university of groningen

faculty of science and engineering

Virtual Ray Tracer in VR

Bachelor Thesis

July 12, 2024

Author:
Adrian Aaen

Primary supervisor:
Jiří Kosinka

Second supervisor:
Steffen Frey

Abstract

Ray tracing is an integral, but mathematically complex topic that can be hard to learn. Visualization is often used to aid in learning ray tracing, but traditional methods of visualization have been subpar. To fill this void and improve the educational accessibility of ray tracing, Virtual Ray Tracer (VRT) was created. It is a versatile tool that offers ray tracing visualization of various dynamic scenes in 3D. With this project, we extended VRT by adapting it to Virtual Reality (VR), in order to make ray tracing visualization more immersive, interactive and easy to use, further improving the learning potential of the tool.

Contents

1 Introduction **3**

2 Background **5**
2.1 Ray Tracing Visualization . . . 5
2.2 VR Applications . . . 6
2.2.1 Building Software for VR Platforms . . . 7

3 VR Adaptation **8**
3.1 User Interface . . . 8
3.2 Input Methods . . . 10
3.3 Locomotion in VR . . . 11
3.4 Physics Based Interaction . . . 12

4 Implementation **13**
4.1 User Interface . . . 13
4.2 Locomotion . . . 14
4.3 Object Interaction . . . 15
4.4 Audio & Haptics . . . 16

5 User Study **17**
5.1 Equipment . . . 17
5.2 Results . . . 17
5.2.1 Part 1 - VR-VRT . . . 17
5.2.2 Part 2 - Comparison with PC-VRT . . . 19
5.2.3 Part 3 - General Questions . . . 20

6 Discussion **21**

7 Conclusion **22**

8 Future Work **23**

Acknowledgements **25**

References **26**

Appendix **28**
User Study Questions . . . 28
Part 1 - VR-VRT . . . 28
Part 2 - Comparison With PC-VRT . . . 28
Part 3 - General Questions . . . 28

1 Introduction

Rasterization and *ray tracing* are arguably the two most important rendering techniques in the field of Computer Graphics. While rasterization is computationally cheap, ray tracing is expensive, and has up until recently only been used in pre-rendering workloads in e.g. film-making and simulations. Due to recent innovations in GPU manufacturing [1], real-time rendering using ray tracing is now possible. As a result, education on ray tracing is now more important than ever.

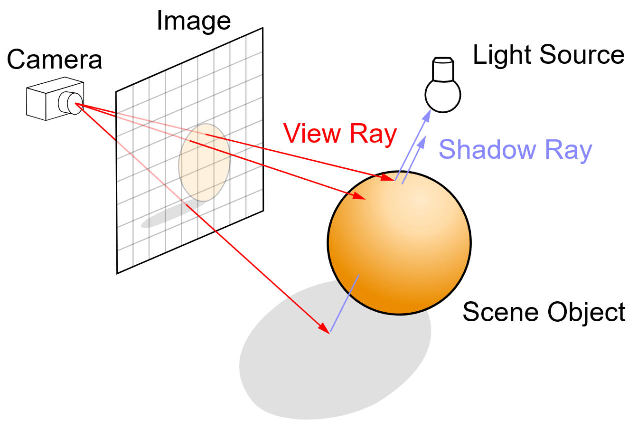
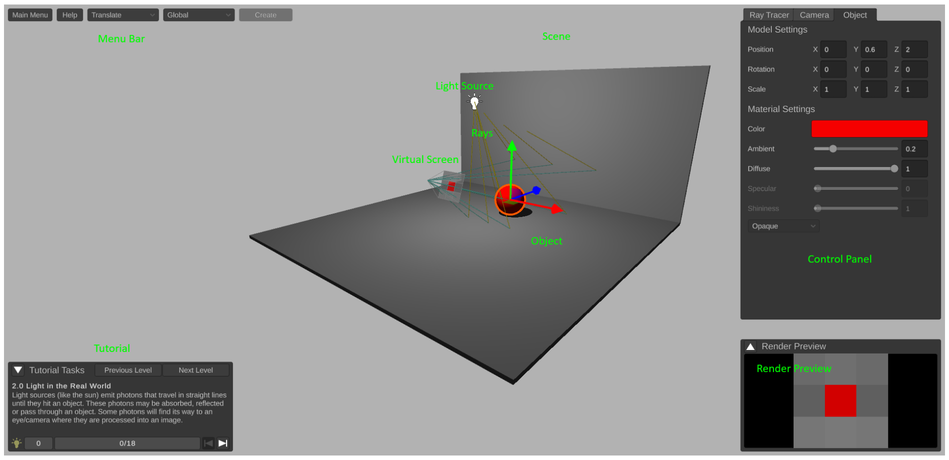


Figure 1: A simple illustration on the concept of ray tracing. *Adapted from* [2]

From a mathematical perspective, ray tracing is a complex topic. The core idea is quite simple, however more advanced techniques can be hard to understand. Visualization is generally regarded as a good way to learn ray tracing. Traditionally, simple illustrations such as Figure 1 were used for this, however, it being a static 2D diagram means its educational effectiveness is limited.

To aid in ray tracing learning in the Computer Graphics course at the University of Groningen, Virtual Ray Tracer (VRT) was created [2, 3], allowing interactive visualization through a dynamic, ray traced scene in 3D (see Figure 2). Though other tools for ray tracing visualization exist [4, 5], VRT is currently the most complete option. The main goals of VRT to facilitate an enhanced learning experience for students learning about ray tracing, as well as attracting future computing science students, especially those interested in the field of computer graphics.



Model Settings

Position X 0 Y 0.6 Z 2
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

Material Settings

Color 🟥
Ambient 0.2
Diffuse 1
Specular 0
Shininess 1
Opaque

2.0 Light in the Real World
Light sources (like the sun) emit photons that travel in straight lines until they hit an object. These photons may be absorbed, reflected or pass through an object. Some photons will find its way to an eye/camera where they are processed into an image.

0 0/18

Figure 2: Scene view of the Virtual Ray Tracer application [2, 3].

Since its creation, VRT has had several updates, including the addition of distributed ray tracing [6], gamification [7], a port to web and mobile [8], and some optimizations using axis aligned bounding boxes

(AABBs) and octrees [9]. Together, these updates have improved the educational ability and accessibility of VRT, and in this project, we proposed to further improve that aspect by porting VRT to Virtual Reality (VR), aiming to answer the following research questions on creating a high quality VR experience:

- What is the optimal UI for a ray tracing visualization tool in VR?
- Which type of locomotion is ideal for a ray tracing visualization tool?
- How can one achieve a high level of immersion and interactivity in VR?
- Will VRT in VR be easier to use compared to VRT on traditional platforms?

By exploring and applying theory on VR game design to answer these research questions, the project aimed to deliver the following contributions:

- An accessible and intuitive UI for VR,
- The debut of ray tracing visualization tools on a VR platform,
- A version of VRT with simple, intuitive controls and the added gamification associated with VR, improving the accessibility of education on ray tracing for (prospective) computing science students.

From this point onwards, Virtual Ray Tracer will be referred to as either VR-VRT, PC-VRT or simply VRT, depending on the platform in question, or lack thereof. Throughout the paper, there will be mentions of ”object manipulation”, referring to linear transformations (translation, rotation and scaling) when applied specifically to the interactable objects within VRT.

In Chapter 2 we explore the existing ray tracing visualization applications, how a VR application differs from a traditional PC application, as well as the motivation for porting VRT to VR. In Chapter 3 we discuss the theory of both creating and adapting software to VR, and the considerations to take into account when doing so. In Chapter 4 we take the theory discussed in Chapter 3 and apply it to VRT. In Chapter 5 we discuss the user study we performed and its results. In Chapter 6 we analyze the results discussed in Chapter 5, to evaluate the project in a scientific manner. In Chapter 7 we draw a conclusion of the project based on the findings of the user study. Finally, in Chapter 8 we discuss ideas for future research.

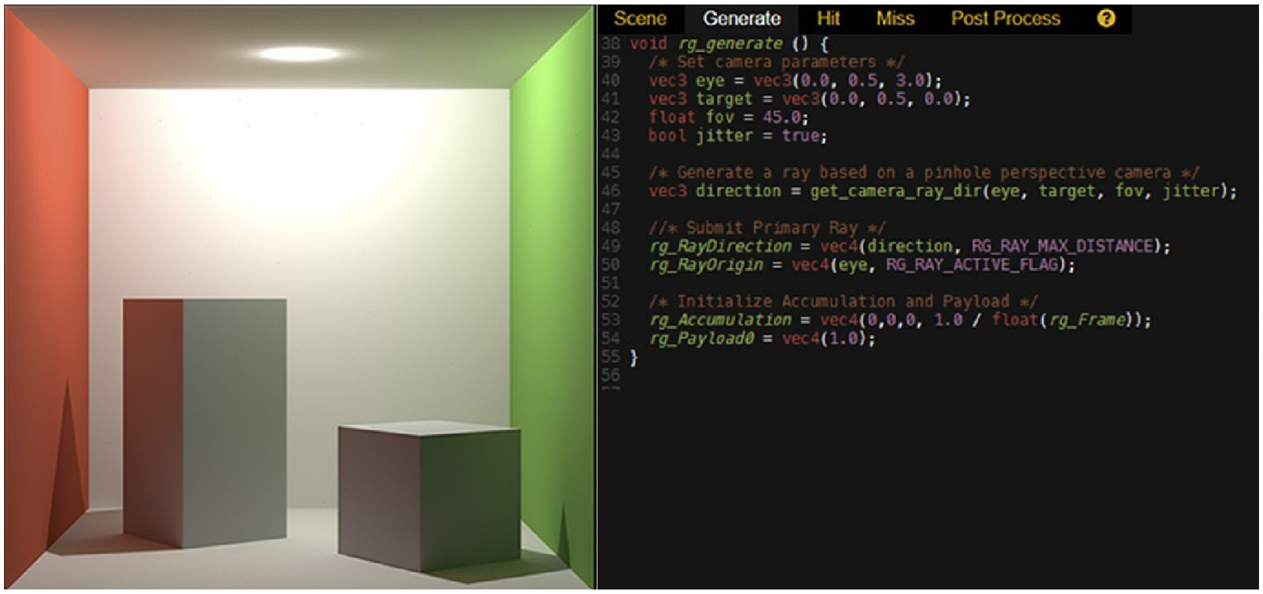
2 Background

In this chapter we explore the existing ray tracing visualization applications, how a VR application differs from a traditional PC application, as well as the motivation for porting VRT to VR.

