while loop. There will be a note/hint available that can highlight this connection.


**Figure 3.10:** The first part of the final puzzle

The second part of the puzzle sees the player creating a variable. A loop with an if-statement is running to check if a variable x is equal to 3. The user has to create this variable by placing multiple cubes on a wall containing operators, data types and values. If the user is struggling to find out how to create the variable they might start looking around the room. The table in the middle of the room contain two notes, one explaining what different data types do, and one for showing examples of variables being created. The second part is shown when the player finishes the first part. A red arrow points to the cubes that are used for creating the variables to make it easier for the user to realize what they are supposed to do. When the task is done correctly, a four-digit code appears for the user to type on a keypad. This opens a door and the game is done. The player can now return to the hub, where a new game can be started by another player.


**Figure 3.11:** The second part of the final puzzle

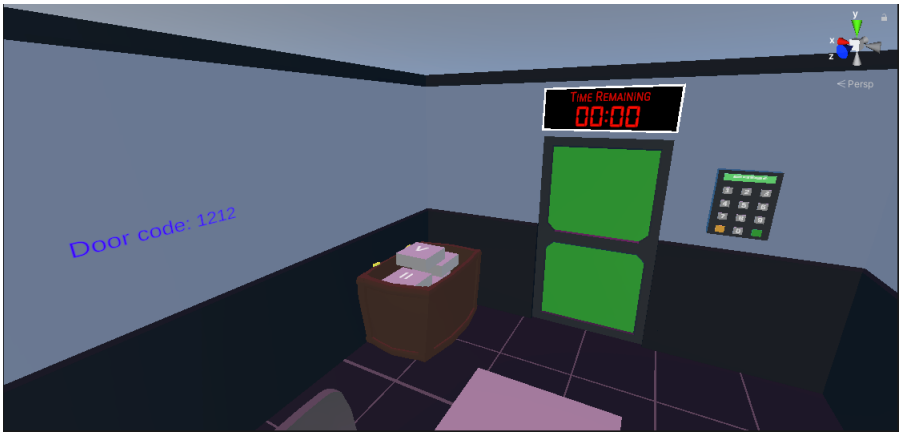


**Figure 3.12:** The four-digit code revealed on the wall

The thought behind the final timed puzzle room is to create an entertaining challenge in the end that represents some of the previous puzzle elements to see if the player learned something during the trial. It is meant to be almost like a final exam that is timed and needs to be done quickly. Since the final part of the escape room is timed it makes sense to have a bit less complicated puzzles. The first part represents the side room with the loop puzzle and requires the player to understand the loop and how to break it in order to show the next hint. The last part is meant to be more of a logical challenge where the goal is to create a variable. The puzzle is short, but

still educational while it highlights some important structural notes about coding in general.

### 3.2.5 Best Time

During the game the player will be able to see how much time they have used by looking at their left hand. When interacting with objects the timer turns invisible, to make looking at objects easier, but the timer is still counting. When leaving the Escape Room scene or finishing the game, the timer stops. If the player finished their time is compared to the current best time, and replaces the best time if the player was faster. If the countdown in the last part of the game reaches zero the timer will stop and not be stored. The best time display is only the best time for this session, meaning it resets when the game is completely restarted. This can be used to keep track of the best time on one event day, such as Open day at UiS. This is intended as a way for students to be able to compete with each other.

# Chapter 4

# Development

The chapter ahead will give an insight on how the game mechanics and puzzle elements were implemented. The development of the game first started with learning the basics of unity in addition to VR implementation. It was clear from the start that the game was to be centered around an escape room experience. Early concepts used in the final product were made from the very start, which have been built and shaped upon during the development process with input from testers, students and supervisor.

The first weeks of the project were dedicated to learning and creating the basics for the foundation of the game including setting up Unity collaboration. One of the most important objectives done in the starting phase was to set up VR for the player which included the movement and hand controllers. With the help of an integrated package/plugin called OpenXR it became possible to operate and test the game with multiple VR devices.

## 4.1 Rooms Design and Player

The first part of the escape room consists of three rooms. When this section is finished access is gained to a final room. There are multiple functionalities in these rooms which include, but are not limited to; audio, doors with

animations, sockets for keys gained from puzzles and a button for returning to the hub. The controllers used by the player also contain some functionality for toggling between teleporting and grabbing, animations and removing an invisible sphere used for pressing buttons on a keypad.

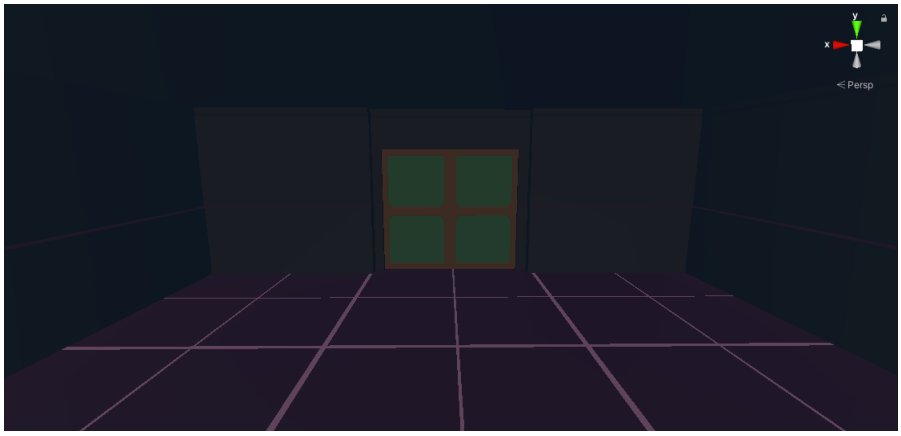


**Figure 4.1:** Where the player spawns in the escape room

### 4.1.1 Scenes and sounds

The first room the player is put in when starting the game contains a button to return the player to the hub. This button simply calls a function which loads the hub scene. Having this button in the room makes it so players who are not able to figure out solutions to the puzzle are able to give up. There is also ambient sound present in the game. Using a script the audio starts playing when the escape room scene is loaded. When all three keys are placed within the sockets this audio stops, and upon entering the room for the final puzzle another audio is played. Changing the audio for the final puzzle is done to create an atmosphere that this is the final obstacle in the way in order to complete the escape room.

