UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Shumpei Morimoto

Accessible VR World Building

Master's Thesis (30 ECTS)

Supervisor(s): Ulrich Norbisrath, PhD
Peeter Nieler, CEO, Criffin

Tartu 2022

Accessible VR World Building

Abstract:

This thesis describes the design and development process of a VR video game called *VR World Constructor*. The main goal of the game is to provide an immersive world-building game that attracts people and creates a base development for the further metaverse project. The thesis starts with an overview and analysis of similar software and previous techniques to get the idea for developing an interesting VR game. The thesis continues with the design and implementation of components in the game using Unity and Oculus Quest 2. The prototype of the game was usability tested by users to find out issues and bugs in the gameplay and to get feedback to improve the game and add new features. The left development to be required is also described for the future project.

Keywords:

VR, Worldbuilding, Unity

CERCS: P170 - arvutiteadus, arvanalüüs, süsteemid, kontroll

Hõlpsalt loodav virtuaalmaailm

Lühikokkuvõte:

Selles lõputöös kirjeldatakse virtuaalreaalsus-videomängu *VR World Constructor* kujundamis- ja arendamisprotsessi. Mängu peaeesmärk on pakkuda paeluvat maailmaloomise kogemust, mis tõmbab inimesi ligi ja rajab aluse hiljem arendatavale metaversumiprojektile. Lõputöö alguses on toodud ülevaade ja analüüs sarnasest tarkvarast ning varem kasutatud meetoditest, et tekiks idee, kuidas arendada huvitavat virtuaalreaalsusmängu. Seejärel kirjeldatakse mängu kujundus- ja juurutuselemente, mis põhinevad Unityl ja Oculus Quest 2-l. Mängu prototüüpi katsetasid mängijad, et selgitada välja mänguelamusega seotud probleemid ning saada tagasisidet selle kohta, kuidas mängu täiustada ja milliseid uusi funktsioone lisada. Kirjeldatakse ka seda, kuidas tulevast projekti edasi arendada.

Võtmesõnad:

virtuaalreaalsus, maailmaloomine, Unity

CERCS: 170 - Computer science, numerical analysis, systems, control

Table of Contents

1 Introduction

For the past five years, several consumer virtual reality (VR) headsets have been released, such as HTC Vive and Oculus (now Meta) Quest One and Two. These affordable headsets encouraged developers to create many VR games.

The advantage of VR games is that they significantly impact the player's immersion compared to classical computer games. VR games have a 360-degree view where the user can

Figure 1. Snapshot of *VR World Constructor*

see and explore. Also, VR gameplay is designed to be more intuitive than classical computer games to feel the reality and increase immersion. VR games use hand and head movement to play the game, while classical computer games use buttons and joysticks. These experiences will help the user to believe the fictional world and feel as if they are other characters in the VR world.

This thesis describes the design and the development of our VR city-building game, *VR World Constructor*, using a VR headset (Oculus Quest 2) and Unity. This game is about building your city from a bird's view and exploring the city you made. In the exploring game mode, you will be one of the city's residents and explore it as if you live in it.

However, when developing a VR game, worldbuilding is necessary not to break the immersion in the gameplay and feel the reality of the game.

1.1 Worldbuilding

Worldbuilding[1] is a creation of an entirely new fictional world. It is a crucial phase that connects the audience visually and emotionally. Worldbuilding is used in novels, movies, games, mangas, or VR experiences to make the audience believe as if that fictional world exists. In the virtual world, we need to make sure that all the essential features are placed for the player to feel the presence of the world and not break the immersion. For example, when you want to develop a VR zombie game, you have to think about what will make people believable to the world. The sound and movement of zombies will be the world's atmosphere.

1.2 Motivation

One of the advantages of VR games will be realizing the player's imagination. The world-building game will increase the player's immersion, and at the same time, people will enjoy expressing their thoughts by building their unique world. This game will make people enjoy creating and describing their imagination through gameplay. Moreover, as a bigger sight of this project, we will be adding a multiplayer feature and other game modes to the world so that the game will act as a metaverse, and people will gather in the game and take conversations while playing it. This thesis will be the first phase of developing the project's base, an intuitive world-building game. For the project, Peeter (CEO of Criffin) will be supporting me.

Criffin[2] is an Estonian company that is active in research and development in Virtual Reality. The project was developed with the support of Criffin, and all the assets used in the game were provided by Criffin. My supervisors (Ulrich and Peeter) are interested in educational aspects of VR game and therefore, they are both supporting the intuitive world-building game as a first step of the project.

[1] https://en.wikipedia.org/wiki/Worldbuilding
[2] https://criffin.com/

2 Previous Work

In this chapter I am looking for previous work related to our VR game to find the requirements that are necessary to make a valuable and useful product. This chapter will analyze the earlier work, the weakness and strengths of their respective products, and think about the best way to implement my game.

2.1 Similar Software

This section will discuss similar software that has been previously published and compare the difference in the systems and design of that software. Since we are creating a VR game, I focused on analyzing existing VR games.

2.1.1 Tiny Town VR

Tiny Town VR[4] is a casual world-building game with VR headsets. The player will be in a bird view during the play and place all kinds of objects (e.g., cars, buildings, people, animals) in the game scene to create your town. The player can also take photos after making their original city. Our work is taking a lot of inspiration from this game, especially the city-building aspect. However, the most significant difference from my game is that *Tiny Town VR* only has building gameplay and does not explore it. Our game has a game mode for exploring the created city. This feature will make the VR experience more enjoyable and immersive.

[4] https://www.tinytownvr.com/

Figure 2. Snapshot of *Tiny Town VR*

2.1.2 Minecraft VR

[5]

Figure 3. Snapshot of *Minecraft VR*

Minecraft[6] is a sandbox video game where you can build your imagination with blocks. The game content is about survival and adventure in the open world, not just building with blocks. The game was developed and released in 2011, and the game is still being updated

[5] https://www.youtube.com/watch?v=pjmzMOJedAk
[6] https://www.minecraft.net/en-us/vr

today. *Minecraft* is popular among all kinds of people. The content of *Minecraft* is related to our game since it has both building and exploring aspects. However, since *Minecraft* was first made on a classical computer and the VR system was supported later, there are some uncomfortable parts to playing. For example, in *Minecraft*, the player has to move around a lot for building and exploring. This could easily cause motion sickness. Motion sickness is a significant problem in VR games to break the immersion. One of our VR games concept will reduce motion sickness and make the player maximize the immersion in the game. Therefore, our game handles better locomotion techniques to succeed in these goals than *Minecraft VR.*

2.1.3 Bricks VR[7]

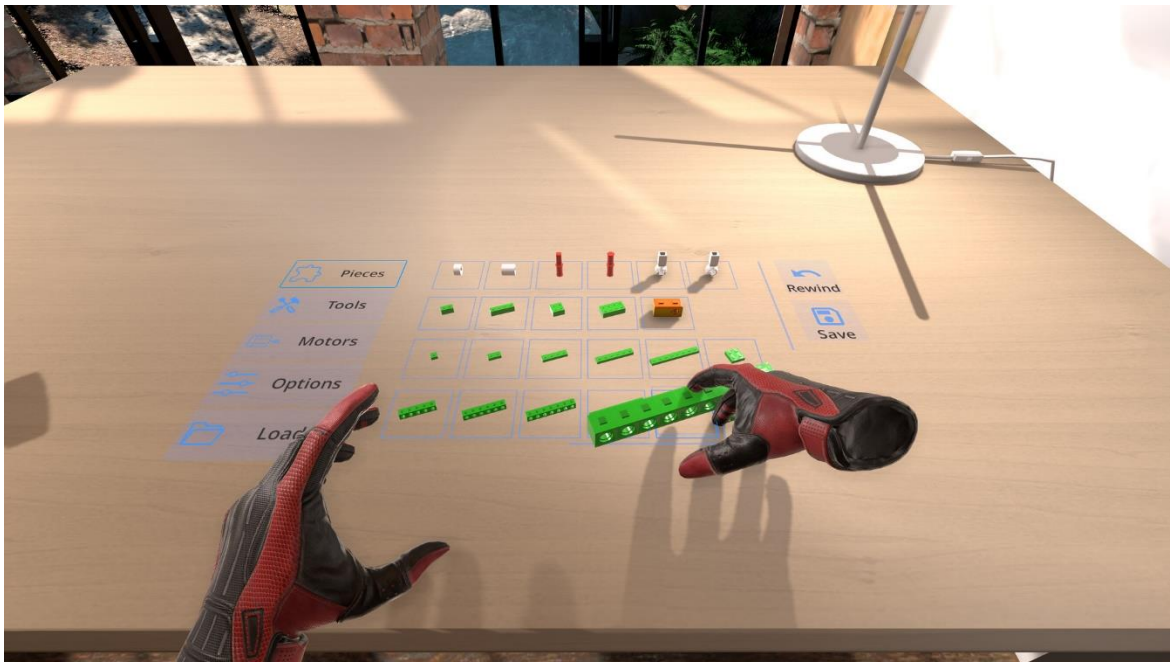

Figure 4. Snapshot of *Bricks VR*

Bricks VR[8] is a multiplayer VR building sandbox game. In the game, the player will see all kinds of sizes and shapes of *LEGO* blocks, and the player can create anything using those blocks. This game is more basically designed than ours since all the items are blocks, so the main gameplay will be making complicated things and not scaled things such as a whole city or town.