As VR is a fairly young technology, high quality research is hard to come by, meaning most references used for the project is of the grey literature type. As this project is quite the practically oriented project and since grey literature tends to skew towards a more practical approach, the increase in grey literature sources compared to more typical academic projects than usual makes sense given the nature of the project.

2.1 Ray Tracing Visualization

There are several existing tools available that can be used to visualize ray tracing, each with a different purpose. *Rayground* [4] is a web-based framework aimed at rapid prototyping for developers who are developing their own ray tracing implementation (see Figure 3). The tool renders the final output, however it does not visualize the rays themselves, making it mainly a development-oriented tool.



```
void rg_generate () {
  /* Set camera parameters */
  vec3 eye = vec3(0.0, 0.5, 3.0);
  vec3 target = vec3(0.0, 0.5, 0.0);
  float fov = 45.0;
  bool jitter = true;

  /* Generate a ray based on a pinhole perspective camera */
  vec3 direction = get_camera_ray_dir(eye, target, fov, jitter);

  //* Submit Primary Ray */
  rg_RayDirection = vec4(direction, RG_RAY_MAX_DISTANCE);
  rg_RayOrigin = vec4(eye, RG_RAY_ACTIVE_FLAG);

  /* Initialize Accumulation and Payload */
  rg_Accumulation = vec4(0,0,0, 1.0 / float(rg_Frame));
  rg_Payload0 = vec4(1.0);
}
```

Figure 3: User interface of Rayground [4] showing a scene view (left) and an editor (right). *Adapted from* [2]

Another existing tool is the *Ray Tracing Visualization Toolkit* [5]. It is a tool used to analyze and debug ray traced scenes and works by connecting to the user's own ray traced program. Compared to Rayground [4], this tool can be used for ray visualization. However, to be able to use the tool, the user still needs a ray traced program that the tool can attach itself to, therefore keeping the barrier to entry unnecessarily high. An example can be seen in Figure 4.

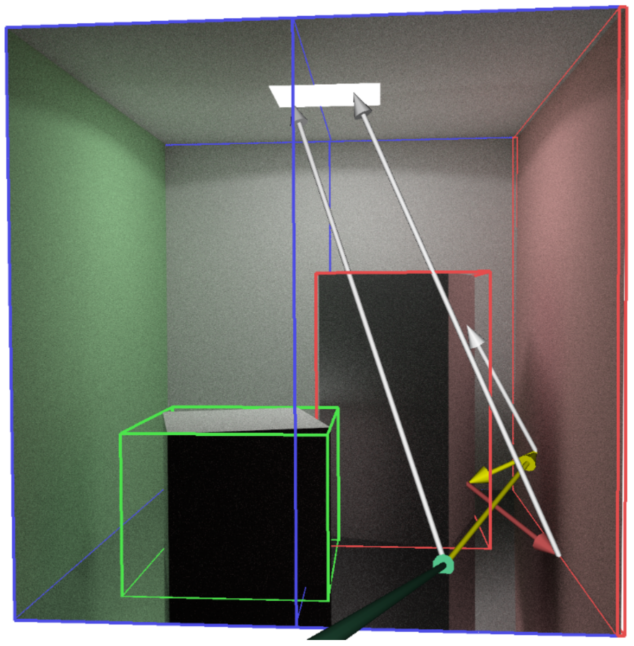


Figure 4: Layered visualization of rays in Ray Tracing Visualization Toolkit [5]. *Adapted from* [3]

Virtual Ray Tracer (VRT) [2, 3] is a tool developed by the Scientific Visualization and Computer Graphics Group of the University of Groningen. The tool was made to be an interactive learning experience, covering the principles of ray tracing through intuitive visualization. It is primarily aimed at students of the Computer Graphics course at the University of Groningen and users new to ray tracing in general, and it contains multiple levels with various scenes, covering several types of ray tracing.

Since its realization, VRT has received further development, with one of the major updates focusing on gamification [7]. The most notable addition to this update was the introduction of step-by-step tutorials, resulting in an improved user experience and a widened educational outreach.

So far, VRT exists on several platforms[7]; including Windows, Linux, MacOS Android and the web (using the WebGL API), and with this project, VRT is the first ray tracing visualization application available on a VR platform.

2.2 VR Applications

At this point, the question “What value would a VR version bring?” might come to mind. Let us explore that.

With VR, the default setup of human interface devices consists of the head mounted display as well as a controller for each hand. Each of these are tracked devices that capture real world movement and translate them to movement in the virtual environment. The VR controllers also usually have a grip and a trigger button, with the main function of the grip button being to grab virtual objects. With this level of input, one gains a level of *interactivity* that enables a user to mimick real life interactions in a way that comes *naturally* to us humans. Take for example the minigame in the VR game “Job Simulator” where you run a convenience store (see Figure 5). As a cashier in the game, interacting with the world and handling groceries is done almost the same as it would in real life, consisting of grabbing the groceries, scanning them, then placing them in bags and handing them to the customer.

Figure 5: A VR game where the user plays the role of a convenience store worker. *Adapted from* [10]

A VR headset also has one display for each eye with most of the light from the outside world blocked, facilitating a 3D visual experience with a natural sense of depth. Add sound to the equation, either with the built-in speakers that come with most VR headsets, or simply a pair of headphones, and the result is a very convincing, *fun* experience that enables a level of *immersiveness* that is hard to recreate with a mouse and keyboard or a video game controller. This level of immersiveness, ease of use and fun brings about an added layer of gamification, on top of what was already in place[7], and as a complete package, should in theory facilitate an increased potential for learning for future students of Computer Graphics. It could also be a tool, for example at events, to gain publicity and to entice prospective students.

2.2.1 Building Software for VR Platforms

Knowing the potential value of a VR version of VRT, we are ready to take the next step towards making a working prototype. However, to have a working prototype in the first place, we need to know how one goes about making a functional build for VR.

From the perspective of operating systems, VR applications do not differ much to traditional applications, however there are minor platform differences. There are two types of VR platforms; PCVR and standalone. As the name implies, on PCVR platforms the VR software itself runs on a PC. To facilitate this, VR applications run on a compatibility layer. One example of this is Steam VR[11], which is the first choice for PCVR video games. On the other hand are standalone VR headsets, which run their own operating system, meaning all VR software runs directly on the headset itself. In terms of software, standalone VR headsets are actually not very different from mobile phones, as the majority run Android based operating systems[12].

Given the nature of VR platforms, producing builds for them is fairly straightforward, as building for PCVR or Standalone means simply targeting Windows or Android respectively. Though, when building for Standalone VR, one of course has to tailor the performance profile of the application to the platform, taking into account the generally lower performance compared to PCVR platforms. Within Unity there are options to set specific performance profiles for specific platforms, and so switching is a fairly hassle-free experience, only requiring a few clicks to switch between the two platforms.

At the time of publishing, the builds and source for this project are available at `https://github.com/norwayman22/Virtual-Ray-Tracer`. If the various versions of VRT are ever unified (See Section 8), then the source and builds will be moved into the main repository, available at `https://github.com/wezel/Virtual-Ray-Tracer`.

3 VR Adaptation

One of the most critical challenges when creating experiences for VR is being able to provide a solid user experience. This encompasses not only the *user interface* (UI) of the product, but also elements such as *immersion*, *sensory feedback*, *interactivity* and the *virtual world* [13]. To evaluate the project based on this context, a user study was held (See Chapter 5), the results of which are discussed in Chapter 6. In this chapter, we discuss the theory of both creating and adapting software to VR, and the considerations to take into account when doing so.

Two undervalued examples that encompas multiple of these elements are audio and haptics. Audio alone often goes unnoticed, however, background music for example brings about an extra sense of *immersion*. Combining audio and haptic feedback can also give a level of *sensory feedback* and *interactivity* that severely enhances the experience, and can guide users in various ways, i.e. a hint of haptic feedback when hovering over an object would help highlight that the particular object is interactable.

3.1 User Interface

In terms of UI, the principles applied appear to be universal. Take for example the UI of The Meta Quest 3. As can be seen in Figure 6, the buttons are generally larger than on a personal computer, possibly a result from having to account for VR systems that can have sub-par controller tracking. In Open Brush, a tool to draw in VR, we see the same design principle in use (Figure 7). Here however, part of the UI is anchored onto one of the controllers (defaulting to the left controller), making the user effectively hold the menu in the same way one holds a smartphone, instead of it simply floating in place in front of the user. This adds to the *interactivity* of the program, one of the four elements of user experience design previously mentioned. Another thing to note is how in the case of OpenBrush, the UI consists of several windows arranged into a rotating carousel, and is controlled by the joystick on said controller. In comparison to the typical large canvas of traditional 2D UI, this allows for a drastic reduction in size, by instead utilizing the space of three dimensional volume, with a fairly minimal compromise to window accessibility.