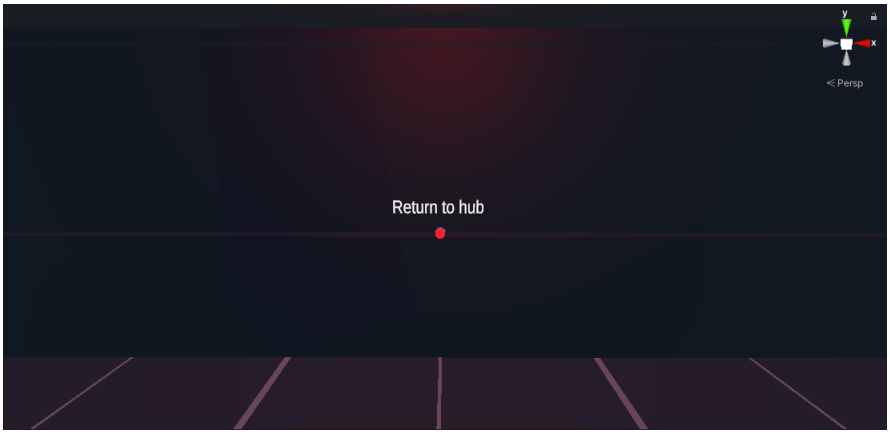


**Figure 4.2:** A button at the start of the escape room for the player to return to Hub

### 4.1.2 Player controls

Due to VR-controllers having a limited amount of buttons teleportation and grabbing are performed using the same button. The trigger button on the controllers are located on the back, and seems to be the most intuitive button for players to use. This is used for both teleportation and grabbing. Teleportation uses an XR Ray Interactor, so the user can point to where they want to teleport to. Grabbing is done with an XR Direct Interactor to keep interaction with objects more like in real life.

Teleportation was implemented as a way of movement due to the fact that the VR-area is not necessarily big enough for the player to move through the entire escape room, and this was a way of moving the player that provided freedom while not making the player motion sick. The player can also tap the touch pad on the controller to snap turn. This is useful as it can prevent the player from getting stuck in cables by repeatedly rotating the same way.

Early implementations of teleportation and grabbing used a combined three buttons so teleportation and grabbing could be done without toggling between modes. The user would use the select button on the side of the

controllers to interact with objects, and a combination of the 3D-axis on the controller together with the trigger button to teleport. Touching the 3D-axis would activate the ray interactor followed by pressing the trigger button to teleport to the desired location. This setup was deemed unintuitive and used too many buttons. The select button, atleast on the HTC Vive, is also not as easy to keep held down as the trigger button, which is an argument for using the trigger button for grabbing instead. Another implementation for teleport used the 3D-axis only. The user would tap and hold the 3D-axis to show the ray interactor, and let go when they wanted to teleport. This implementation made it so the user could not easily chose to not teleport when the action was started.

The current version before Open day on the 8th of March of using both grabbing and teleportation was not intuitive, and was not completely working as intended. Due to this a decision was made to simplify for the people who would test the game, and have the teleporter on one hand, and being able to grab with the other. The trigger button could now be used for both teleportation and grabbing, however a player could only teleport with their left hand, and only pick things up with their right. The controls seemed intuitive, and no user had problems with moving around the room nor did anyone have problems picking up objects. The controller setup could be left at this point, however being able to do both teleportation and grabbing with both hands was still desirable.

Changes were made to the same logic previously used to for having grabbing and teleportation on the same hand, but instead of using the 3D-axis to activate the teleportation ray, and deactivating it when the user lets go, the select button is used as a toggle for teleportation mode and grabbing mode. Pressing the select button once activates the teleportation ray and deactivates the grab interactable, and pressing it again reverses this. This seems to be a good way to keep teleportation and grabbing intuitive. To make it easier to separate between the two modes different models are used. When in teleportation mode the controller is displayed as a laser pointer, and in grab mode the controller is displayed as a hand.

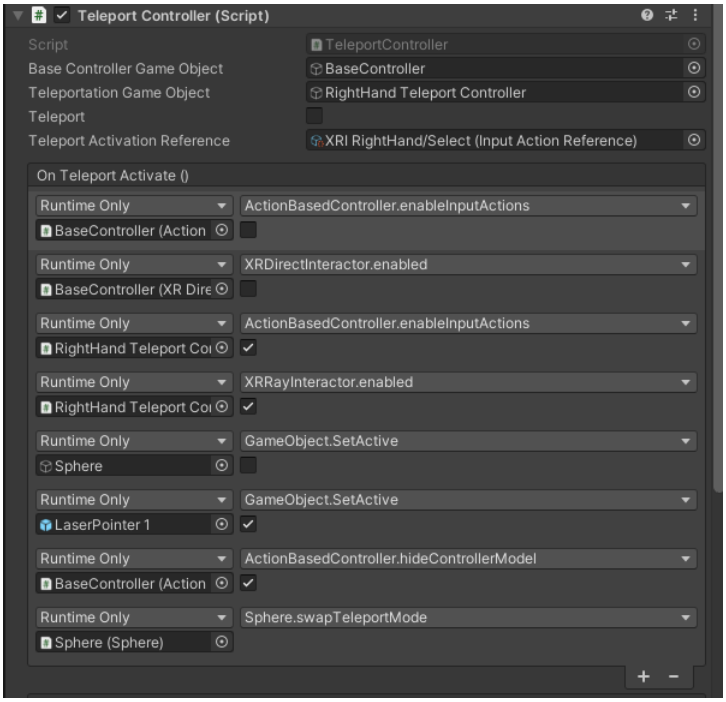


**Figure 4.3:** On Teleport Activate() is invoked by pressing the select button

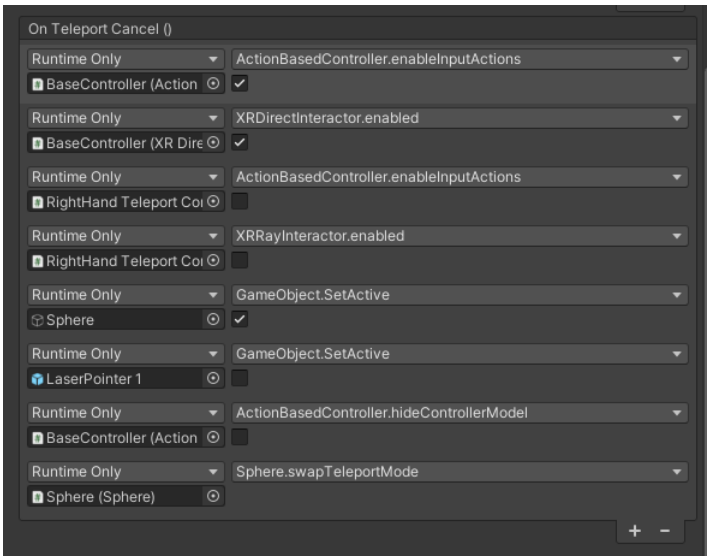


**Figure 4.4:** Pressing the select button again invokes On Teleport Cancel()

When in grab mode the hand model contains a script called ImprovedHandController that is responzible for the animations of the hand. When the user presses the trigger button the hand closes to a fist, and when letting go it opens. Depending on how far the trigger button is pressed in the hand closes or opens a small amount.