[7] https://store.steampowered.com/app/1665570/Brickbuilder_VR/
[8] https://bricksvr.com/

2.1.4 Arkio

Arkio[9] is a similar software recently released for VR from Arkio Ehf. on March 11, 2021. *Arkio* is a collaborative design tool in VR to sketch and model buildings intuitively. Users can sketch buildings and interiors and the entire city with other people. It can also change between bird view and ground and can be sketched from both viewpoints. This is not only possible in the game, but it also has something called mixed reality to blend the physical environment and virtual environment. Additionally, *Arkio* has cross-platform collaborations, and it can be used on desktops, tablets, and phones. This is definitly something worth considering as an extension to *VR World Constructor*.

There are a lot of parts to gain impressions and thoughts from this software since it covers a lot of things we want to implement in the future. However, this software has more aspects for designing and modeling buildings. Our game has more gaming aspects where people will enjoy the VR environment.

Figure 5. Snapshot of Arkio

2.2 VR Game Locomotion

[9] https://www.arkio.is/

In this section, we will be talking about previous locomotion techniques used for other VR software. The locomotion system is one of the essential features to think about when designing a VR game. Unlike classical computer games, VR can easily cause motion sickness when playing a game. Motion sickness will break the immersion of the world, and the user will not be able to play the game for a long time. Also, locomotion is the primary system when playing a 3D game, so it has to be intuitive and straightforward so that the player will not be confused about how to move around. Therefore, we need to carefully consider which locomotion type will be the best for our game.

2.2.1 Joystick Locomotion

The player can use the controller joysticks to move around and rotate. This is the simplest way since most console games use a joystick locomotion system for movement. People found the controller-based locomotion to be easy to master due to their previous experience with classical consumer games. This effortless locomotion style makes people focus on the gameplay and get immersed in the VR environment quickly. However, this type of interaction is comfortable to get used to, but it will also cause motion sickness [1]. This type of locomotion would be better to implement to our Building Mode locomotion since it is a most common locomotion in game so most of the people will be used to using it (see section 4.5.1).

2.2.2 Teleportation

Teleportation is the unique locomotion system where the user can change its place by pointing the controller to the desired location, showing where to jump. Users can teleport to the pointed location. This is also easy to use, and people get used to the system fast. It also reduces motion sickness because the locomotion happens when teleporting, so the user will always stand still when looking around. While teleportation is a good technique for moving with low-motion illness, we must be careful when implementing it in our game. Teleportation is a noncontinuous movement and reduces spatial awareness so that it might break the

immersion of the VR experiences. Also, people can quickly get tired due to the visual jumping effect [1]. In our game, both game modes (Exploring and Building) needs a continuous

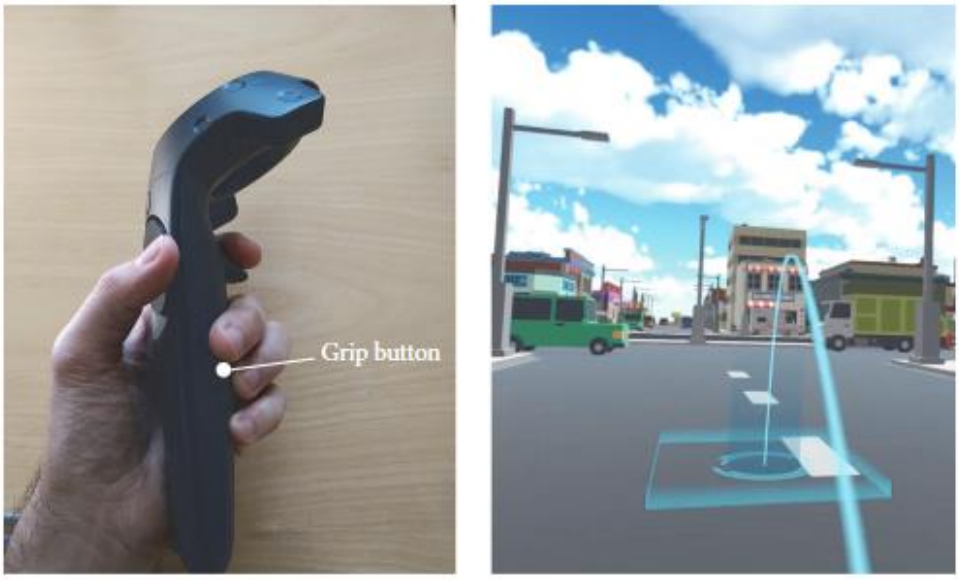

Figure 6. Interacting with the HTC Vive controller for the teleportation technique (left) and a view of the virtual environment (right).

locomotion. Therefore, we did not implement this type of locomotion to our game. However, there are lots of existing solution for this locomotion so if we need to implement it, there will be any problems.

2.2.3 Walking-In-Place

Walking in place is also a unique way of locomotion for VR games in which the movement is due to the direction of your limbs. The locomotion speed depends on the user's speed of their actual body. This locomotion is close to the movement of humans, so it will reduce motion sickness compared to the joystick locomotion. It is a continuous movement, so that it will be more immersive than teleportation, but it might make people tired from using their body to move, and some people think they might bump into physical objects and get scared. On the other side, it makes people fun and increases the immersion of the VR experience [2]. This is the perfect locomotion type for our Exploring Mode (see section 4.5.2). In the mode, people will walking in the city and walking-in-place locomotion will increase the immersion.

Figure 7. A user testing the walking-in-place technique.

2.2.4 Out-of-Body Locomotion

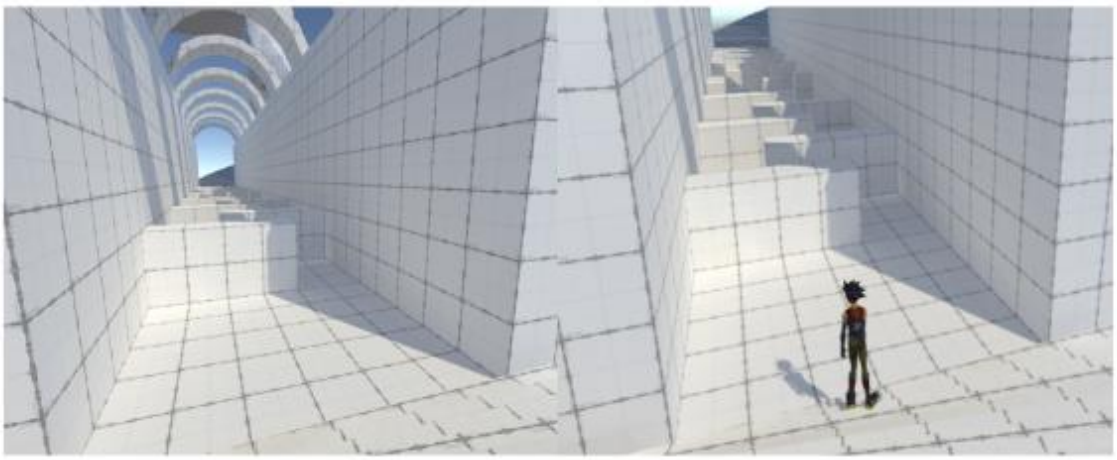

Figure 8. First-person view (left) and third-person view (right) including camera position that shows user's avatar.

This type of locomotion is similar to teleportation, except it also has a third-person camera to see the VR body. The VR body can be moved continuously, and the user can change their first/third-person view. In this way, it reduces motion sickness, but at the same time, the player can feel immersed in the world. While the first-person view can be more accurate to the interactions, the third-person view provides spatial awareness [2]. As I discussed on teleportation, noncontinuous locomotion is not the type of locomotion we need for our game so we did not implement this locomotion to our game.

2.2.5 Grab Locomotion

Grab locomotion can be done by pushing the trigger button on the controller and moving as the user is grabbing the world and moving others. This way, the movement is happening around, and the user is not moving in their sight, reducing the motion sickness [3]. Precise movement is necessary to our games Building Mode so we implemented this locomotion to our game (see section 4.5.3).

2.2.6 Climbing

This is intuitive locomotion used for climbing ladders vertically or moving ropes horizontally with their hands. This locomotion is a type of whole-body interaction. For example, when climbing ladders, the user has to grab and hold the target rung, step onto the first rung from the bottom, etc. By repeating this movement, the user can climb the ladder. When moving ropes with their hands, the user will grab their hands alternatively to move horizontally [4]. The prototype of our game did not had a gameplay of climbing ladders or move with ropes so we did not implement on our game. However, on the later project, adding this

Figure 9. Depiction of hand and foot interaction while climbing the ladder (left) and output in VR (right).

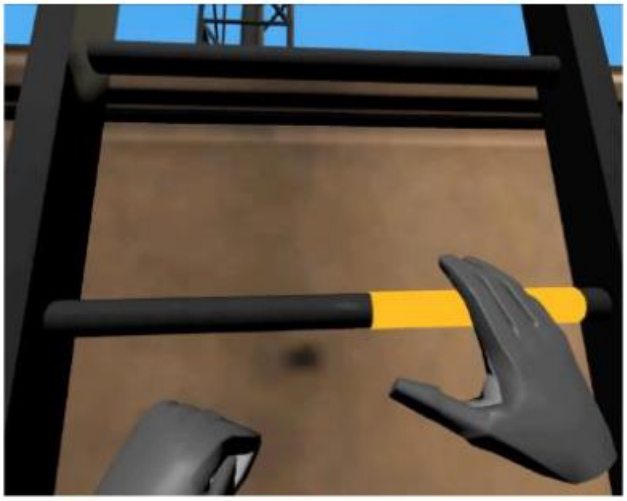

Figure 10. User's hands interacting with ladder in VR. The Orange bar on the rung is for visual cue.

locomotion will increase the immersion very well. There might some difficulties on implementing this locomotion because the prefabs we have provided from Criffin is not a complicated model so we need to modify the 3D model to be able to snap our hands to each part of the prefab.