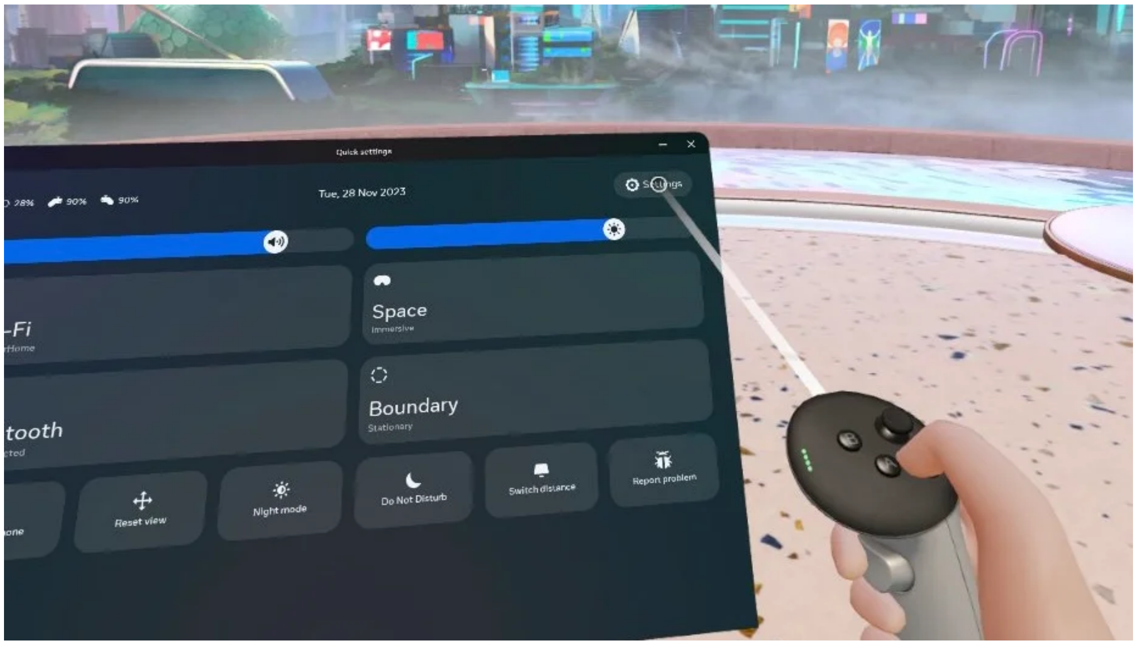


Figure 6: Meta Quest 3 quick menu showing a user pointing with their controller at the settings button. *Adapted from* [14]

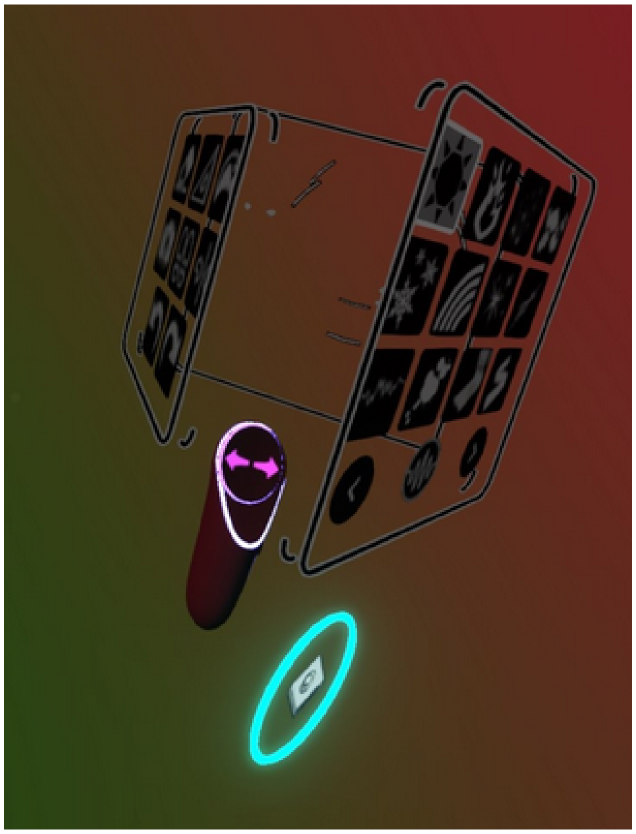


Figure 7: Menu in OpenBrush attached to one of the user's controllers. *Adapted from* [15]

Due to the nature of the controls in VR, there are other options to consider as well in terms of UI design. The UI could for example be positioned on the user's wrist or even on an interactable virtual tablet (see Figure 8 and 9 respectively), and it could even be controlled with virtual physical buttons. VR controllers have traditionally been using *ray interactors* when interacting with a UI, where using ray casting, the user would point the controller towards the UI like a remote and then pressing the trigger button for selection. Lately however, it is also common to see VR software adopting so called *poke interactors*, where the user navigates a UI by physically touching the tip of the controller against it, simulating how humans interact with touch screens.

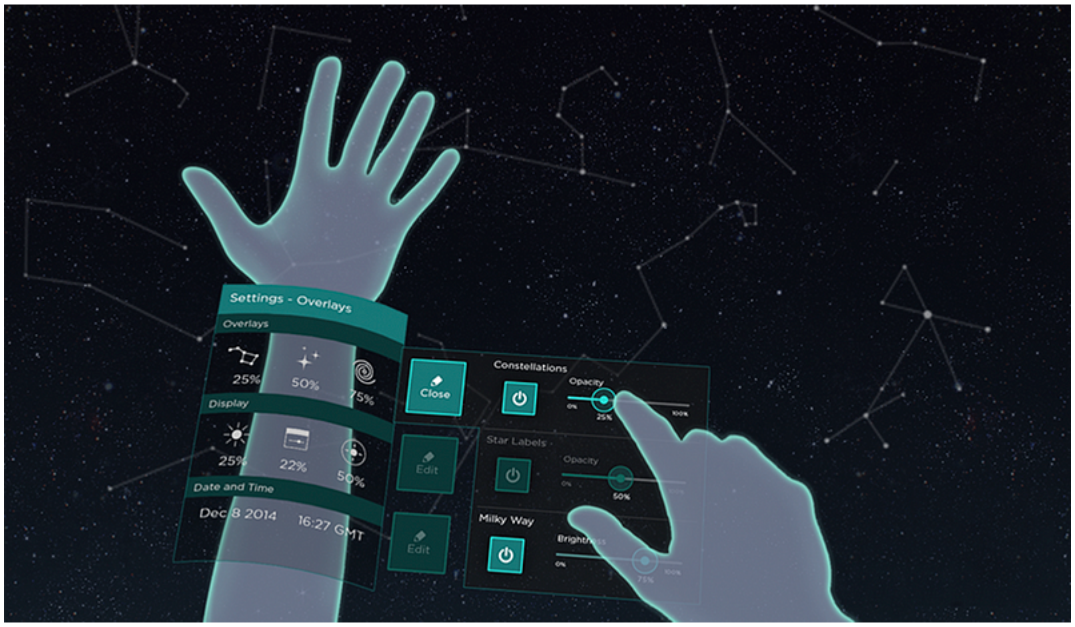


Figure 8: User Interface attached to the wrist of the user. *Adapted from* [16]

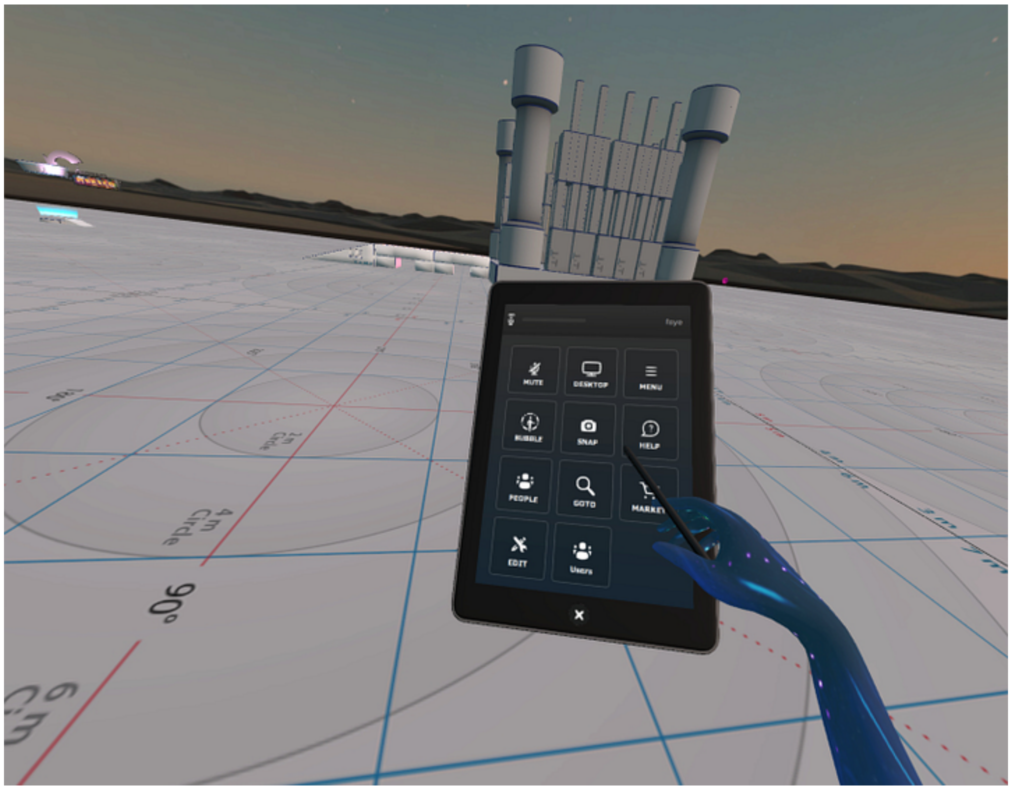


Figure 9: User Interface running on a virtual tablet. *Adapted from* [17]