When just using XR origin and a main camera without additional components the player can put their head through walls to see through them. This is because the location of the headset is tracked, and there is no rigidbody on the headset for collisions. To fix this a rigidbody is added to the camera offset, as well as adding a body object under the camera offset. This mostly worked well for not being able to see through walls, but created its own set of problems that needed solving. Other objects in the room could now push the player. This could also be done with the players own hands. To prevent this from happening the body of the player was placed in a player-layer which only interacts with walls. This is done by using the layer collision matrix which can be found in project settings. There was also a problem where if the player teleported to doors that were in the closing animation they would be pushed and glide until they hit something. To stop the gliding a script was created which removed all velocity on the player when exiting collisions.

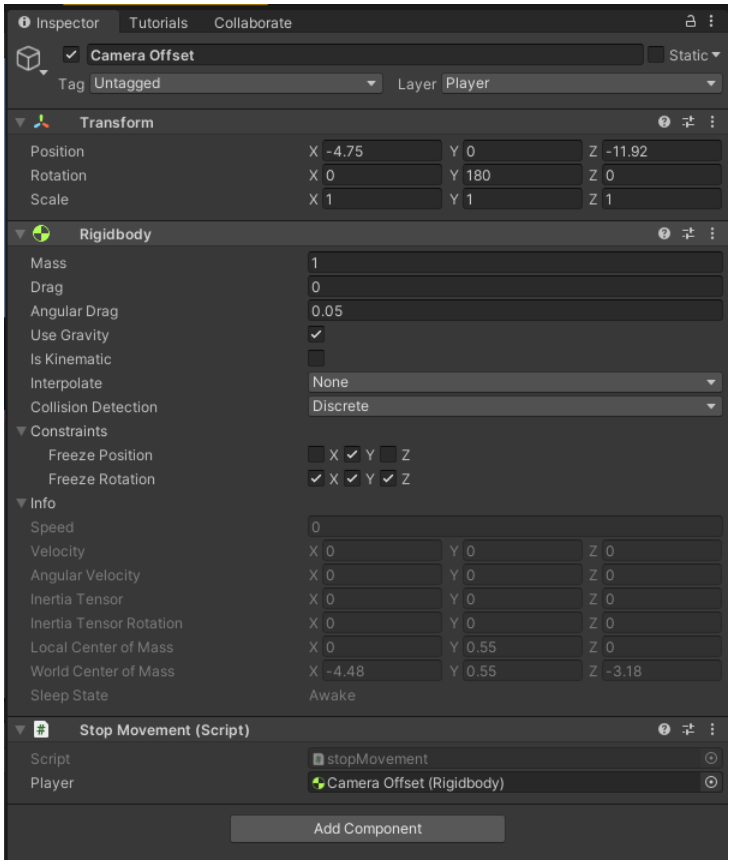


**Figure 4.5:** Rigidbody making the player collide and the Stop Movement script prevents gliding.

### 4.1.3 Events and Animations

In the early stages of development all tasks where contained in one room. This is the version that was tested during Open day. Feedback from students and observations made showed that people started one task and did not finish it before starting another. This was due to the fact that there where no proper borders between them. To make it simpler for users to separate the puzzles more different rooms where created for each task. The first three can be accessed at the same time, but are separated by doors. Stepping close to a door automatically opens it, and stepping away closes

it. The opening and closing is accomplished by using trigger areas.

One bug occurred due to the animations of these doors, where moving close to a door and then moving away instantly would keep the door open, and approaching the door would then close it. This seemed to be a problem with the time animations spent moving the doors, and that they could not be closed while in the process of opening. The fix for this was easily handled by delaying the event when exiting the trigger by 1 second. This makes it so the opening animation has time to finish before the doors start closing.

One of the doors in the main room is for the final task, and can not be opened until all other tasks are done and all keys are placed within a key socket. The collider for this final door works the same as for the other doors, but with a check for if all sockets are filled. This door also does not open again when entering the final room as the player is finished in the main room when starting the final task.

The keys that are to be placed within sockets to open the final door are in a key interaction layer, and only objects within this layer can be placed within these sockets. An integer is changed using the XR Socket Interactors interactor events for calling functions when items are placed within the sockets, and when items are taken out of the sockets. Upon placing objects in the socket the integer is incremented, and when removing them the integer is decremented. When the final key is placed the locked door to the final area can be opened.

## 4.2 If Puzzle

### 4.2.1 Development

The main goal of the if puzzle is to have the user place a code snippet inside the condition of an if-statement written on the wall. This is done by having a socket inside the parentheses that only takes in objects in one layer to

prevent placing other items inside the socket. Creating sockets is done by using "XR Socket Interactor" from the "XR Interaction Toolkit". Snippets can be found on a table next to the wall the statement is written on, but not every snippet is on the table. The user has to teleport around the room to look for extra code snippets to be placed within the condition. A decision was made to not have the user type out their own code, and use snippets instead, as a normal keyboard is not used in VR.

### 4.2.2 Implementation

This task was one of three ready for Open day at UiS on the 8th of March, however the way it was implemented was not the same as the finished product. Attempts at using XRInteractor.selectTarget was made, but this way of getting information about the object placed inside the socket was deprecated and could therefore not be used. A solution was not found for Open day, and using multiple sockets on top of each other was done as a workaround. Two layers for the snippets were created for this implementation, one for the correct snippet, and one for the rest. Placing a snippet inside a socket turned off the other socket by using interactor events within the XR Socket Interactor component. This was done to prevent having multiple snippets in the sockets at he same time. When an item was placed in the correct snippet, a key was enabled.

Later on the logic behind this task was changed. Two scripts are now being used, one manager script to enable and disable the key depending on if the condition within the if-statement is correct, and one script for getting information about the object inside the socket. The final implementation works by creating a variable "socket" and creating listeners for when objects are entering and exiting the socket. A check is done when an object enters the socket that compares the objects name with what has been defined as the correct snippet.