3 Implementation

In this section, I will be talking about the developed environment and all the implementation of our game. First of all, we will be talking about the environment we used to create the game, and after that, we will be talking about the actual implementation and design we have done to our game.

3.1 Unity

Unity[10] is a 2D/3D game engine used by many developers worldwide. Unity deals with multiplatform for developing games such as Windows, Mac, Linux, iOS, and Android. It can be played on game platforms such as Playstation, Xbox, and Wii U. Unity supports JavaScript, C#, and Boo for developing the games. Also, it has an asset store that has all kinds of 3D assets and animation tools. There are other game engines such as Unreal Engine or Godot. However, I was most familiar with Unity and Criffin was also more familiar with it therefore, we implemented all the features with Unity.

3.2 Oculus Quest 2

Oculus Quest 2[11] (Quest 2) is a VR headset released in 2020 with the price of $300 from Facebook Technologies, Oculus. It is the latest VR headset from Oculus, and it is powerful

Figure 11. Oculus Quest 2

[10] https://unity.com/
[11] https://store.facebook.com/quest/products/quest-2/

enough to develop and play most VR games. Quest 2 has a high resolution (1832x1920 for each eye) with excellent VR picture quality. Oculus provides a development tool for developing a VR game using the oculus quest in Unity called Oculus Integration.

3.3 Oculus Integration

For this thesis development, we were planning to use only Quest 2, so we used Oculus Integration for our product. Oculus Integration is a development toolkit provided by Oculus for making VR software using Unity. It supports lots of features like direct grabbing, distant grabbing, or locomotion and it is also easier to learn and implement. However, this toolkit is only for development in Oculus Quest, but it would be better to switch to the XR interaction toolkit on the later project.

3.4 XR Interaction Toolkit

The XR interaction toolkit will better develop VR software on multiple platforms. XR interaction toolkit brings all types of controllers and VR headsets under a standard programming structure. Therefore, when we expand our project to a more significant project, for example, adding multiplayer features, we should switch it to the XR interaction toolkit.

3.5 Locomotion Used in Game

We have two types of game modes Exploring/Building in our game, so we need to use different locomotion systems for each game mode. Therefore, we propose three intuitive and straightforward locomotion types: Joystick locomotion, waking-in-place, and grab locomotion. All movement is designed intuitive and straightforwardly to reduce motion sickness without breaking the immersion.

3.5.1 Joystick Locomotion

This type of locomotion was explained in section 3.2.1, and it is used for Exploring Mode. People will be simply walking around the city they build in the Exploring Mode, so it will be helpful if the locomotion is simple enough to get used to so the user will not be confused. Joystick locomotion is the most known way for users to be comfortable [5].

The player can move in the direction they lean their left controller joystick.

3.5.2 Walking-In-Place

This locomotion type is also used in Exploring Mode. This game mode is the main feature of the game for the player to feel the immersion of the VR environment. The player will be part of the city character and walk through the city as if they are living there. As we explained this locomotion type in section 3.2.3, we will be using our arms to detect the decided players' locomotion. When the player swings their arms with the controllers faster than the threshold speed, the distance of the controller is calculated, and the VR body will move from the controller distance [6]. When implementing this feature there some challenges of detecting the controller speed or how we set the speed of the character or dumping values.

3.5.3 Grab Locomotion

Grab locomotion is used in the Building Mode. Unlike the Exploring Mode, locomotion in Building Modes needs to be more precise and accurate to place the items correctly. Since in Building Mode, players are over in the sky looking down at the city to build. Grab locomotion is the best choice to move around in this game mode because players are grabbing items to the place and grabbing the world to move, so the immersion and the design match the situation. When we were implementing on our game, we struggled on setting the rotation of the camera to make the grab locomotion work properly. Since we change Exploring/Building views frequently while playing, the camera rotation changes and it made the locomotion to the wrong direction [5].

3.5.3.1 World Rotation

There was feedback (section 5.2) about rotating the world. In our first game version, we did not have a system for changing the player's rotation with its hands. When the player wants to change the direction of the view, they have to change their position and direction physically. This was very uncomfortable and a disadvantage when long time playing.

Therefore, we added world rotation with hand movement. When the player presses both secondary buttons, it will enable the world rotation. The rotation angle is calculated using Unity's built-in function, ***SignedAngle ()***. This function returns the smaller degree of two vectors with -180 ~ 180. The calculation of vectors and angle of the right hand is below.

Right Hand Angle Calculation

$$CenterPivot = FirstGrabbedPoint\ (right) + FirstGrabbedPoint(left)$$

$$Vector_a = FirstGrabbedPoint\ (right) - CenterPivot$$

$$Vector_b = CurrentHandPoint\ (right) - CenterPivot$$

$$RightHandAngle = \boldsymbol{SignedAngle}(Vector_a, Vector_b, Vector.up)$$

Each side's rotation angle will be calculated from three-position values from the calculation above. The pivot for rotating the player is calculated by getting the median value of both hands' grabbed point.

3.6 Building Mode

As discussed in the previous sections, we have two types of modes in our gameplay, Building Mode and Exploring Mode. The player will be positioned up in the sky in the Building Mode, looking at the whole city structure from the air. This section will discuss all the implementations done for this game mode. There will be item panels for showing and choosing the items, items used for building, and performances of grabbing and snapping the items.

3.6.1 Item Panel

When grabbing and placing items in the Building Mode, we will be using an item panel shown in Figure 12. The panel is at the fixed position on the left hand, and the player will

be grabbing and touching buttons with their right hand. This item panel was referred to from the game Tiny Town VR, and it felt very intuitive and easy to grab and place objects in the world. In this panel, the player can see all the items they can put in the city, and by changing the category of the item, the player can visit other items such as houses, vehicles, and trees. It also has buttons to change Building/Exploring views, or a pause button to pause the game and save, quit or go back to the title screen.

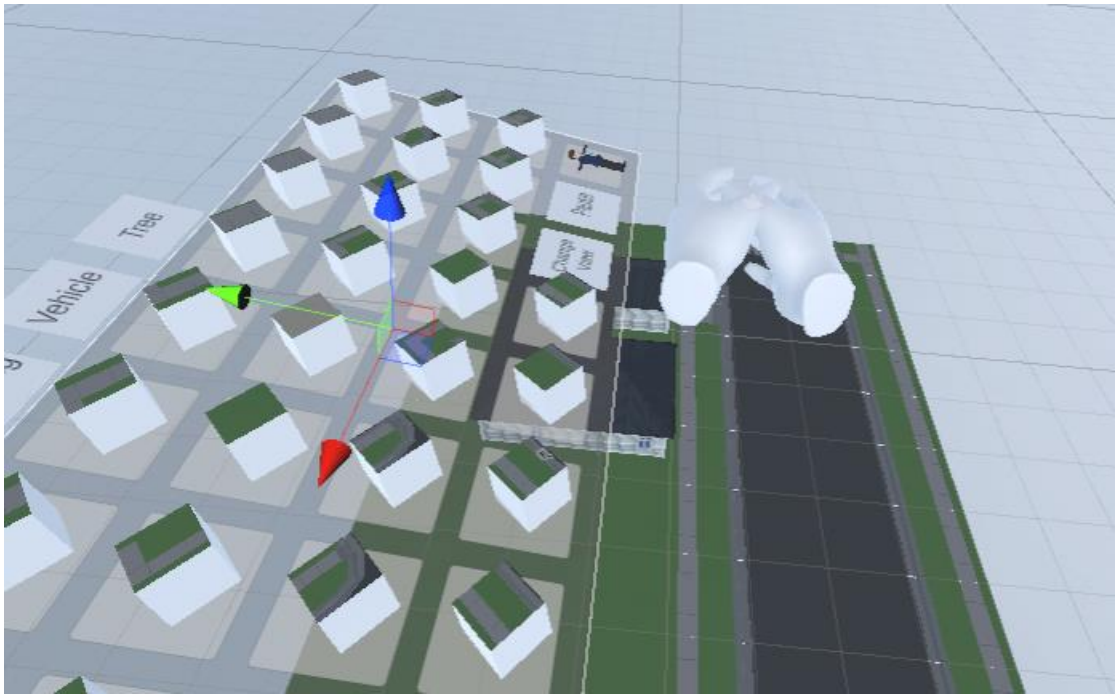

Figure 12. Item panel used in the game

3.6.2 Items

Criffin provided all the prefabs used in this game, and there were enough features to cover the construction of the immersive city. In the following section, I will discuss the detailed information on items we used for the game.

3.6.2.1 Ground Item

The virtual item object used in our game is the ground item. Ground items will be used in the lowest layer of the city, so when the player wants to construct and expand their city, they

will firstly be using this item to extend the land. There are several ground items, such as road items, grassy items, or sidewalk tiles, so that the player can create their unique world.