3.2 Input Methods

On the topic of human touch, in addition to the typical *controller based interaction* model, some newer VR systems now also come with *hand tracking based interaction* built in. With the newly released Vision

Pro, Apple decided to forego controller based interaction entirely, relying solely on a hand tracking based interaction model [18]. This means that any VR experience wanting to reach the largest audience possible will also have to consider hand tracking based interaction, on top of the usual controller based model, resulting in two, possibly very different modes of interaction depending on the interaction design of the respective program. Some methods of interaction, e.g. the previously mentioned *poke interactor* would work particularly well with this interaction model, as it perfectly mimicks real world interaction, without having to rely on a controller in-between the user's hands and the interactable in question. On the other hand, *ray interactors* would be more challenging to implement for a *controller based interaction* model, however it is not an unreasonable choice. Besides, who would not want to fire lasers out of their own hands? Laser hands might not seem very educational, but it could prove itself as a useful tool in the context of gamification.

3.3 Locomotion in VR

Another important topic in the user experience design of VR, is the topic of locomotion. The term locomotion refers to the methods used for movement and navigation within an environment. In the case of VR locomotion, that environment is of course virtual. As XpertVR describes [19], selecting the right locomotion techniques allows users to have a better immersive experience, with the reduced chance of discomfort, motion sickness and reduction in the quality of the experience that comes with a mismatch in locomotion technique. They further specify that this allows educators to create "a more realistic and engaging VR environment for their students", one of the major goals of this project.

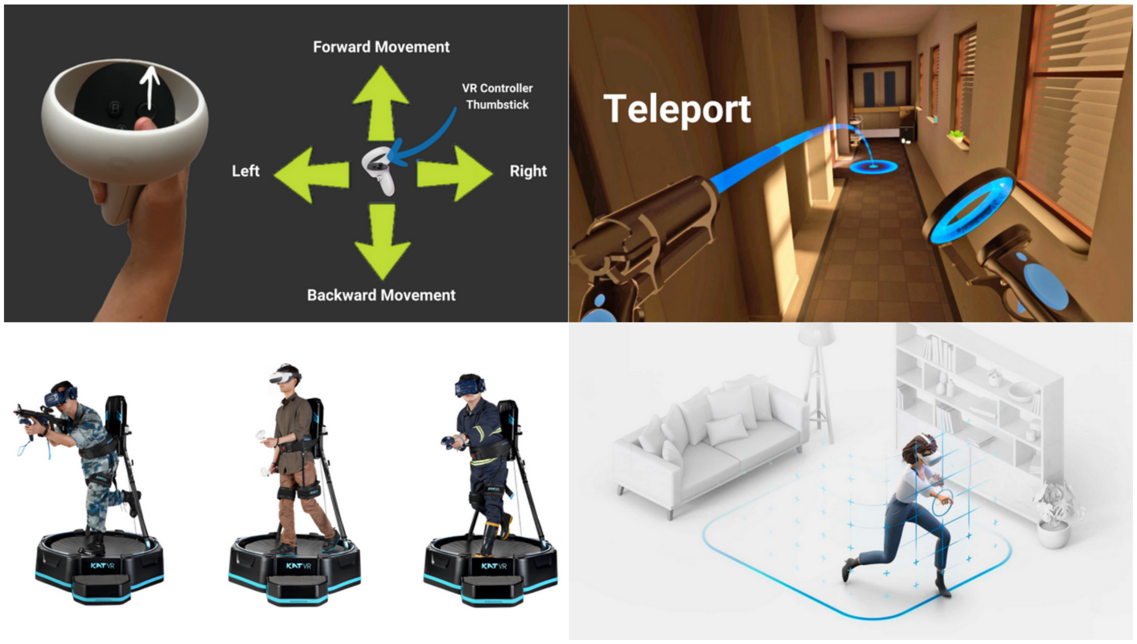


Figure 10: The four types of locomotion in VR. *Adapted from* [19]

In VR, there are four major types of locomotion [19]. The first type is controller-based locomotion, where virtual movement is based on the input from a joystick. The second type is room-scale-based locomotion, which is based on movement within a designated physical space in real life. The third type is motion-based locomotion, where physical movement from the user, (i.e. by using an omni-directional treadmill) is translated into virtual movement. The fourth and final type is teleportation-based locomotion, in which the user usually marks a point in the virtual environment, and the environment transitions to the new location. The transition is usually handled by some form of visual effects based on the idea of teleportation or running. Each of the four options have their strengths and weaknesses, all of which had to be thoroughly analyzed to decide which of them would be used for the VR port of VRT.

3.4 Physics Based Interaction

Complex physics systems in both simulations and games are challenging to implement. However, when executed well, it can be worth the time commitment. In VR specifically, a physics based interaction system would open up an enhanced level of depth to the experience through heavy focus on *interactivity* and *immersion*. The video game Boneworks is an excellent example of this [20], as almost any object can be interacted with, in the same manner as real life. This allows an impressive level of replayability, by allowing the individual to take advantage of their creativity to navigate through the various levels, as they see fit.

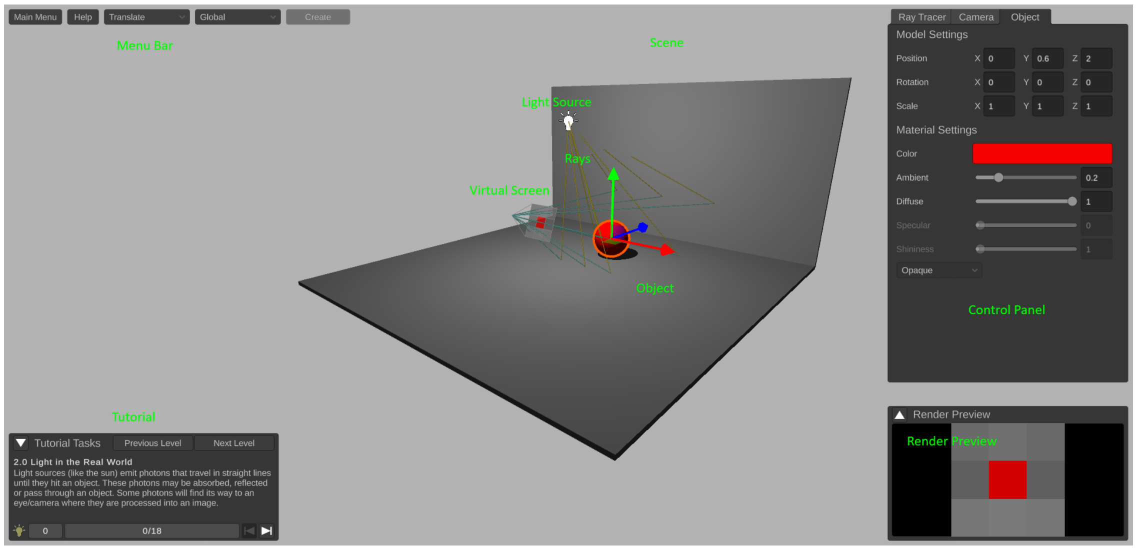
4 Implementation

By definition, to practice is the process of applying theory. However, taking theory and implementing it into a project is not always such a straightforward path. There can be twists and turns, which can sometimes even hide a surprise or two. In terms of this project, one could say that there were indeed a few twists along the way. That is what we will explore in this chapter.

At the core of this project is the XR Interaction Toolkit (XR-ITK) available in Unity. It contains all the developer tools needed to implement VR elements into a Unity project, and the development of this project would not have been as smooth without it. This toolkit will be further discussed in Chapters 4.2 and 4.3.

4.1 User Interface

In terms of UI, VRT has many different windows. In addition to the ones visible in Figure 11, there are also various windows tied to the main menu, such as e.g. the level selector and the settings window. Adapting all of these menus into a VR friendly format is easier said than done.



2.0 Light in the Real World
Light sources (like the sun) emit photons that travel in straight lines until they hit an object. These photons may be absorbed, reflected or pass through an object. Some photons will find its way to an eye/camera where they are processed into an image.