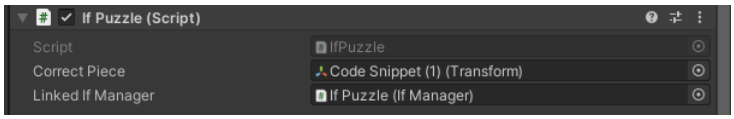


**Figure 4.6:** Setting the correct piece is done in a field in the inspector.

## 4.3 Code interpretation Puzzle

### 4.3.1 Development

The main goal of this puzzle is to have the player read some code to try and understand the value of a variable when the code is done running. This is another way of having the user interact with coding without typing out text themselves. The first ideas for this task came early in development when it was decided one of the tasks would have the user interpret some code. Ideas ranged from the code giving a hint as to where a key was located in a room, hints for other puzzles, or the solution for a puzzle. In the end it was decided that a variable would change values when the code was running, and that the user would enter the value of the variable onto a keypad. When the correct code is entered a key appears in the room.

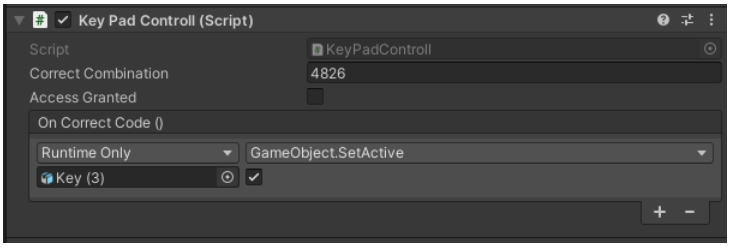


**Figure 4.7:** Entering the correct key code displays the key

### 4.3.2 Implementation

The only logic behind this puzzle is in the keypad, as the code the user has to interpret is static text on a wall. Three scripts are used for the keypad, these are a main control script, a feedback script to make it easier for the user to see that they have hit a button, and a detector script to detect when a button is being hit. An invisible sphere object is also placed on the controller, and is what interacts with the buttons. The detection script is placed on this sphere.

The most important script for the keypad is the detection script placed

on the sphere. When the scene is loaded display and keyPadControll is set by finding GameObjects in the scene with these tags. The rest of the functionality from this script comes when the sphere enters a collider that is set to trigger. Trigger makes it so a piece of code can be ran when objects enter or exit the collider. Only triggers with the tag "KeypadButton" are used to run the rest of the code. When hitting buttons with numbers on them, the number is typed into the display, given that four numbers are not already in the display. There are two buttons on the keypad which do not have numbers. One is green, the other is orange. Green is to confirm the entry, and checks if the numbers in the display are correct. Orange is a delete button, and removes one digit from the display. When confirming, a function, CheckIfCorrect(int combination), from the control script is ran. This function then sets accessGranted to the correct value. When access is granted to the user the key for this puzzle appears.

## 4.4 Loop Puzzle

This puzzle introduces the player to loops. From the previous tasks the user is already acquainted with if-statements, and might try to use one to solve this task at first. This will show a key for a brief second before it is disabled again. This is where the user is supposed to try using while-loops to run the code. The key will now be permanently showing. The goal is for the user to understand that a while-loop is like an if-statement that runs multiple times as long as the condition is true.

### 4.4.1 Development

One of the early ideas for a puzzle was something to do with loops. A section of code was to be repeated multiple times for something to happen. At first suggestions ideas like keeping a door open were made, however in the end it was decided that a key would appear when the correct block was added to the inside of the function.

### 4.4.2 Implementation

This puzzle uses two sockets, one for the block of code to be placed outside curly brackets, eg. "if(True)" or "if(False)", and one to be placed inside the brackets. There is one script connected to each of these sockets that function in a similar way. Both set up listeners for the socket, and create three variables for the different blocks that can be placed inside. Placing a piece inside the sockets will do function calls to the manager script depending on what piece was placed in the socket. The manager keeps track of what is placed in each of the sockets by using two variable called outsideState and insideState. These would be booleans if there were only two states, however due to the fact three blocks of code are able to be put in each socket the variable has an integer value ranging from 0 to 2. A button is placed beneath the sockets, and depending on the state of the variables different things happen when this button is pressed.

Similarly to the if puzzle a simpler version of this puzzle was created for Open day where there was only one correct and one incorrect block for each of the two sockets. This made it so there was no way for the code to be ran only once, so there was no logic for showing the key fora short duration. To create this later on a floating point variable is used and decremented for each frame by the time between frames. When this variable is 0 the key is disabled.

There is a block that can be placed within the brackets that is named "PinkLight()" which sets the lighting in the seen to be pink, depending on the block placed outside the brackets. This is mostly there as something fun that could happen if the player tries to brute force their way through the task, which can be done as there are only six blocks of code in total that can be placed in the sockets. This is done intentionally so users that are not able to figure out the solution can try multiple times to finish the task. When the key appears they will then be able to see the correct answer and try to understand it. This makes it so the game is always able to be finished, even though the user can not understand a puzzle.

## 4.5 Final Puzzle

### 4.5.1 Development

For the player to test if they have actually picked up on the concepts in the escape room a final puzzle is accessible when the player has placed all keys from the previous tasks. This puzzle is timed, and contains elements from the previous tasks. With no new concepts present in the task the player should be able to figure out how to finish this final puzzle within the time limit.

The first part of this tasks sees some code on the wall, similar to many of the previous puzzles. On the wall opposite to this code there are three paintings. The code contains a while loop that runs as long as there are paintings on the wall, and when the loop is broken the second part of the task appears. To stop the loop from running the player must take the paintings off the wall.

The second part of this puzzle has the player declaring a variable by placing blocks in the same sockets the painting were in. The blocks contain data types, operators, numbers and strings. On a wall there is a loop containing an if-statement with the condition that x has to be exactly equal to 3. The can decide if x is either a boolean, a string or an integer, set its value and has to chose if ==, < or = is used when declaring a variable. Upon creating the correct variable the text on the wall changes to reveal a four digit code.

When the second part of this puzzle is done the player can enter the four digit code onto a keypad to complete the final puzzle. This executes onWin() from Victory.cs. This opens a door to a victory area, the timer for this room stops counting down, and the timer for the best time also stops and compares the players time spent to previous players. The player can walk out to a field of straws. In this field is a pillar with a button for returning to the hub.

If the player does not finish the task in the given time a message with the words "Out of time" is displayed. The door for leaving to the field still opens as the game is now done and the player is supposed to return to the hub. When going through the door the out of time message disappears as to not block the sight for the player for too long.