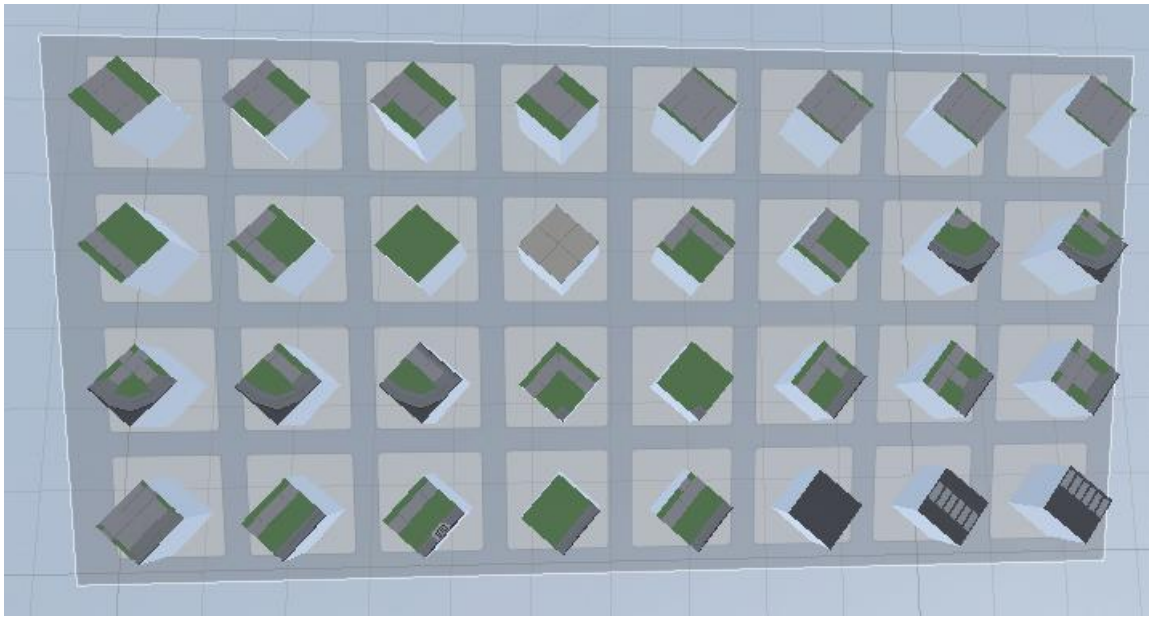

Figure 13. Groud items in our game

3.6.2.2 Other items

We have all kinds of items in our game, such as houses, vehicles, trees, etc., where you can add to the city. Figures 14, 15, and 16 are some panels of trees and houses. These are used in the game.

Figure 14. Plant items

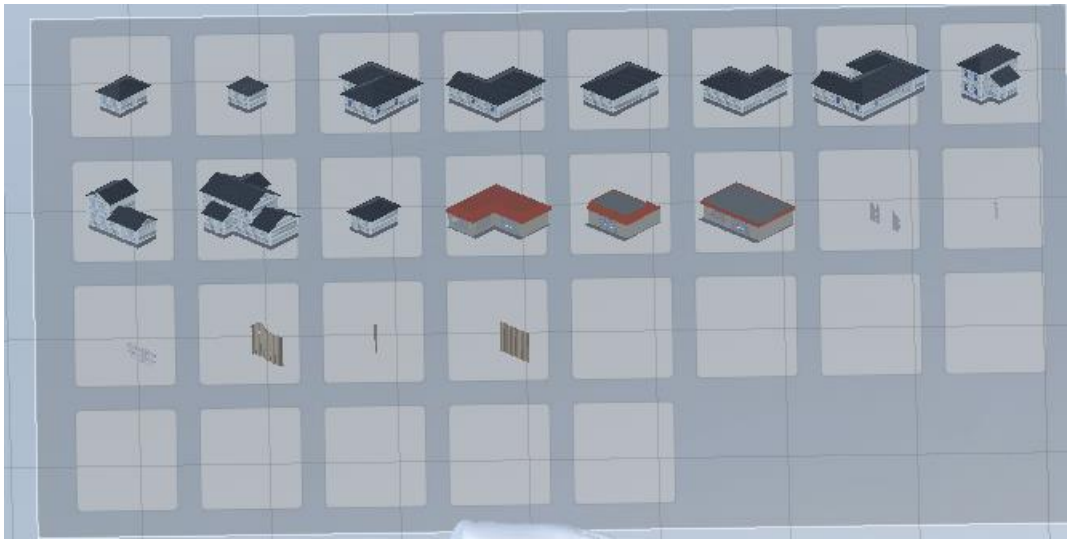

Figure 15. Building items

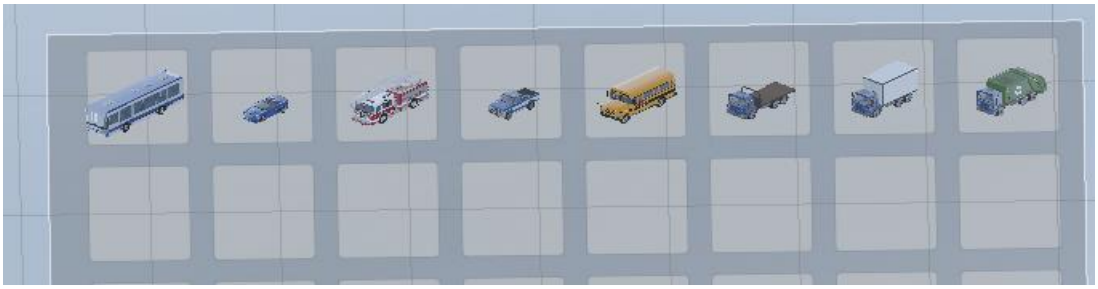

Figure 16. Vehicle items

3.6.3 Grabbing

As I explained in the item panel section, the player will be grabbing the items in the left hand with their right-hand controller. This can be done by touching the right hand to the player's item and pushing the right secondary trigger button. While the player presses it, it will remain grabbed and release the button to place the item in the desired position. The grabbing function was relatively simple since we used the OVRGrabber and OVRGrabbable script provided from the Oculus Integration (section 4.3). When the player wants to replace the items already placed in the city, it can be done by this provided script.

However, when grabbing from the item panel, we need to modify the given script, OVRGrabbable. When the player grabs the item from the item panel, it will instantiate the item so that it will not take the item shown in the item panel. After the item grabbed from the item panel is placed, the item's tag will be changed so that when the player holds the item again, it will not be instantiated.

3.6.3.1 Distant Grabbing

After the first evaluation, there was the feedback that grabbing items directly is far when the player wants to grab items at the corner of the item panel. To solve this problem, we added distance grabbing using DistanceGrabbable script supported from Oculus Integration. Distance grabbing allows the player to grab the items from a distant place. In this way, the

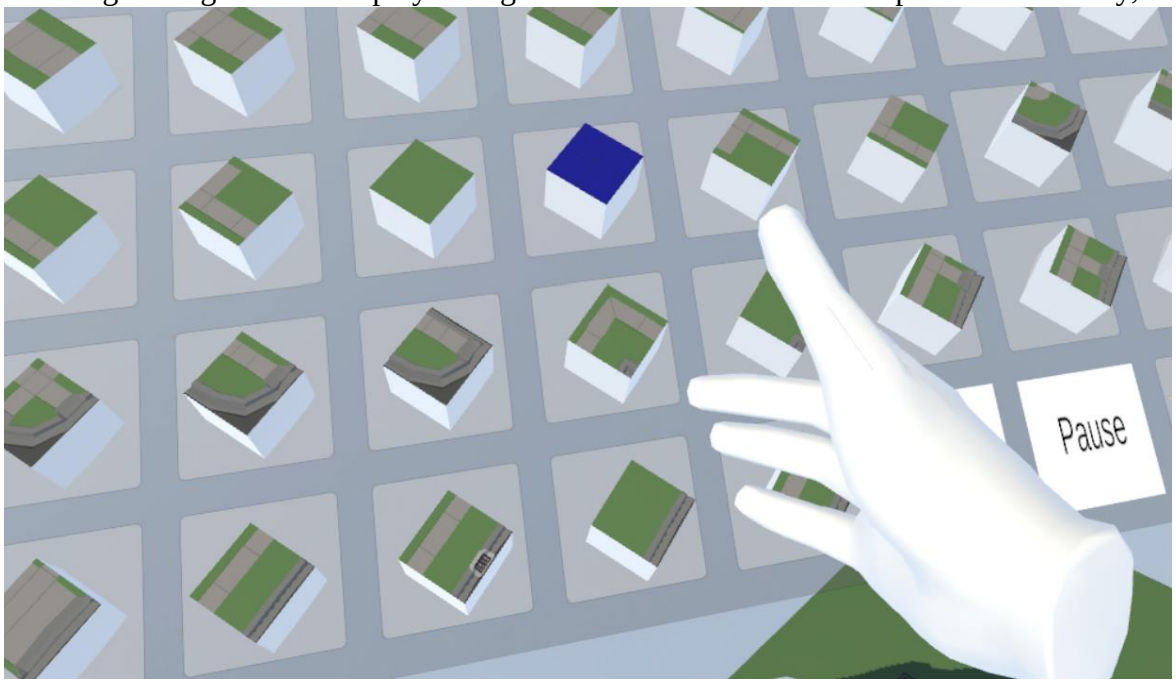

Figure 17. Pointed item highlighted in blue

player will not need to reach their arms to the corner of the item panel. When the player points their arms to the item they want to pick, it will be highlighted blue. When the player presses the grab button when the item is highlighted, it flies towards the hand.