0 0/18

Model Settings

Position	X	0	Y	0.6	Z	2
Rotation	X	0	Y	0	Z	0
Scale	X	1	Y	1	Z	1

Material Settings

Color	🟥
Ambient	0.2
Diffuse	1
Specular	0
Shininess	1

Opaque

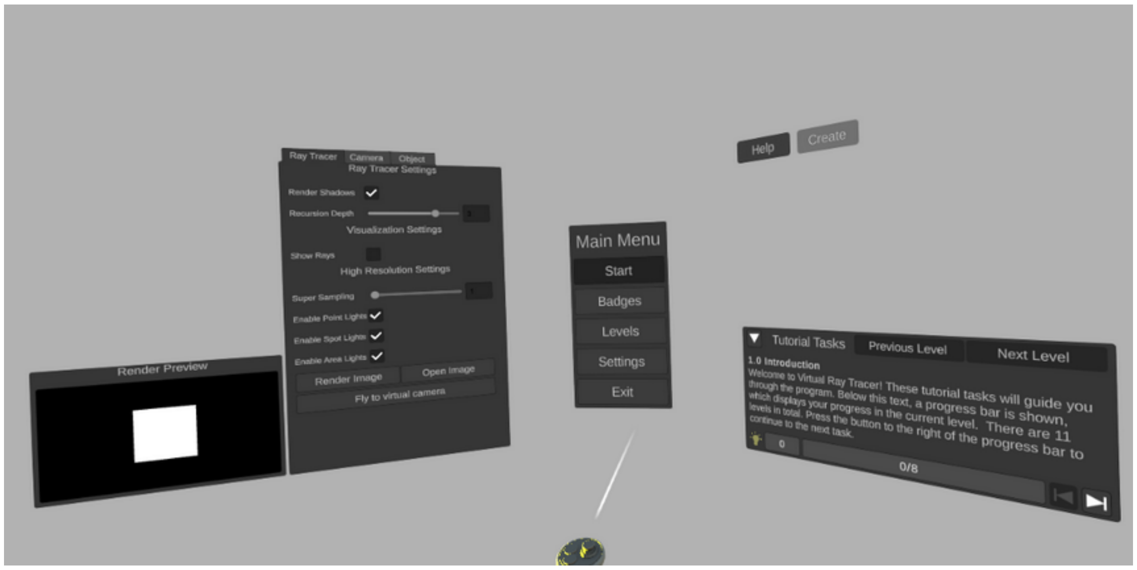
Figure 11: VRT layout on PC. *Adapted from* [21]

After having gone through several sources on UI design [22, 16, 17] and drawing inspiration from existing VR applications [14, 15], we eventually settled on a UI layout inspired by the UI from OpenBrush (see Figure 7) discussed in Section 3. It took time to get the design right, and it felt great to use once it was complete. However, as it turns out, when following the tutorial, users tend to often navigate between it and the control panel. In the current UI design, these windows were placed on different panels, resulting in a mess of constantly going back and forth to be able to complete tutorial tasks. Placing both of these windows on the same panel would have been possible, however, due to space constraints of the individual panel, we decided the better choice would be to discard the design entirely and rethink the design strategy, in favor of something more accessible.

Having seen the UI of Oculus[14], SteamVR[11] and the Apple Vision Pro[18], we realized that the foundations of all three UI designs are largely the same, with one main panel in the center, occasionally also with some panels on the sides, similar to how one would use multiple monitors on a PC. In VR, this type of UI is most often used in UI focused applications and works well for that use case. VRT on the contrary has more of a balance between UI and world focus, which will have to be accounted for if this style of UI is to be applied to it. That is a challenge we were willing to entertain.

Adapting this idea into a workable UI required attention to detail on multiple fronts. Making space for each UI element seen in Figure 11, required three fairly big panels, with one in the middle and one on each side, positioned at a slight angle. To make the UI easy to use, the main menu has been placed in the center panel. This type of layout can easily take up more screen space than necessary, therefore, great care was exercised into balancing the size of the UI with the text clarity, as well as the precision needed to interact with the UI.

Further reducing the visual obstruction, the UI has been made ambidextrous and appears attached to one of the controllers, activated by the trigger button on said controller being held down. This way the UI will not be in the way when not needed. The result can be seen in Figure 12.



1.0 Introduction
Welcome to Virtual Ray Tracer! These tutorial tasks will guide you through the program. Below this text, a progress bar is shown, which displays your progress in the current level. There are 11 levels in total. Press the button to the right of the progress bar to continue to the next task.

Figure 12: UI layout in VR-VRT.

With the new design, the UI is quite capable and easy to use. However, it is on the larger side and not optimally space efficient. For a future version, a redesign with focus on dynamicity and space efficiency would be welcomed, though such an advanced design would not be able to fit within the scope of this project, due to time constraints.

4.2 Locomotion

As mentioned in Chapter 3, there are four types of locomotion, each with their own purpose. Deciding on which to implement in VRT, motion-based locomotion is obviously the least attractive option. Although the VR market is growing, VR headsets still are a non-significant investment, and are therefore still relatively uncommon. Investing in an omni-directional treadmill in addition to a VR headset would of course account for a quite small demographic of users, meaning it is not worth pursuing.

The instantaneous movement of teleportation-based locomotion can be convenient, however it also has some downsides. Firstly, teleportation requires a teleportable surface, which would not work well with VRT, as there would not be much area that could be teleported to. Additionally, the instantaneous movement can be quite disorienting depending on the user, meaning users prone to motion sickness pair particularly poorly with this type of locomotion.

Controller-based locomotion is the obvious choice, as it is familiar for anyone who has ever used a video game controller, and works with practically any environment. In VR, it works slightly differently however. Due to the user controlling the camera with movements from their physical head, the forwards, backwards, left and right directions of controller-based locomotion actually depend on where the user is physically looking, including the height aspect. This means that if a user were to move forwards while facing the floor, they would move downwards instead of forwards. This seems slightly disorienting at first glance, however users quickly acclimate to this type of movement (see Chapter 5).

Last, there is room-scale-based locomotion. Its availability is dependent on the VR headset, though it appears to be fairly common. The main issue is actually having the physical room to be able to move within.

If available to the user however, this type of locomotion pairs well with controller-based locomotion and has been added to VRT.

Camera movement is closely tied to locomotion, and there was one choice to be made within this context as well. The choice lies between continuous turning and snap turning. Since this project aims to be accommodating for users prone to motion sickness, the instantaneous movement of snap turning was turned down in favor of continuous turning. This type of camera movement is also largely the same as when using a video game controller, except one aspect. In VR, the user also turns the camera with their head, meaning vertical turning with the controllers would be quite disorienting and is therefore disabled. This means that one can only turn horizontally when turning continuously with a VR controller.

The locomotion and camera movement available in VR may appear slightly unorthodox on the surface. However, these quirks have been tried and tested over the course of many years, giving each of them their deserved place in Unity's XR-ITK. That is, each type of locomotion excluding motion-based, as the toolkit unsurprisingly does not contain a provider for it, largely due to the reasons mentioned previously.

4.3 Object Interaction

Interacting with objects is one of the core features of VRT. In all versions of VRT, selecting an interactable object reveals gizmos that can be dragged to translate, rotate or scale said object (see Figure 11). Reusing this system for the VR version would not pose any issue, however in practice, it does not make as much sense as the alternatives.

As VR tends to promote more natural types of controls, object interaction is usually implemented using ray or grab interactors, or a combination of both. For VRT, we chose to use a ray interactor, as it is more fitting to manipulate a scene in a simulation tool from the perspective of an observer, rather than the user to some extent being part of the scene.

Version 3.0.4 of the XR-ITK was available at the start of the project, however the Unity package manager refused to acknowledge the existence of anything newer than version 2.4.3. The cause for this is unknown, though not having access to the extra layer of polish available in the newer version has had quite an impact on this project, especially in terms of object interaction.

The ray interactor available in the older version of the toolkit is based on a linear ray without any motion smoothing, making it quire rigid. Having implemented object manipulation using this ray interactor, handling objects felt fine, however it was quite rough, and we were definitely not satisfied with the results. UI interaction was also hampered, making interactions which require even a hint of precision challenging to execute, for anyone not possessing the hand stability of a surgeon. This dissatisfaction fueled the discovery of a workaround which allowed the newest version of XR-ITK to be installed without any issues.

In the up-to-date version of XR-ITK, the ray interactor uses a curve based ray with motion smoothing (See Figure 13), resulting in a severely improved experience in which interacting with both objects and the UI requires far less precision. In the updated toolkit there are also models for the Meta Quest 3 controllers, which is used to visualize the tracked positions of the controllers, and comes with animations for the joystick, grip button and trigger button to reflect their actual state, enhancing immersiveness.

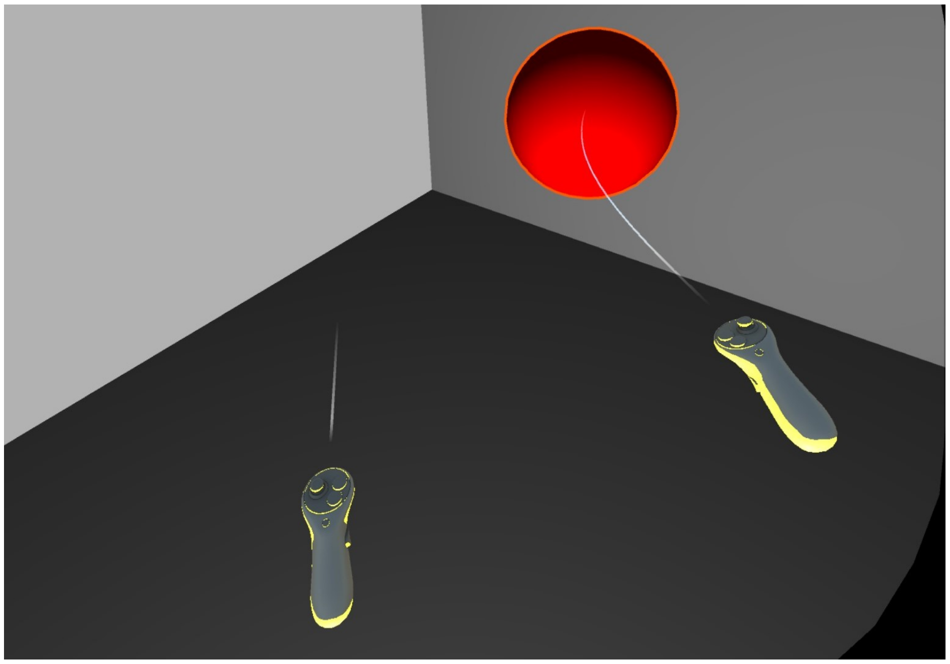


Figure 13: A sphere in motion, being moved towards the user's left. Notice the arched ray caused by the movement of the right controller, as well as the short ray on the left controller due to the ray not being hovered over any interactable object.