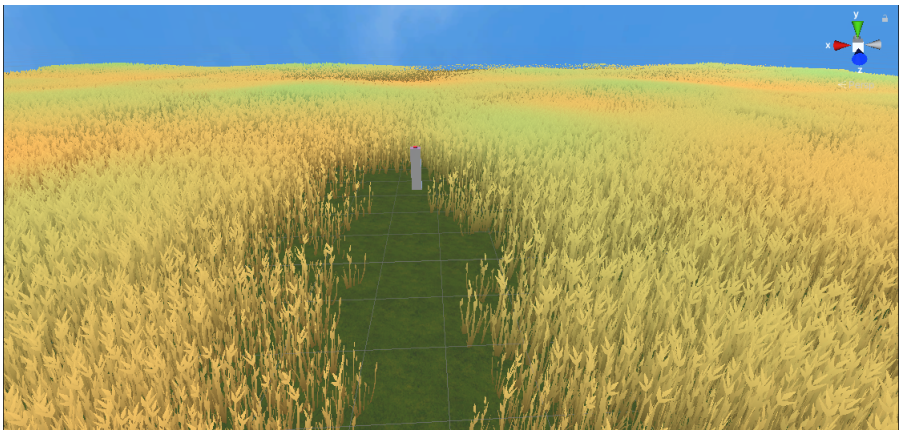

**Figure 4.8:** The field the player enters upon completing the game

### 4.5.2 Implementation

The sockets for this task function similarly to the ones from previous tasks in that they listen to when objects are placed and removed. When the pictures are removed from the wall a boolean variable in a managing script for this puzzle is set to false. If a picture is placed back on the wall this is again set to its starting value of true. When all sockets are empty the second part of the puzzle can begin.

The logic for creating a variable is mostly within the sockets that the painting from the previous part of this puzzle were in. Text is revealed when the first part is done with instructions for the second part of the puzzle. When placing objects within the sockets a check is done on whether or not they are the correct piece for this socket. When all sockets are filled with the correct object the four digit code is revealed to the player.

The keypad in this tasks functions the same as the one used for the Code Interpretation puzzle, however the code and what is done upon entering it is different. When the correct code is entered a function for when the game is won is invoked. Upon entering the correct code the exit door is opened. The two timers are also stopped by changing the value of a boolean in the timer and clock scripts.

When creating the keypad for this task some bugs started to occur. These where due to the fact the detection script for when a key was pressed used a function for finding objects with a specified tag. Seeing as the tags where the same at first only one keypad remained functional. There are multiple workaround for this without changing the original script too much. Having two instances of the same script, but using [SerializeField] or public variables to easily change the tags the script looks for seems to be the simplest. The tags on one of the keypads are changed to fit the second instance of the script. Other ways to go about fixing this problem could be to not use tags but instead reference specified objects set in serialized fields.

## 4.6 Hub Area

During Open day the controls had to be explained to the players, and although somewhat intuitive at the time, a decision was made that there needed to be a hub where the player could test the controls before starting the game. In the hub the player can try grabbing items by throwing a ball down a bowling lane, and teleporting around the room. Directions for how to do these things are placed within the hub. There is also some text on a wall indicating the current best time for completing the escape room, a fun way for players to see how they did compared to others that tried the escape room at events like Open day. Lastly a button is placed within the hub for the player to start the game.

### 4.6.1 Development

Thoughts of a hub were always present, however it was not until Open day it was really decided that it needed to be created. Restarting the game by completely closing it and opening it again quickly became old, and the ability for players to learn controls without someone telling them specifically what each button did had to be created. Mainly the hub works the same as the escape room, but without all the puzzles. The controls are the same at all times within the game.

### 4.6.2 Implementation

A separate scene was created for the hub. Swapping between scenes is easily done by using the "UnityEngine.SceneManagement" class. Pressing a button calls the function that loads a new scene, both from the escape room and from the hub. The button is similar to the one that was used for running code in the loop task. When loading scenes new instances of objects and scripts are created and the old are destroyed. This makes it so almost nothing is saved between the scenes. This is both good and bad as it makes it easy to start a completely new game, however information to be present in multiple scenes is not automatically saved.

The best time for the session is saved as a static variable. Static variables are the same for every instance. This makes it so changing the variable from the escape room also changes it in the hub, and that it stays changed every time the user returns to the hub.

When the user is doing the escape room a clock can be seen on its left hand. This indicates how much time it has used. If the user completes the escape room faster than someone else before in the same session, the bestTime static variable is changed, and the time shown as fastest in the hub is now the time this user spent.

# Chapter 5

# Discussion

## 5.1 Open day and user feedback

Developing a game can be challenging and requires a lot of testing. Understanding the difficulty when designing the tasks and puzzles for the escape room can be one of the most challenging parts about the project. It needs to fit the audience, otherwise it becomes either too easy and boring or too hard and impossible to complete for the right audience. Therefore, being able to get user feedback can be essential for fixing problems like difficulty or exposing bugs and also come up with new ideas.

On the 8th of March 2022 there was an opportunity for the game to be tested and receive feedback on the experience. This was at the Open Day event at UiS which is a yearly event. Open Day features many exciting stands and projects that high school students (vgs students) can visit and learn more about. It is a great opportunity to learn more about what the university has to offer. The game was targeted for high school students, so it was perfect for the occasion. A spot was reserved for the project where equipment could be set up for students to try the escape room. Notes were also taken during the event to make sure the feedback from the testing could be implemented or improved upon afterwards.

The escape room that was tested during the Open Day event did not represent the full experience, but 3 out of 4 puzzles were represented. At this time the game only contained one room with all the puzzles being located close to each other. This is the same area that contains the if puzzle in the final version. During the event there were several participants and testers that came to test and ask about the game. Some of the testers had experience with programming from before while others had less knowledge on the subject. Both the developers were present during the day to observe the results. After each turn with the game the participants got questions about how they liked the game, difficulty and what they thought could be improved upon. All the students that tried out the game seemed to enjoy it with only minor technical issues occurring.

### 5.1.1 Improvements and changes after Open Day

The feedback for the game was great and most of it was quite positive. Everyone completed the game, but some required a little assistance to get started with some of the puzzles. Some of the students with less experience with programming were having trouble understanding some of the puzzles, mostly the code interpretation and the if puzzle. To make it more suitable for everyone to understand, the if puzzle was remade to make it easier to understand. The mechanic of the task is still the same were you have to place code snippets in the if statement, but the variables are not as confusing and there are less variables to think about. There is also an additional hint to be found that explains the purpose of an if statement better.