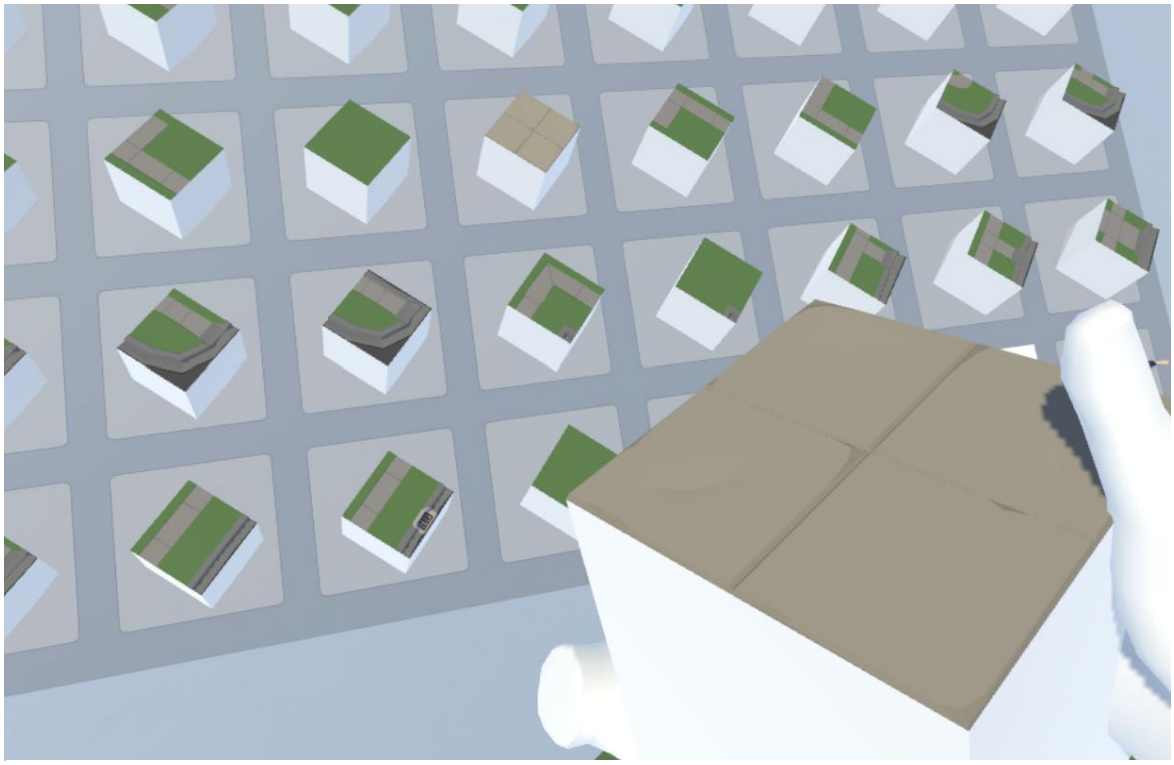

Figure 18. Highlighted object pulled to the player's hand

3.6.3.2 Deleting Items

After the player grabs items, it needs a function for the grabbed item. For this system, We implemented throwing deletion. The item will be destroyed when the player throws the items with a higher velocity than the threshold value we set.

3.6.4 Snapping

The adjustment of the item position is also implemented by modifying the OVRGrabbable. For this game, we chose the simple snapping technique, grid snapping. Since the ground item was a rectangle-shaped prefab, the grid snapping was good enough to place the item. Grid snapping is a technique that restricts the movement of the crosshair to intervals that are defined. In our game, the environmental items (ground, house, tree) x, y, and z position will

always be adjusted to the nearest integer. In this way, the ground item will always be at the integer position, and it will be easier to align items to create a flat surface.

Other items (vehicles, people) can be snapped in this way, but the snapping can also be disabled to place other items to create a crazy and funny scene.

3.6.4.1 Position Expectation Visualization

Items can be placed in the world by bringing the grabbed item directly to the place the player wants to position it. Still, it will be more accessible when the game indicates where the thing will be positioned after the player releases the item. We implemented a ghost of the item that will be placed after the release, as in figure 19 [7].

In figure 19, the player is holding the ground item, and the ground item's y-axis will always be snapped to the value 0. Therefore, the ghost of the item will be positioned right under the real object. In this way, the player does not have to bring items to a specific place every time and feels much more comfortable placing them.

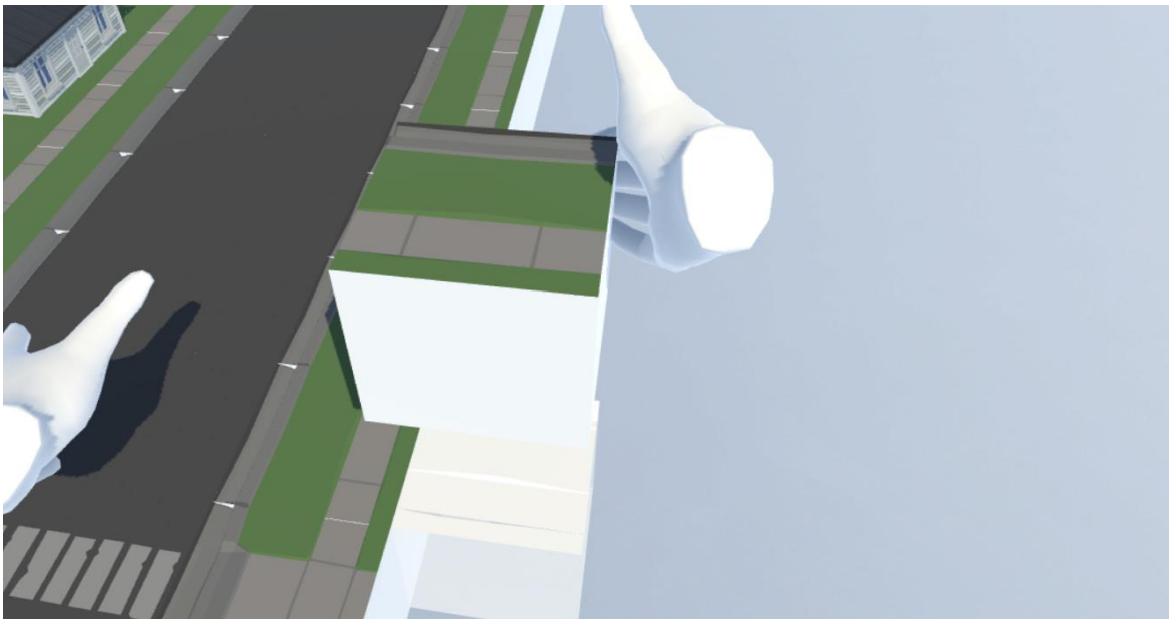

Figure 19. Item position visualization when released

3.6.4.2 Continuous Item Placing

When constructing a large city, rapid placement of the item is necessary. When the player presses the “A” button while grabbing an item, the grabbed item will be copied and snapped to the specific place. In this way, players can constantly press the duplication button to create a large city.

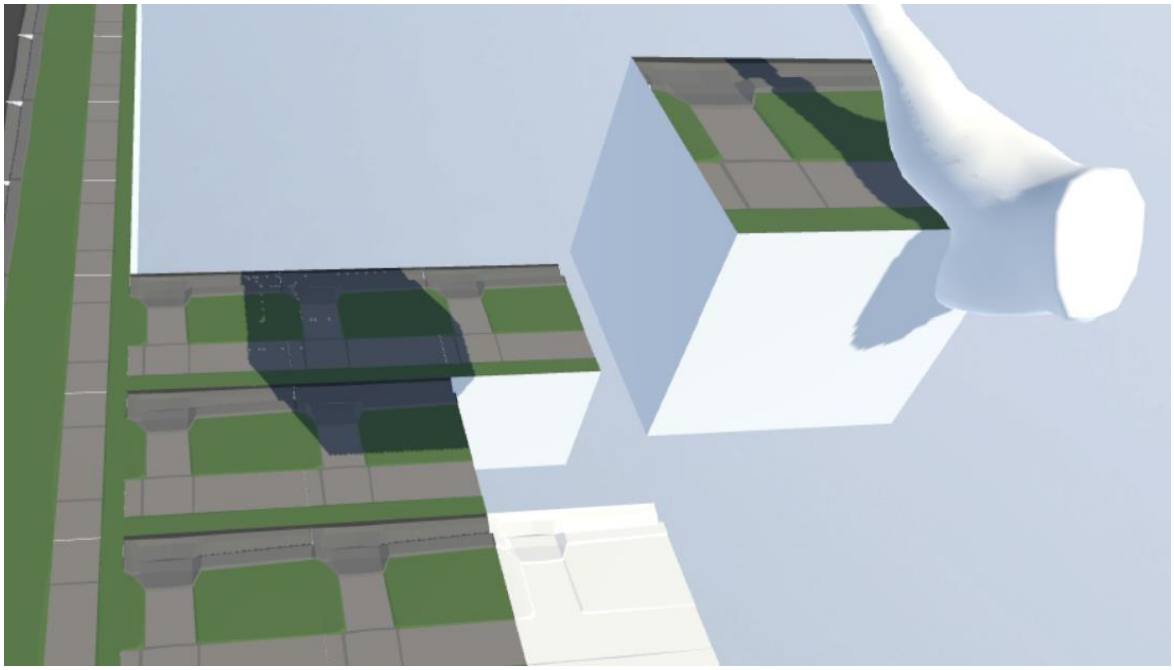

Figure 20. Item copyed by pressing buttons

3.6.4.3 Item Placement Error Indicator

When the player places the ground item in the position where there is already another item, it will be highlighted in red and will destroy the object when put in the place. This error indication will happen to both the grabbed items and to position planned objects. Therefore, when the player tries to place or duplicate in the red highlighted position, it will destroy it so that the ground object will never be doubled at the same position.