4.4 Audio & Haptics

In terms of audio, the existing feedback sound effects from VRT have been ported. These have to be added manually, one by one, meaning thorough testing would have to be performed to be able to guarantee that all sound effects have been successfully ported, however everything should in theory be in place.

The XR-ITK has built in haptics logic for its interactors, meaning any interaction features a slight vibration in the respective controller. This would for example be hovering over an interactable object or grabbing said object. This works in tandem with the interactor ray lengthening when hovering over an interactable object, serving as a visual hint for the action in question.

Personally, VR-VRT feels natural and intuitive. The locomotion and camera movements feel familiar, the object interaction is smooth and seamless and the UI displays all the relevant information on command, without unnecessarily obstructing the field of view in any way. Of course, this is all based on subjective testing from someone who holds a stake in the results of the project, bringing about an inherent bias. A need for a more objective manner of evaluation is the main motivation for the user study, the contents of which is discussed in the next few chapters.

5 User Study

In an attempt to objectively measure the quality of the project, a user study was held. The user study has been split into 3 separate sections. For the first section, the participants were asked to play through the tutorial of VR-VRT, and were then asked to answer questions on the UI, object manipulation and movement system. For the second section, the participants were asked to play through the tutorial of PC-VRT, followed by answering questions similar to those in the previous section, but in the context of comparing the two versions. For the third and final section, users were asked to fill in a fully optional section, with demographic related questions, such as age, education and familiarity with VR and any 3D modelling adjacent software. To promote the contribution of constructive criticism throughout the user study, each section featured multiple questions asking the participants for remarks on the various topics.

The goal of the user study was to first attempt to assess VR-VRT without a context, followed by a comparison between VR-VRT and PC-VRT in an attempt to highlight the strengths and flaws of both versions, as well as the possibility of finding any weaknesses within VRT itself, independent of any platform (See Chapter 5.2.1 and Chapter 5.2.2 respectively). All questions can be found in the Appendix. There are seven participants. While, there is enough data available to draw conclusions from, more participants would have been ideal.

5.1 Equipment

The setup for the user study consists of a 24.5" 1920x1080 LCD monitor, with a refresh rate of 240Hz. Assuming the user maintains a proper posture, the monitor will be positioned at a distance of roughly 60cm, resulting in ~40 pixels per degree (PPD), a measure of perceived visual clarity. The monitor was connected to a gaming laptop capable of running VRT at a frame rate matching the refresh rate of the monitor. For the VR setup, we used a Meta Quest 2 VR headset, which features one 1832x1920 LCD per eye, running at a refresh rate of 120Hz and with a reported PPD of 20. The headset can operate in both PCVR and standalone modes and we chose to use the former for this test. The Quest 2 runs on the Qualcomm XR2 platform, meaning there would be a reduction in visual fidelity if we were to use the VR headset in standalone mode. With VR-VRT running on the PC and the VR headset in PCVR mode, connected to the PC with a high speed USB-C cable, the graphical fidelity should in theory be identical to running PC-VRT with the PC able to match the frame rate with the refresh rate of the VR headset. Still, PC-VRT is at an inherent advantage, as the perceived visual clarity and motion clarity is theoretically twice as high on the monitor compared to the VR headset, based on their respective PPD and refresh rate values. This could have a slight effect on the difference in quality, subconsciously perceived by the participants. Keep that in mind as we progress through the results.

5.2 Results

As the user study is split into several parts, we will discuss each of them separately.

5.2.1 Part 1 - VR-VRT

To begin, participants were asked to complete the tutorial of VR-VRT. The tutorial consists of six tasks, with the participants on average quickly acclimating to the controls, as well as comprehending the tutorial instructions with relative ease, resulting in the users swiftly completing each tutorial task.

For the questions in the first section, 85.7% of the participants found both the controls and UI intuitive, with 57.1% finding some UI elements too small, making them slightly too hard to select. In response, the sizing of problematic UI elements has been rectified after the completion of the user study phase.

In terms of locomotion, 71.4% indicated that the movement system feels good. One participant (14.3%) in particular heavily disliked the movement system, stating that they experience significant motion sickness from general lateral movement, rendering any type of movement in VR triggering to the particular individual.

Moving on to object manipulation, the general consensus was that translation, scaling and rotation of objects felt "natural" and "like second nature", however there was a general lack of precision that the participants would have liked to see remedied. One participant suggested the possibility of a toggle-able advanced scaling mode, which could perhaps limit interaction to a single axis or plane at a time. We agree with the individual, however, this would not be within the scope of the project, and has therefore been placed in Chapter 8, in which future work is discussed.

The results of the linear scale questions for this section can be found in Figure 14.

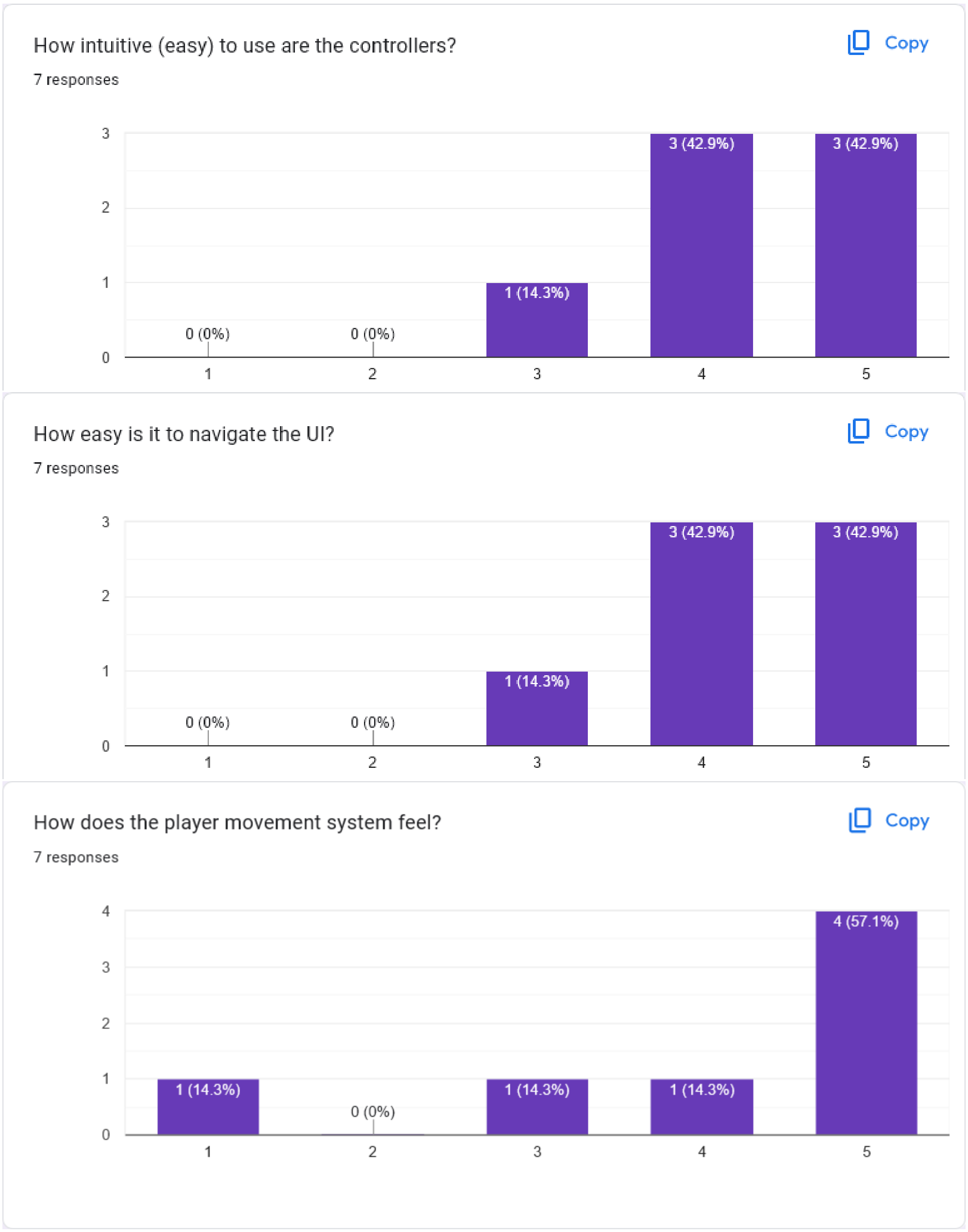


Figure 14: A summary of the results of the linear scale questions in the first section, with 1 being the lowest rating and 5 being the highest.