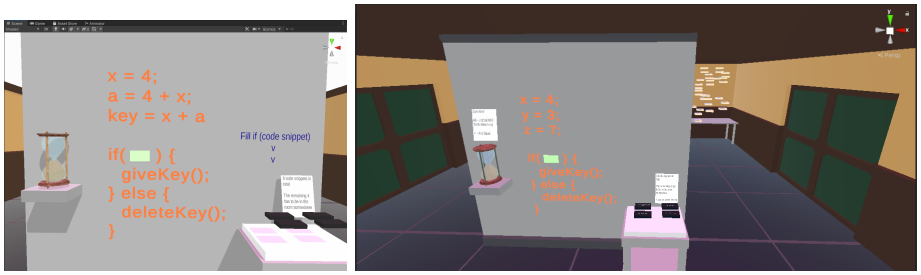


**Figure 5.1:** Before Open Day on the left

**Figure 5.2:** After Open Day on the right

One comment indicated that it was inconvenient not being able to grab and teleport with the same hand. The way the hands were designed at the time, mainly focused on easy controls with few buttons to press to make it less confusing. So the only buttons that needed to be pressed were the trigger buttons on the backside of the controller. Not being able to grab with both hands made it less intuitive than intended because the only option was to grab with the right hand. It was therefore one of the first things to be fixed after the event. The final version now uses both the trigger and the select button on the controller where select can be used to toggle between the teleport and grab function. To clarify the controls, a hub area was also created to make sure that players could get a short tutorial before starting the game. The hub works as a practice room where the players can test grab and teleport, and it also contains the "best time" feature, which was added later on. Another requested feature was the ability to add snap turn so that you can turn around easier without having to actually turn all the way around. Turning and looking a lot in the escape room is a central piece of the gameplay loop and can cause wires from the VR device to get in the way. So the most efficient way of solving the problem was to add a snapping feature that snaps the camera around 45 degrees to either side when pressing the touchpad or control stick. Last, but not least it was also observed and commented that the puzzles were kind of hard to find or commit to. All of the tasks available were present in the same room which caused several of the players to walk around and not committing to one task which makes the game feel overwhelming while also wasting time. After the observation a new layout design was thought out. The game now features the same main room as before, containing the if puzzle and the door to the final puzzle, but two side rooms have been added. One for the code interpretation and one for the loop assignment. Each room also has an automated door that opens once the player steps in front of it. This design choice helps separate the puzzles more from each other causing them to be easier detected and solved.

Another perk from the event was the opportunity to observe every player and their actions during the game. This made it possible to pinpoint some interesting bugs that otherwise might have been overlooked. One of the bugs was an unintended game mechanic with the teleport, force pulling. Force pull made it possible to quickly pull an object towards the player by holding down the teleport button and pointing the ray on an object. Force pull was a highly considered game mechanic to put in the actual game.

It was scrapped due to the fact that the game requires a lot of searching and object grabbing by hand. Having a ray interactor for grabbing would make it way less engaging to interact with the environment. This was fixed modifying the layer settings in Unity so that the teleport ray does not interact with the "grabbable" layer. The "grabbable" layer is placed on all the objects that are possible to grab. There were some occasions where the participants would try to interact with the keypad numbers while holding down the grab/trigger button. This makes it impossible to interact with the numbers because the sphere collision for the hands gets deactivated once a button is pressed. The bug had an easy fix and needed to stay active while the button was pressed. This was modified in the interactor events inspector on the right and left hand.

## 5.2 Performance Results

To get an idea of how well the game performs a performance analysis was done. VR games in general are more resource heavy than ordinary non-virtual games and usually requires high end computers to run. The analysis measures frames per second (FPS) during the game and gives an accurate description of where the game potentially struggles to keep up and receives inconsistent frame rate readings. Unity provides a built in profiling tool which can be accessed from the window tab under rendering. Optimal FPS readings for a virtual game would be atleast 60 FPS. Anything under 60 FPS can cause a choppy and inconsistent experience. The lowest acceptable readings would be 30 FPS, while the best experience could be achieved by measuring readings over 90 FPS which severely reduces the chances of getting motion sickness while playing.

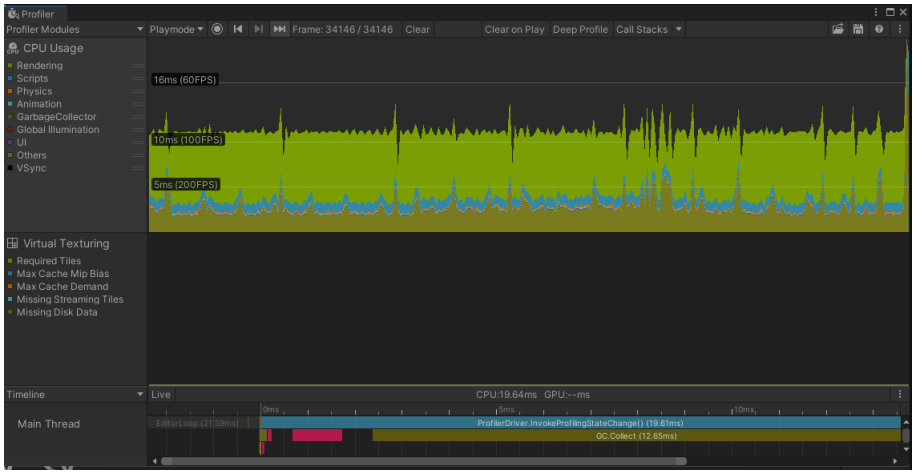


**Figure 5.3:** Performance test from the final part of the game (Final Puzzle and Outside area)

A performance test for the whole game was recorded, but since the FPS was stable for the most part during the whole experience it was decided to only include the final part of the game. The beginning and middle part stayed above 90/100 FPS during the test with the exception of a spike when loading a new scene. This is acceptable as nothing is being done by the player at this point. Even though the final part of the game ran well with mostly the same reading as before there were some larger spikes present in this part. As seen in the picture above the CPU usage spiked a bit during the end causing the frame rate to drop. Since the final room had two parts with more objects spawning in when the second part began it caused higher CPU usage. Upon completion the doors to the outside area was opened and the final most noticeable spike occurred. As the player moved through the grass field some larger performance drops were recorded due to the rendering of the grass as well as the swaying effect that came with it. Even with the drops the frame rate was still well above 60 FPS, and it was determined to be insignificant for the player experience. The spike all the way to the right from the picture above represents the scene changing when loading back into the hub and does not affect the user.