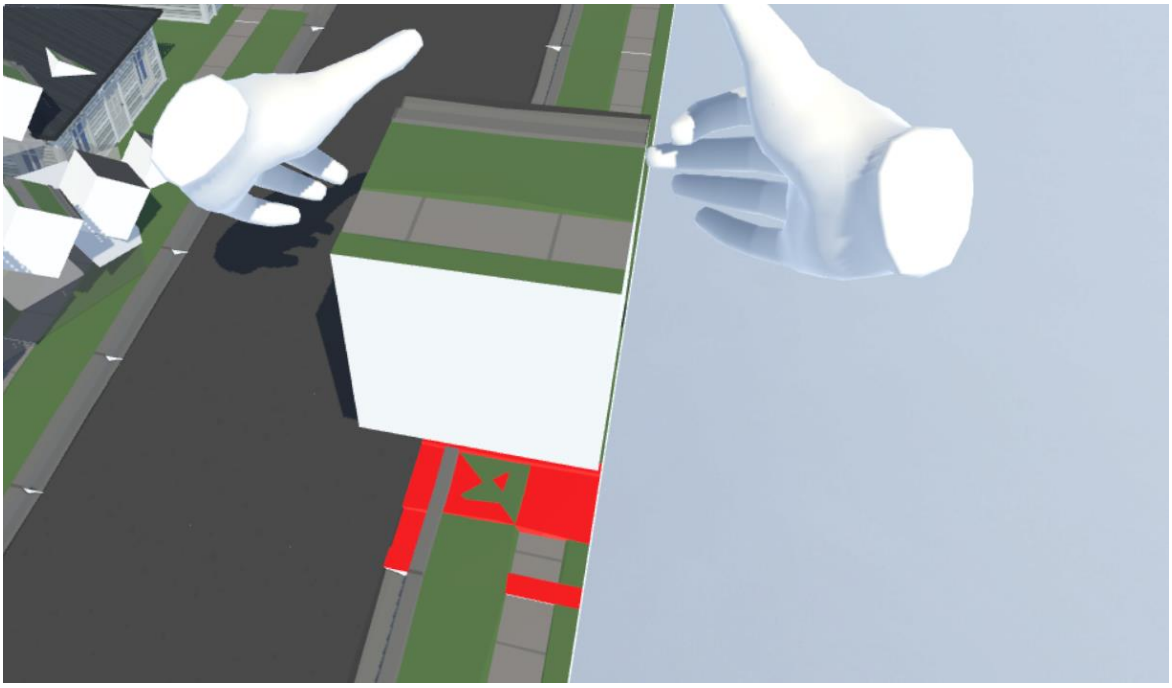

Figure 21. Item placement error highlighted in red.

3.6.5 View Transition

As explained in the previous section, we have two game modes, building and Exploring Mode. Players can change the Building Mode and Exploring Mode whenever they want to check the city they build. In the item panel, as we explained in section 4.5.1, we have a change view button in the panel to change the view between building and exploring. This can be done by changing the camera position. Every time the player presses the change view button, the game saves the camera's position and changes the view. When the player changes their view, it will constantly adjust to the character's position. The player has the character item on their item panel for the characters position panel. When the player wants to change their view to the ground viewpoint, they need to pick the character item and place it on the ground. After that, when the player touches the change view button, the camera will perform a transposition to the character item so that the player's view will change to the character's position. In this way, the player will feel more immersed in the game since they think they are going inside the character in the city.

Figure 22. Character item set in the item panel

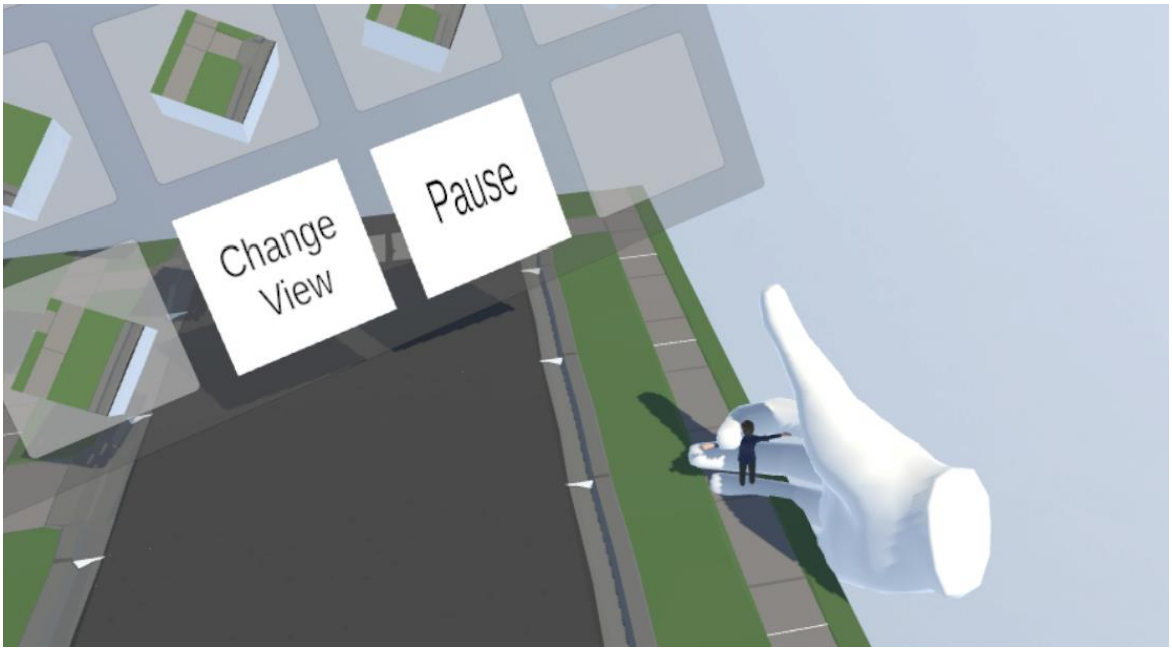

Figure 23. Charcter placed by player to the city

3.7 Exploring Mode

In this mode, players can walk around the city they made and feel the immersion of the virtual world. Building Mode had an aspect of realizing players' imagination, but it is more about the immersion in the world in the Exploring Mode. In the feedback (section 5.2), all the participants gave a positive feedback on the locomotion while Exploring Mode which is the walking-in-place locomotion. They felt the immersion to the game even though there was nothing to interact in this game mode. Therefore, this game mode has a great perspective of making players to be the character of the game. Adding other features such as residents will make people immerse more.

3.8 Saving and Loading

Saving and loading is one of the essential features for the player to play the game continuously. When the player opens the game next time loading the saved file, all the objects of the previous world are instantiated to the world, and players can continue from the last point they have ended. We have three different saves for our game that the player can store the world objects file. The player can go to the saving menu from the pause button on the item panel and choose which file to save their world. For the type of saving, we used a JSON file to save and load the world data to make it easier to handle and add other new implementations to it. JSON files can be used easily in Unity.

3.8.1 JSON Utility

In Unity, a function is available in the standard library called JSON Utility. This class is for working with JSON files and provides how-to “serialize” and “deserialize” to and from JSON-based data. For saving, we used the “ToJson” function to generate JSON representation, and for loading, we used the “FromJson” function to create an object from its JSON representation.

{"WorldName":"GameScene","CharacterPos":{"x":-8.729999542236329,"y":1.0,"z":30.8
":21060},"object_name":"SM_Env_Road_Crossing_02 (16)","position":{"x":25.0,"y":0
":"SM_Generic_Cloud_03 (3)","position":{"x":49.900001525878909,"y":62.2999992370
,"z":0.0,"w":0.0}},{"_object":{"instanceID":26044},"object_name":"SM_Env_Sidewal
9877810478,"z":15.300003051757813},"rotation":{"x":0.0,"y":-0.9411760568618774,"
0,"z":0.0,"w":0.0}},{"_object":{"instanceID":28990},"object_name":"SM_Env_Road_0
tion":{"x":0.0,"y":-1.0,"z":0.0,"w":0.0}},{"_object":{"instanceID":26212},"objec
00305175781,"y":0.09853938966989517,"z":77.20000457763672},"rotation":{"x":0.0,"
"x":14.199996948242188,"y":0.0,"z":37.400001525878909},"rotation":{"x":0.0,"y":-
,"object_name":"SM_Prop_Streetlamp_01","position":{"x":-42.5,"y":0.0,"z":-9.7999
osition":{"x":5.899997711181641,"y":0.0,"z":57.099998474121097},"rotation":{"x":
,"z":65.0},"rotation":{"x":0.0,"y":0.0,"z":0.0,"w":1.0}},{"_object":{"instanceID
ject":{"instanceID":26068},"object_name":"SM_Env_Grass_Edge_01 (1)","position":{
01 (53)","position":{"x":15.0,"y":0.0,"z":5.0},"rotation":{"x":0.0,"y":-0.707106

Figure 24. JSON save file example of our game

Figure 24 is the JSON save file of our game. When the player pushes the save button, it stores the world name, exploring and building view position, and each object information: name and, objects position and rotation. When we store the game objects directly as a "GameObjects" type to the JSON file, it generates an instance ID for each object but this ID changes when we load a new scene. Instead, we used the object's name to instantiate from the "Resources folder."