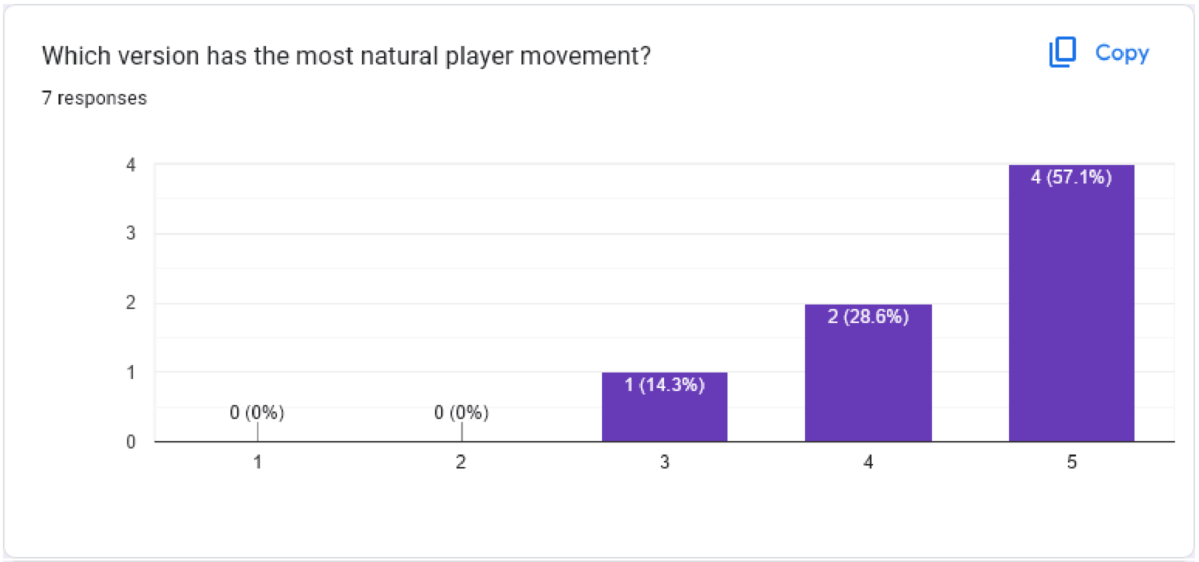
5.2.2 Part 2 - Comparison with PC-VRT

For part two, the participants were once again asked to complete the tutorial in VRT, but this time on the PC version. This tutorial consists of 20 tasks, with the participants on average spending slightly longer time per task compared to VR-VRT, resulting in the participants needing approximately 50% - 100% more time to complete the tutorial on the PC, relative to in VR.

As for the questions, 85,7% of the participants found the player movement more natural in VR-VRT and 100% found the VR version to provide the most fun. Asking the participants to compare the object manipulation, the results from the previous section remain relatively unchanged, with the consensus being that manipulating objects feels more fun and natural in VR, but with less precision.

When asked for general remarks on the strengths and weaknesses of both platforms we see largely the same ideas repeated, with VR-VRT being perceived as being more intuitive, natural, fun and engaging. In comparison, the participants found the PC-VRT to not trigger the motion sickness that can occur in VR, as well as having a much higher precision object manipulation system.

The results of the linear scale questions for this section can be found in Figure 15.



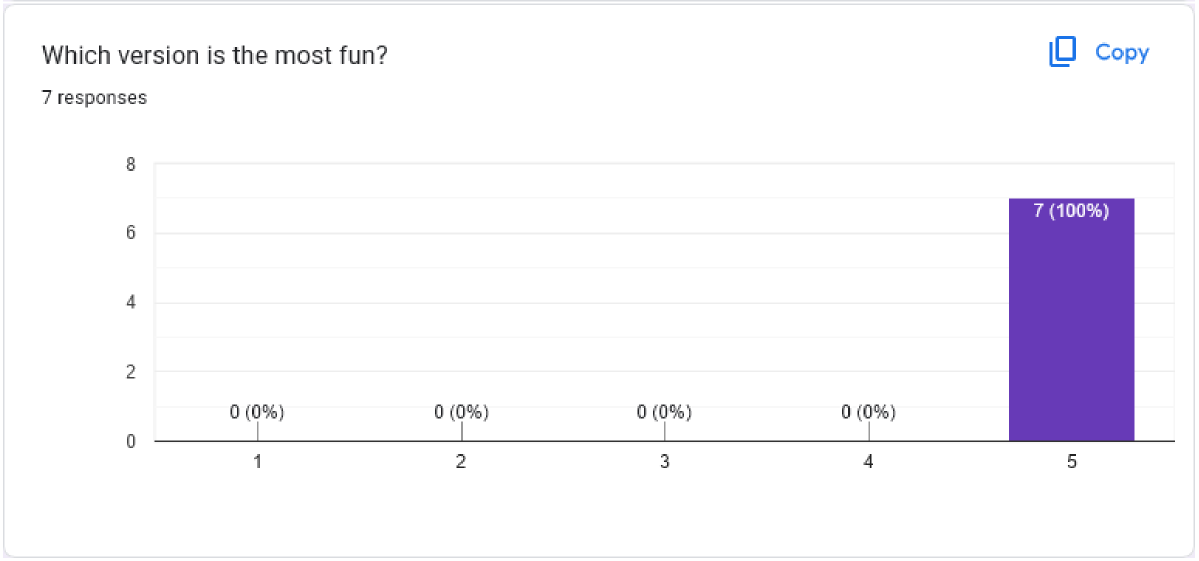


Figure 15: A summary of the results of the linear scale questions in the second section, with 1 indicating a substantial preference towards PC-VRT and 5 indicating a substantial preference towards VR-VRT.

5.2.3 Part 3 - General Questions

In the final section, the participants were asked general questions, all of which were optional, with the main goal of accounting for any encountered demographic based variance. All seven participants responded to each question in this section.

In terms of age, almost all of the participants (85,7%) were in the range of 21-26, with both their identified gender and highest achieved level of education being roughly equally split between "male" (57.1%) and "female" (42.9%), as well as "High School/Upper Secondary School/VWO/HAVO/VMBO or equivalent" (57.1%) and "University/University of Applied Sciences/WO/HBO/MBO/equivalent or higher" (42.9%) respectively.

Perhaps the most interesting results of this section are the levels of familiarity the participants reported having with technologies such as computers, XR devices and software related to 3D modelling. As can be seen in Figure 16, all participants reported being very familiar with computers, while also possessing some familiary of XR devices, but a low familiarity with 3D modelling adjacent software, indicating another possible disadvantage for VR-VRT.

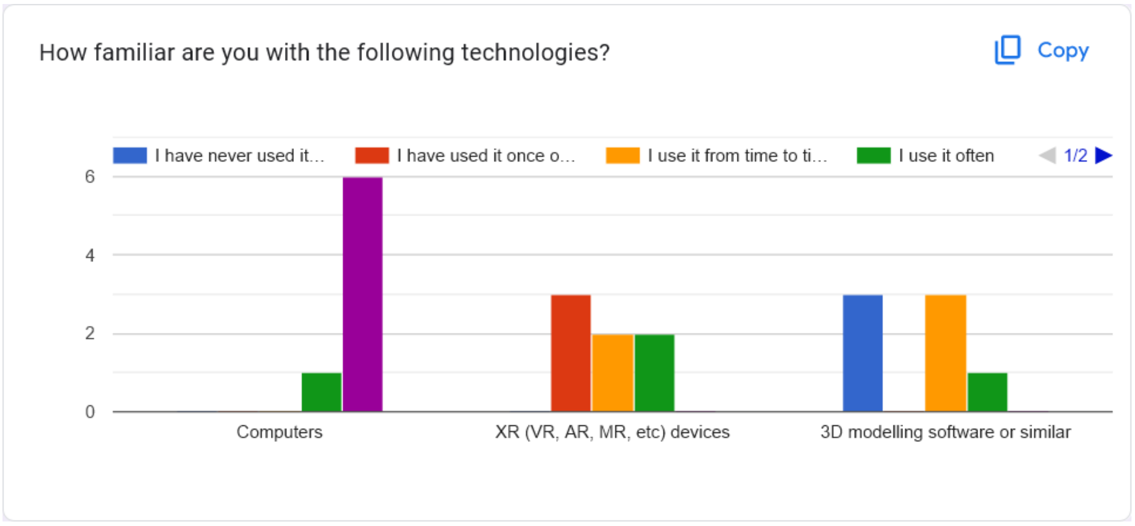


Figure 16: A graph in the third section, displaying the participants' reported familiarity with computers, XR devices and software related to 3D modelling.

6 Discussion

In this chapter we analyze the limitations of VRT, based on the results of the user study.

As mentioned in Chapter 5, there were a total of seven participants in the user study. Gathering more participants was planned, however there were two roadblocks preventing the expansion. First off, two of the prospective participants wore strong prescription glasses, which cannot be used with the VR headset, resulting in an unsolvable incompatibility. The second roadblock was caused by a major delay, due to multiple external factors, which had a noticeable effect on the timeline of the project. Without these roadblocks, we estimate that the number of participants would have been around 15.

Discussing the user study results, we mentioned VR-VRT having six tutorial tasks for the first level, compared to PC-VRT which has 20, as well as the participants on average also spending less time per tutorial task in VR-VRT. These tutorial tasks cover the basics, mainly locomotion, object manipulation and UI interaction.

Due to the nature of VR controls, VR-VRT features heavily simplified locomotion and object manipulation controls, foregoing the multiple key binds used in PC-VRT that users tend to find confusing, based on observation during the user study. Because of these simplified and more intuitive controls, the tutorial can be heavily simplified, as seen in this project. When there is less time needed to acclimate to the controls, it allows the individual to spend their time focusing on the main goal of VRT itself, teaching ray tracing, resulting in an increased learning output. Additionally, based on the theory of gamification[7] the heightened level of fun present in VR-VRT also enables users to maintain focus for longer, increasing their learning ability.

When tasked with completing the tutorial in PC-VRT, multiple participants found themselves stuck on one tutorial task in particular. This tutorial task aims to teach users how to orbit the camera, which is performed by "moving your mouse while holding down left-click and left control or by using the arrow keys while holding down left control". What was found was that to orbit the camera, the users would in reality need to first hold down left control, which is then followed by either of the two options mentioned, instead of the other way around. However, due to slightly ambiguous wording, the users would hold down left control after performing the first action, which VRT does not recognize as valid input. This can be easily remedied by a quick rewording of the tutorial task of course, but the point is that PC-VRT in general features controls with this level of complexity. Something that VR-VRT particularly excels at. Some participants also found that panning the camera did not feel as intuitive as expected, due to it being controlled by the arrow keys, with said participants preferring the use of the W, A, S and D keys instead, as is usually the case in video games and similar software. In general PC-VRT could benefit from a general redesign of the controls.