## 5.3 The State of the Game

Looking back at the assignment of the thesis it is clear that the main goal has been accomplished. There are several fun and engaging puzzles that are highly educational. Playing the game gives an introduction to some basic programming concepts while also making the player search for clues and pieces for the puzzles like a proper escape room experience. One of the hardest aspects of the development was the puzzle designs. They had to be relevant for the theme and represent some sort of similarity to actual programming. While the tasks themselves don't make you write any lines of code you have to understand the basic concepts in order to succeed. It was hard to know exactly how hard and difficult a task was going to be and at the same time not give too much information away, which could potentially ruin the point of the game. The intended target audience were clearly based on high school students to give them a taste of what the university has to offer. The project presented here succeeds well on these parts and presents a fine and balanced game experience.

To quickly summarize the puzzles and give a short description of their purpose, it can be said that the main room containing the if puzzle functions as an introduction to if-statements and conditions. Every task is made so that every student no matter the educational background can solve the puzzle through trial and error. The code interpretation introduces some new concepts including else-statements. The loop room creates some interesting choices for the player to make that triggers different events. The concept can be easily understood through trial and error and examining why and how the events happen depending on the input choice. The final puzzle room works as summary of sorts that consists of 2 parts. Both of them are easier than the other puzzle rooms themselves, but they still require some knowledge of the previous tasks. All the puzzles are aimed for people who could be interested in Computer Science and Engineering. It gives a small taste of what awaits in the coming subjects. The concepts presented can be related to several programming languages and therefore highly relevant for students who want to learn more.

### 5.3.1 Fun versus Educational designs

One of the main thoughts always present when developing a game like this is how to make the game educational, but at the same time create fun and interactive designs. Some of the puzzles like the ones requiring the player to find code snippets allows for interactions with the environment and active searching to find the correct pieces. The loop task can be used to make some interesting events in the room like changing the lighting. The final puzzle room offers the most variety and a short countdown timer to make the situation even more exciting. Animations for the hands when grabbing makes it feel more real while the counting timer on the hand can create competition among the participants to battle for the best time. Finding important items like the keys for the door or puzzle pieces for the tasks all contribute to the entertainment part of the game.

## 5.4 VR Equipment and Tool Choices

From the very beginning there were thoughts about using two VR devices, the HTC Vive and Oculus Quest 2. The reasoning behind this choice was to make sure that it was possible to run the game on multiple VR devices, without it being tied to one device only. Another reason was that it became possible to test the HTC Vive headset at the lab when working on the university and at the same time be able to work at home with the Oculus. Over the years Unity has also gotten updates including the ability to enable OpenXR which makes it possible to play the game with different headsets without too much trouble. Using two headsets during development also made it more efficient to work with. Both of the VR headsets had their benefits, but the Oculus Quest 2 definitely had the easiest setup because it was wireless and sensor free. Some issues were present when testing with both. There were occasions where the Oculus could interact with certain objects and the Vive could not causing the game to crash. There were other minor bugs with the controllers as well, but all of them were fixed after being discovered. Using two different devices proved to be very helpful in this regard, and to discover bugs that would otherwise not have been found using only one headset. It is therefore safe to say that multiple VR devices can be used to run the game without trouble.

Unity was the clear choice for the development of the project. It was recommended by previous theses assignments and also highly praised as one the best and most used game engines online. Unity also offered a wide variety of resources including integrated packages and an easy to use asset store. Plenty of support online were also available and a huge archive of Unity documentation on their main website. Even though the learning process took a while it became easy to organize and get used to because of the organized look of the program. The VR support for Unity is among the best available and is still as of today considered to be the best platform for VR creation. C# was also supported in Unity and became a perfect match since both of the developers had worked with C# earlier. Some knowledge about the language was used during the project, but there was also new knowledge to be learned about the scripting behaviour in the game engine. Using Unity to develop the game has been a positive experience for both and it was great for VR-development. Both group members came out wanting to learn more and create more after making the educational game for the project.

## 5.5 Future Development

There are plenty of opportunities for further development from improving features to creating completely new ones. The tasks are somewhat similar, and could be changed for more diverse puzzles. This would create more variation for the player, but if the puzzles are too different it might be harder to play through the game. There could also be multiple levels and difficulty. From the hub there could be different buttons for accessing different escape rooms. These could have different themes or the same as for the escape room that is already created. The same rooms could also be created with easier to understand text and more hints as a sort of easy mode for players with less experience coding. This was considered but ultimately dropped for the sake of time. When it comes to improvement of current features this is mainly on the player and the room, not necessarily the functionality of the tasks. An example of improvement on the player is collision. Colliding with walls creates a difference in where the players head is in game and in reality which might not be desirable.

# Chapter 6

# Conclusion

## 6.1 Individual Reflections

The final section of the thesis consists of some personal reflections from both of the developers where the main focus is to give personal thoughts on the collaboration and final product.

### 6.1.1 Alexander Pedersen Halvorsen

Coming into this project I had very limited knowledge about game development. I had previously only made simple games using pure programming, but never used any game engine to create bigger game projects before. The project to create an Educational game seemed like a fun opportunity to learn something new. The assignment itself was pretty open and creating a VR game wasn't a requirement. Debating whether or not to create a Virtual experience was hard since VR development seemed like a complicated choice for two beginners. After learning more about the development and experiencing the final product it was clear that VR was the right choice. Over the years Unity had developed more and more support for VR making it easier to deal with. Both me and Mads Henrik had previous knowledge and experience using the programming language C#, which was one of the

main reasons why Unity was chosen.

Teamwork has been quite important during the whole project. Helping each other on some of the biggest implementations and troubleshooting/testing the game was an important factor to how we were able to push it towards perfection. I am happy with the final product and think we reached the goal for the thesis. The game features a lot of fun factors as well as many educational puzzles. During my time developing this project i gradually became more fond of creating and scripting in Unity. Going forward I could definitely see my self creating more Unity projects as it was an exciting and fun experience. I am forever grateful for getting this opportunity to create such a unique experience in a thesis.

### 6.1.2 Mads Henrik Tonheim

I really enjoyed getting to experience creating a proper game. This is something I wanted to do for a really long time, and getting to do it for my thesis was a really fun experience. I had heard of Unity before, but had never tried it. Unity proved to make game development easier than I had previously imagined, and I will definitely try creating more games using the program. The first month was spent almost solely on learning Unity seeing as neither of the authors of this thesis had done any game development before. Starting a new project in Unity in the future will be easier and faster as I do not have to learn the basics again. Although VR proved to have its own set of challenges I am happy that we decided to create a VR game as VR seems to become more and more used and knowledge of it might prove useful in the future.