3.8.2 Resources Folder

The resources folder is Unity's built-in method to load assets during the gameplay. There should be a Resources folder in the project with the loading asset. When loading the asset, it can be done by **Resources.Load(assetname)**. When building the application, the resources folder will be data transferred into relevant data. It is effortless and straightforward to use, so we used it as a prototype of our games loading system. However, this Resource folder has a lot of disadvantages when the project gets bigger. When the size of the folder gets bigger, the build time, application size, the application activation time will increase. Also, we cannot update the folder after the application is built. We need to change the loading system for these disadvantages when the projects become bigger.

3.9 Requirements

There are some minimum requirements to be accomplished before publishing the game for publishing the software. This section will discuss the requirements we need to complete and how many of those I have succeeded in this thesis work. There are sixteen demands, and I have finished more than half, but it should be. It still needs to be polished.

1. Building Mode locomotion (section 4.5)

Building locomotion, the grab locomotion, has already been implemented in the game. It works very well, so this part polishing is not needed.

2. World rotation (section 4.5.3.1)

World rotation is also implemented with the grab locomotion works very well.

3. Exploring mode locomotion (section 4.5)

Exploring locomotion that is walking-in-place and joystick locomotion has already been implemented in the game. Walking-in-place works very well, but the joystick locomotion still needs to be polished.

4. Items for building the world (section 4.6.2)

All the items are provided from Criffin, and essential items such as ground, house, and trees are implemented. However, complicated items such as furniture, toys, or residents are not added to the application, so these should be added to the later version.

5. Item panel (section 4.6.1)

The item panel to get the item from is implemented and looks well. Position, design, and method of changing panels can be polished but works fine for now.

6. Grabbing items (section 4.6.3)

Grabbing items directly and from a distant place is implemented and works fine. It can still be polished by adding a laser pointer to show which item the player is pointing to.

7. Snapping items (section 4.6.4)

Snapping items is implemented and works fine. It might be better to change from position-based snapping to item-based snapping.

8. Item deletion (section 4.6.3.2)

Item deletion works very well, so this does not need to be polished.

9. Continuous item placing (section 4.6.4.2)

Works well. Depending on the snapping function, this should also be changed if the snapping is altered.

10. Changing game view (section 4.6.5)

Changing view with character placements is implemented and works fine.

11. Save and load (section 4.8)

Saving works very well, and it is easy to change which information to save, but the loading function should be changed later, as I explained in section 4.8.2.

12. Changing from tile to block (section 5)

We got the feedback about the changing the object type to block in order to make the Building Mode more comfortable.

13. Default world setups

The world where the player enters when starting a new game and is there in the game. It can be polished to make a better default world but not necessary.

14. Menu scene

The menu scene is implemented during this thesis but not great. It should be polished to make the player more interested in the game.

15. Sounds

There are no sounds to the game, and it needs to be added to increase the immersion of the VR environment.

16. City interactions

The current version of the game has no significant interactions, while the player can walk through the built city he Exploring Mode. More interactions, such as people saying hello, might increase the immersion and make the game more interesting.

17. Game setting

Game settings such as audio and graphics are not added to the game. It needs to be added at later point.

Table 1 shows the time it took to implement each component.

Table 1. Spent time per component

Components	Spent time(hours)
Building Mode locomotion	15
Exploring mode locomotion	33
Direct grabbing	30
Distant grabbing	30
World rotation	5
Item panel	20
Snapping items	80
Position plan visualization	20
Continuous placing	10
Item deletion	5
Continuous item placing	5
View transition	70
Save and load	70
Setting up items for building world	10
Default world setups	5
Menu	5
Bug fixing	40

Total	453

The grabbing function and snapping function was implemented with connected scripts so this feature alone took about 150 hours to implement. Even if we already used the grabbing script provided from Oculus Integration, we struggled a lot to modify for our game. Implementing locomotion features took more than we expected but it was mostly about getting values from Quest 2 inputs and the function worked well as soon as we successfully got the input values.

Overall, it took a lot of time implementing desired features and bug fixing took much longer than we expected.

4 Evaluation

To improve our game and gain a broader sight of the game features, testing our product with others and getting feedback about the strength and weaknesses of our product is necessary for developing a better product. In this section, we will explain the evaluation of our game.

4.1 Methodology

The testing and the reviewing was done by four people including three from the main stakeholder Criffin. All of them have played VR game before and thus, it will not affect to the evaluation for being surprised to the virtual environment. All testers were given the game file and ran it on Oculus Quest 2.

The participants had the basic instructions of the game (how to move, how to grab, and each button meaning) before they started playing and were asked to test both the building and exploring game modes.

After playing the game, the participants were asked to answer the questionnaire that we put together for the game testing. The questionnaire had seven following questions.

1. Did you enjoy the game?
2. How was the usability of building the city?
3. Did you feel an immersion while playing?
4. What did you like in the game?
5. What did you not like in the game?
6. Do you have any suggestions or ideas to improve this game?
7. Do you think changing the ground item from tile to block?

The given feedback will be described in the following subsection.

4.2 Feedback

The general playing experience was rated positively, and the feedback given from the participants was high scored (Figure 25) which means Criffin people were very pleased to the game. The first three questions were to rate the game's performance on a scale of 1 to 5, and every participant rated higher than three. All the participants rated over four, especially

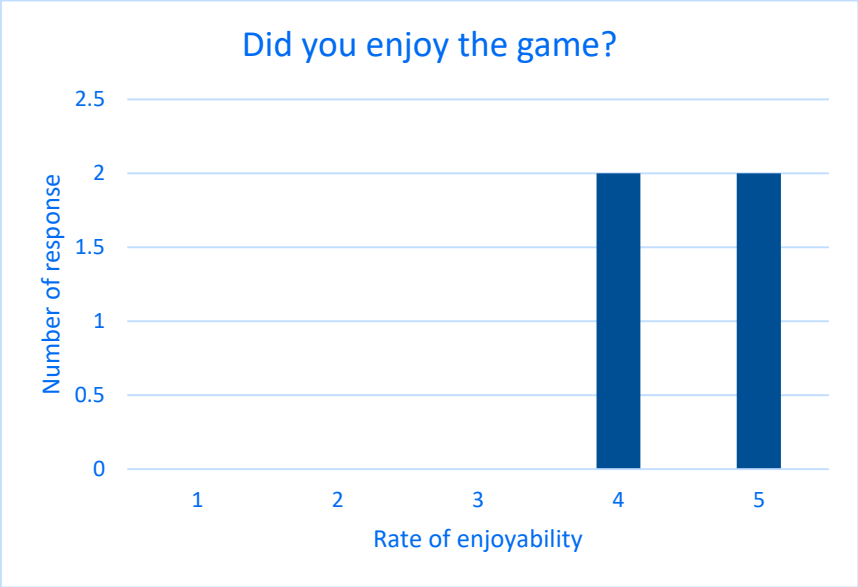

Figure 25. Rate of enjoyability

for the rate of enjoyability and immersion while playing the game (Figure 26, 27). Most of the participants pointed out that the locomotion of the Exploring Mode was excellent in question four. One tester even said the arm-swing locomotion felt like the immersion of

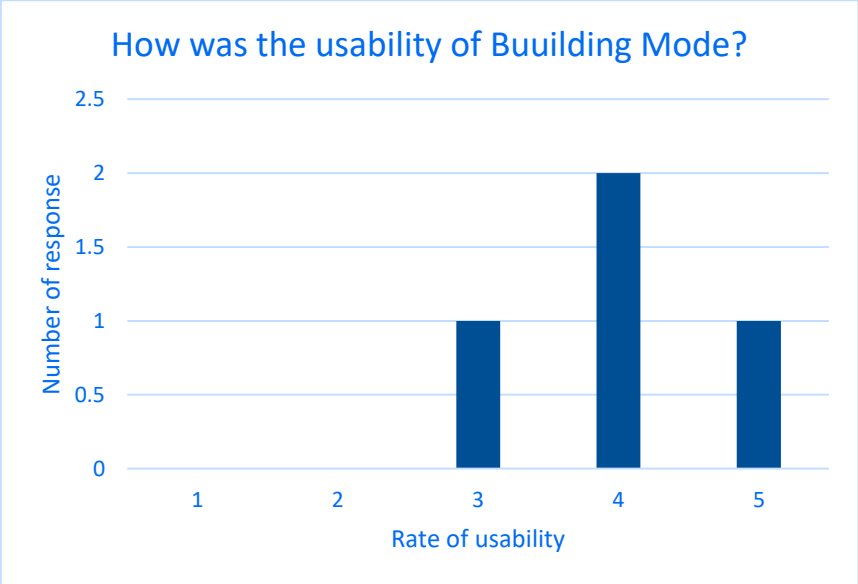

Figure 26. Rate of usabilility

being a pedestrian in the city. Also, another tester noted that switching view systems was

very interesting. However, in question five, participants pointed out the negative part of the game. One tester said that the building city could be better. The game version then had a buggy snapping function, and it was suggested to make it easier to snap. There was also a problem that the game crashed at some point.

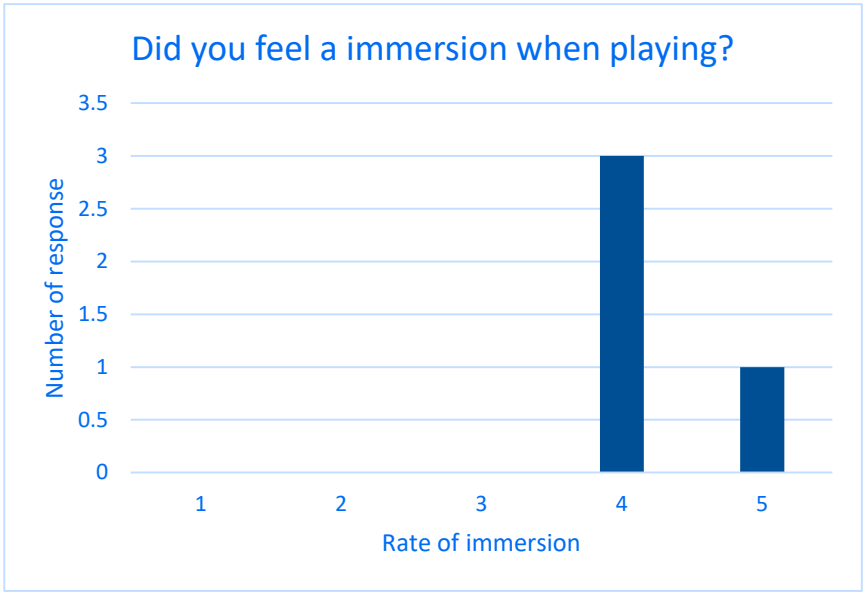

Figure 27. Rate of immersion

For the final question, I asked the participants if changing the ground from tile to block would be better (Figure 28, 29). All the participants said "Yes" to this question.

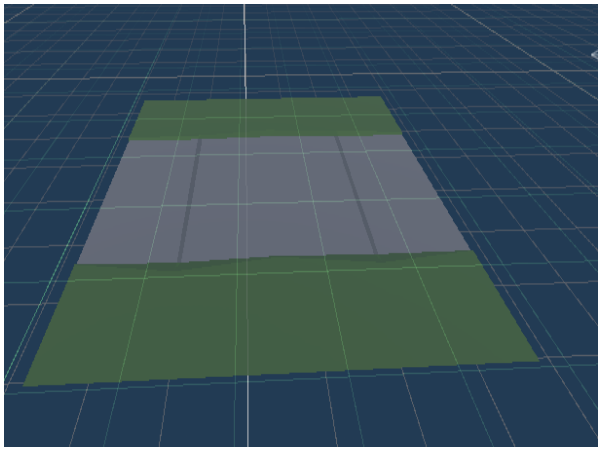

Figure 28. Tile shaped ground item

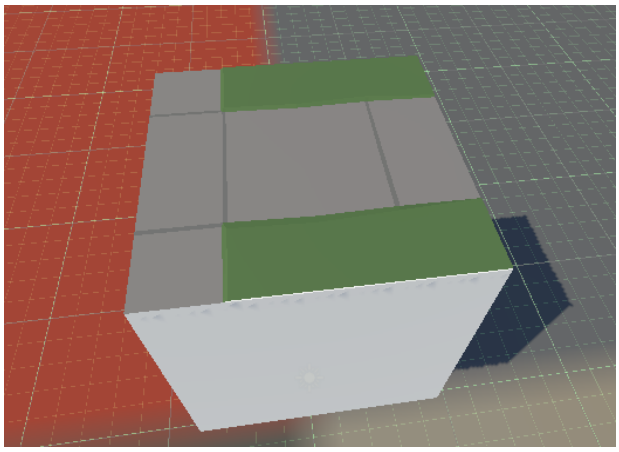

Figure 29. Block shaped ground item

Suggestions and ideas given from participants

- Add skybox.
- Add landscape around.
- Change index and secondary trigger input.
- Add zooming feature.

- Add rotating world feature.
- Change snapping to object based.
- Add smooth object rotation.
- Add sounds.
- Add plane with sound.
- Add catalog for changing building assets.
- Add laser ray.
- Highlight selected item.
- Change delete function by throwing.
- Add other form of locomotion.
- Snap building menu to the plane to release the left hand.

4.3 Improvements

From the feedback given by the participants on question six, I changed some parts of the game and improved the game experience. The following will be the updated things after the testing.

- Changed from tile to block shape.

As discussed in section 5.2, I changed the ground object to block.

- Added world rotation.

There was no rotation system in the game, so I added intuitive world rotation (see section 4.5.3.1).

- Skybox.

Added skybox to increase immersion.

- Changed from button deletion to throw deletion.

Before the testing, the item panel had a delete button to delete the grabbed item (Figure 30). I took off the button from the item panel (Figure 31) and changed it to throwing deletion to make the function more intuitive (see section 4.6.3.2).

Figure 30. Old item panel

Figure 31. New item panel

- Changed index and secondary trigger input action.

There was feedback on switching the index and secondary trigger input to make it a more natural VR game and switching the locomotion and item grabbing trigger.

- Added save and load function.

The previous version had no saving function, so all the placed items were initialized when the game stopped. Therefore, we added the save and load function to continue the game after exiting it (see section 4.8).

- Added continuous item placing

Players had to grab items from the item panel every time, even if they needed the same item. This made the game experience uncomfortable I added the continuous item placement by pressing a button each time (see section 4.6.4.2).

- Added distance grabbing

Adding distance grabbing so that the player does not need to reach their arms all the way to the item (see section 4.6.3.1).

- Changed item transition

There was a “Next” button in the previous item panel to change the item page (Figure 30). Instead, I added the catalog so that the player knows which page they are currently on and is more accessible on which page to see. Changing pages can be done by using the left joystick.

The second evaluation is ongoing.

5 Conclusion

The VR world-building game called VR World Constructor was developed through this thesis. This thesis aimed to create a world-building game that attracts players and makes them feel the game's immersion. The thesis described the development process and the tools used in implementations.

This thesis analyzed the similar existing products and locomotion techniques used in previous software. Based on this analysis, the goal was set, and therefore, all the implementations were decided, which are discussed in the implementation section.

The game prototype was tested and evaluated by other participants, and the improvement based on their feedback was described.

For the future plan, the requirements discussed in section 4.9 are still not completed. Essential features such as sounds, game settings, and city interactions should be implemented. There are still a lot of places that can be polished, and the menu scene must be polished or changed.

Also, as discussed in section 1.2, the overall plan of this project is to make a metaverse using a worldbuilding game. The final vision of the project has a feature of multiplayer and has lots of extensibility using the built city such as adding different game mode. It can be also used for educational place such as virtual classroom. My supervisors (Ulrich and Peeter) are stakeholders of the metaverse and educational aspects, they will be supporting for the further project.

In conclusion, we implemented a game that will lead to the bigger project and had a positive feedback from the users. Hopefully, this project will continue to the goal we have set for the project.

6 References

[1] C. Boletsis and J. E. Cedergren, “VR Locomotion in the New Era of Virtual Reality: An Empirical Comparison of Prevalent Techniques,” 2019.

[2] N. Griffin and E. Folmer, “Out-of-Body Locomotion: Vectionless Navigation with a Continuous Avatar Representation,” 2019.

[3] C. H. Lee, A. Liu and T. P. Caudell, “A study of locomotion paradigms for immersive medical simulation environments,” 2009.

[4] V. Kamboj, T. Bhuyan and J. S. Pillai, “Vertical Locomotion IN VR Using Full Body Gestures,” 2019.

[5] milkyway8008, “[Unity] VR 空間を移動する方法 3 選 [Oculus], VR locomotion types in VR environment,” 31 01 2020. [Online]. Available: https://milkyway8008.hatenablog.com/entry/2020/01/31/014244#%E6%8E%B4%E3%81%BF%E7%A7%BB%E5%8B%95.

[6] T. Horiuchi, “[Unity] Unity と Quest2 で VR 空間を移動する手振り歩行の実装, Implementing arm-swing locomotion with Unity and Quest2,” 18 03 2021. [Online]. Available: https://qiita.com/takafumihoriuchi/items/0f4bf9e6ed060bc2efbe.

[7] J. Jerald, The VR Book Human-Centered Design for Virtual Reality.Association for Computing Machinery and Morgan & Claypool, 2016.

[8] “Unity Script Manual,” [Online]. Available: https://docs.unity3d.com/Manual/ScriptingSection.html .

Appendix

I. Glossary

Metaverse - the metaverse is a hypothetical iteration of the Internet as a single, universal and immersive virtual world that is facilitated by the use of virtual reality (VR) and augmented reality (AR) headsets.[12]

Gameplay - A pattern defined through the game rules and a way in which the player interacts with the game.[13]

Game Engine - A software framework primarily designed for the development of video games.[14]

Assets - A representation of any item that can be used in your game or project. [15]

Index trigger - A trigger button placed on the index finger position on the Quest 2 controller.

Secondary trigger - A trigger button placed on the middle finger position on the Quest 2 controller.

[12] https://en.wikipedia.org/wiki/Metaverse
[13] https://en.wikipedia.org/wiki/Gameplay
[14] https://en.wikipedia.org/wiki/Game_engine
[15] https://unity3d.com/quick-guide-to-unity-asset-store

II. License

Non-exclusive license to reproduce thesis and make thesis public

I, Shumpei Morimoto,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to:

    1.1. reproduce, for the purpose of preservation and making available to the public, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright, and

    1.2. make available to the public via the web environment of the University of Tartu, including via the DSpace digital archives until expiry of the term of validity of the copyright,

of my thesis

Accessible VR World Building,

supervised by Ulrich Norbisrath, Peeter Nieler

2. I am aware of the fact that the author retains these rights.

3. I certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Tartu, **17.05.2022**