One regret I have is that GitHub probably should have been used from the start of the thesis, or more time should have been spent learning Plastic SCM to make collaboration more smooth. At times values put in serializable fields where deleted upon fetching a changeset that the other person had done. This created a lot of unnecessary problems that probably could have been avoided. Other than this problem the collaboration with my fellow writer was great, and I enjoyed the experience of working on a big project together with someone else.

# Bibliography


[1] HTC Corporation. Specs and details. https://www.vive.com/eu/product/vive/. Accessed: 2022-05-13.

[2] LLC Facebook Technologies. Develop for the quest platform. https://developer.oculus.com/quest/. Accessed: 2022-05-13.

[3] LLC Facebook Technologies. Oculus integration. https://assetstore.unity.com/packages/tools/integration/oculus-integration-82022#description. Accessed: 2022-05-13.

[4] Asset Store Originals. Snaps prototype | office. https://assetstore.unity.com/packages/3d/environments/snaps-prototype-office-137490#description. Accessed: 2022-05-13.

[5] Bridget Poetker. What is virtual reality (+3 types of vr experiences). https://learn.g2.com/virtual-reality. Accessed: 2022-05-13.

[6] Shapes. Nature starter kit 2. https://assetstore.unity.com/packages/3d/environments/nature-starter-kit-2-52977#description. Accessed: 2022-05-13.

[7] Aural Space. Free music for puzzle games. https://assetstore.unity.com/packages/audio/music/free-music-for-puzzle-games-152395#description. Accessed: 2022-05-13.

[8] Unity Technologies. Terrain sample asset pack. https://assetstore.unity.com/packages/3d/environments/landscapes/terrain-sample-asset-pack-145808#description. Accessed: 2022-05-13.

[9] Unity Technologies. Vr beginner: The escape room. https://assetstore.unity.com/packages/templates/tutorials/vr-beginner-the-escape-room-163264#description. Accessed: 2022-05-13.

[2] [1] [5] [3] [9] [4] [7] [8] [6]

# Appendix A

# Appendix A

## A.1 Youtube video of the game

Playthrough of the Escape room

## A.2 Setup

### A.2.1 Setting up Unity and SteamVR/Oculus

Unity has been used for development and can be downloaded through the Unity website along with the hub to manage projects and editor versions easily. The editor version used on this project was 2020.3.33f1

The project requires either SteamVR or the Oculus App in order to make the game work properly. HTC Vive requires SteamVR and Oculus Quest 2 requires the Oculus app. System requirements for both headsets will be listed below.

- Minimum requirements for HTC Vive
- GPU: Nvidia GeForce GTX 970, AMD Radeon R9 290 or better
- CPU: Intel Core i5-4590, AMD FX 8350 or better
- RAM: 4 GB or more
- Video Output: HDMI 1.4, DisplayPort 1.2 or newer
- USB Port: 1x USB 2.0 or better port
- Operating System: Windows 7 SP1, Windows 8.1 , Windows 10 or later

- Minimum Requirements for Oculus Quest 2
- Memory: 8 GB
- GPU: NVIDIA GeForce GTX 970
- Dedicated Video Memory: 3 GB
- CPU: Intel Core i5-4590
- File Size: 10 GB
- OS: Windows 10 or later

## A.3 Script explanation

### A.3.1 Controllers and body

**BodyCollider.cs** helps with player and wall colliding.
**ImprovedHandController.cs** is responsible for animating the hands.
**Sphere.cs** is used to turn the a sphere on the hands on and off when the user is in teleport mode and gripping something. This sphere is used to press buttons.
**TeleportController.cs** toggles between grab and teleport mode.
**gripActivate.cs** creates unity events that call on sphere.cs when gripping.

**stopMovement.cs** prevents the player from endlessly gliding if it gets pushed.

### A.3.2 Load scene

**Restart.cs** loads the hub or the escape room depending on which scene the player is in.

### A.3.3 Door events

**DoorAnimationEvent.cs** animates the doors in the escape room when the player enters a trigger.
**FinalDoor.cs** animates the door to the final puzzle when the player enters a trigger, if all keys are in the keyholes.
**ExitDoor.cs** opens the final door when the last puzzle is completed.

### A.3.4 Key hole

**keyMainScript.cs** keeps track of how many keys are in the key holes.

### A.3.5 If puzzle

**IfPuzzle.cs** sets up listeners and calls different functions in ifManager.cs depending on the piece that was placed within the socket is the correct piece.
**ifManager.cs** Turns on or of the key for this task depending on if the correct piece is placed within the socket.

### A.3.6 Keypad

**KeyDetector.cs** is placed on an invisible sphere on the hand and detects when the player touches a button on the keypad. This does different things depending on if a number, confirm or back is hit.
**KeyFeedback.cs** shows the player when they hit a button by turning it green. A sound could also be played.
**KeyPadControll.cs** checks if the code entered on the keypad and invokes a unity event if the code is correct.

### A.3.7 Loop task

**ButtonVR.cs** Invokes unity events when entering or exiting a trigger on the button.
**InsidePiece.cs** Checks what piece is placed inside the loop and either shows key, turns light pink or does nothing.
**OutsidePiece.cs** Checks what piece is placed outside the loop and either runs what is inside for 1 second, forever or not at all.
**LoopManager.cs** Manages the other scripts for this task and controlls the main logic.

### A.3.8 Final puzzle room

**Timer.cs** starts a timer that counts down when the player enters the final puzzle room. If the player finishes the puzzle the timer stops. If the timer ends before the player finishes the game is done.
**Victory.cs** calls functions for when the game is done from Clock.cs, Timer.cs and ExitDoor.cs
**finalPuzzleSockets.cs** sets up listeners for the sockets in the final puzzle and calls different functions in finalPuzzle.cs
**finalPuzzle.cs** hides and shows the different parts of the puzzle and keeps track of how much has been done on the different parts.
**inFinalRoom.cs** calls FinalDoor.inside(); when the player enters the final puzzle room to make the door stay closed.

### A.3.9 Time

**BestTime.cs** keeps track of this sessions best time and displays it in the hub.
**Clock.cs** keeps track of the time the player spent in the escape room and calls on a function in BestTime when the game is done.

### A.3.10 Audio

**AudioTrigger.cs** Starts audio when the player enters the escape room, and stops it when all keys are placed in the key holes.

## A.4 Source code

GitHub repository

All .cs files can be found under Assets/Scripts