As discussed in Chapter 5 regarding Figure 16, the participants were fairly familiar with computers, but less so with XR devices, and even less with 3D modelling adjacent software. Taking into account the inherent advantage of the PC setup, featuring twice the PPD and refresh rate compared to the VR setup, the original assumption was that VR-VRT would be a roughly equal match to PC-VRT. The idea was that the advantage of PC-VRT would be balanced out by the theoretically more human controls of VR-VRT. Even with the extra advantage, users heavily preferred VR-VRT, due to its theoretical strengths previously discussed, which in practice makes a strong impact. Those who had less familiarity with computers did on average have a higher preference towards VR-VRT, compared to participants who are very familiar with computers.

7 Conclusion

The goal of this project was to create a VR adaptation of VRT, in which users could experience VRT in a natural and intuitive environment featuring familiar controls and an easy to use UI. To achieve this, we explored, analyzed and implemented theory on VR game design with extra attention on UI design. Said theory also includes topics on locomotion, world interaction, audio and haptics.

The results of the user study corroborated the success in achieving said goals, with 85.7% of the participants finding the controls and UI easy to use, and 100% heavily preferring VR-VRT over PC-VRT in terms of fun. The participants found the object manipulation system to be easier and more fun in VR-VRT, but lacking the degree of precision present in PC-VRT.

From this, we conclude that VR-VRT is better suited towards public demos, light simulation use and for people who are less experienced with computers or technology in general, while PC-VRT is better suited towards more serious simulation and users who do not own a VR headset.

8 Future Work

- **High precision object manipulation system in VR-VRT**
  Currently, the object manipulation system in VR-VRT feels very natural, intuitive and fun, however with that comes a lack of fine controlled precision. Building on the work done by Bora Yilmaz on optimizing VRT[9], a way to improve on this could be the addition of an interactable axis-aligned bounding box (AABB), which could be grabbable by the edges or surfaces to limit transformations to one axis or plane respectively. Another option would be to port over the gizmos present in other versions of VRT. VR-VRT also features no way of entering precise numbers into the control panel, which would achieve roughly the same goal. Implementing that would add a more mathematical-oriented precision. Either way, any feature addressing this weakness of VR-VRT would be appreciated regardless of its form.

- **Undo option**
  During the user study, one participants mentioned wanting an undo button. The option to undo performed actions would be a good addition in terms of quality of life.

- **Visual hints**
  Currently, there is no indication on how to access the UI in VR-VRT. Adding a visual hint to indicate how to open and use the UI would help with that. There could also be visual hints for other aspects, such as for hovering over interactable objects. The tutorial would especially benefit from added visual hints, as this would add a great amount to the interactivity and ease of use of VRT.

- **Better camera controls in PC-VRT**
  During the user study (see Section 5), the participants found the camera controls in PC-VRT to be hard to grasp. Re-evaluating said camera controls might help make the controls of PC-VRT easier to learn.

- **Unifying all versions of VRT**
  With the debut of the VR version, there is now yet another version of VRT which is not compatible with the main code base. As discussed in Section 4, due to the nature of Unity, simply merging the VR version into the main code base might be enough to warrant its own project entirely, let alone merging all the other versions.

- **UI Redesign**
  The current UI works well, but leaves more to be desired. A redesign, perhaps focused on dynamicity and space efficiency would be a great addition. A window management system could also work well, as it would allow the individual user to customize the UI layout to their own liking. However, unifying all versions would allow a solid foundation for future additions.

- **Saving rendered images**
  The button to save rendered images currently does nothing. The ability to save rendered images to the VR headset would be a good addition.

- **Ray perspective mode**
  Individual pixels on the render preview can be selected to view individual rays. The option to follow the path of an individual ray, from the perspective of that ray would add to the learning potential and gamification of VRT.

- **Localization**
  Except for a special version which is also available in Dutch, all other versions of VRT are only available in English. Translating VRT to other languages might help educate foreign Computer Graphics students.

- **Saving progress**
  Progress is currently stored in memory, meaning progress is wiped when exiting VRT. The ability to automatically save the progress on a periodic basis would be a great addition to VRT.

- **Larger user study**
  The user study for this project was held fairly late in the development cycle. Due to time constraints, finding participants in time proved quite challenging. Multiple of the potential participants were also unable to partake due to wearing high strength glasses, making the sample size for the user study even smaller than anticipated. A larger user study could provide more insight on the quality of the project.
- **More animations and sounds**
  In terms of sound and animation, this version has parity with other versions. Still, it would be nice to have more of both to add further gamification and enhance the interactivity of VRT.
- **External Ray Tracer**
  As of now, VRT uses the built in ray tracer made by the original authors (Chris van Wezel[2] and Willard Verschoore de la Houssaije[3]). Allowing users to test, as well as modify their own ray tracer code within VRT would be a great addition, allowing users more freedom and flexibility.

Acknowledgements

I would like to thank my supervisors Jiří Kosinka and Steffen Frey for their support throughout the project, by quickly responding to questions and providing feedback. Additionally, I would like to thank Chris van Wezel for providing resources and answering questions. Finally, I would like to thank my friends, family and colleagues for their support and for participating in my user study.

References

[1] 10 Years in the Making: NVIDIA Brings Real-Time Ray Tracing to Gamers with GeForce RTX, 2018. `https://nvidianews.nvidia.com/news/10-years-in-the-making-nvidia-brings-real-time-ray-tracing-to-gamers-with-geforce-rtx`. Accessed: 05-03-2024.

[2] Chris van Wezel. A Virtual Ray Tracer. *BSc Thesis, University of Groningen*, 2022. https://fse.studenttheses.ub.rug.nl/26455/.

[3] W.A. Verschoore de la Houssaije. A Virtual Ray Tracer. *BSc Thesis, University of Groningen*, 2022. https://fse.studenttheses.ub.rug.nl/24859/.

[4] Nick Vitsas, Anastasios Gkaravelis, Andreas-Alexandros Vasilakis, Konstantinos Vardis, and Georgios Papaioannou. Rayground: An Online Educational Tool for Ray Tracing. In *Eurographics 2020 - Education Papers*. The Eurographics Association, 2020. DOI: 10.2312/eged.20201027.

[5] Christiaan Gribble, Jeremy Fisher, Daniel Eby, Ed Quigley, and Gideon Ludwig. Ray Tracing Visualization Toolkit. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, I3D '12, pages 71–78. Association for Computing Machinery, 2012.

[6] J.R. van der Zwaag. Virtual Ray Tracer: Distribution Ray Tracing. *BSc Thesis, University of Groningen*, 2022. https://fse.studenttheses.ub.rug.nl/27881/.

[7] Peter Jan Blok. Gamification of Virtual Ray Tracer. *BSc Thesis, University of Groningen*, 2022. https://fse.studenttheses.ub.rug.nl/27596/.

[8] Roan Rosema. Adapting Virtual Ray Tracer to a Web and Mobile Application. *BSc Thesis, University of Groningen*, 2022. https://fse.studenttheses.ub.rug.nl/27894/.

[9] Bora Yilmaz. Acceleration data structures for Virtual Ray Tracer. *BSc Thesis, University of Groningen*, 2022. https://fse.studenttheses.ub.rug.nl/27838/.

[10] Job Simulator — Owlchemy Labs — Early Access 2016. `https://www.indiedb.com/games/job-simulator/images/convenience-store`. Accessed: 03-07-2024.

[11] SteamVR. `https://store.steampowered.com/app/250820/SteamVR/`. Accessed: 01-07-2024.

[12] Meta Horizon OS. `https://en.wikipedia.org/wiki/Meta_Horizon_OS`. Accessed: 01-07-2024.

[13] What are the 4 elements of Virtual Reality? `https://www.engati.com/glossary/virtual-reality`. Accessed: 20-02-2024.

[14] How to enable Battery Saver Mode on Quest 3. `https://www.trustedreviews.com/how-to/enable-battery-saver-mode-meta-quest-3-4396202`. Accessed: 26-03-2024.

[15] OpenBrush. `https://openbrush.app/`. Accessed: 05-07-2024.

[16] VR Design Best Practices. `https://medium.com/@LeapMotion/vr-design-best-practices-bb889c2dc70`. Accessed: 04-07-2024.

[17] High Fidelity Tutorial: Learn to Create a VR Tablet App. `https://medium.com/@fayeli/high-fidelity-tutorial-learn-to-create-a-vr-tablet-app-cf81af46ecfd`. Accessed: 04-07-2024.

[18] Apple Vision Pro review: A revolution in progress. `https://www.tomsguide.com/computing/smart-glasses/apple-vision-pro-review`. Accessed: 20-02-2024.

[19] Exploring the Different Kinds of Locomotion in VR. https://xpertvr.ca/the-different-kinds-of-locomotion-in-vr/. Accessed: 20-02-2024.

[20] Boneworks - Next Gen VR Gameplay!, 2019. https://www.youtube.com/watch?v=GJ2lzV2LLwM. Accessed: 20-02-2024.

[21] Virtual Ray Tracer - An application to visualise the ray tracing process used in computer graphics. https://github.com/wezel/Virtual-Ray-Tracer. Accessed: 05-07-2024.

[22] Cornel Hillmann. *UX for XR: User Experience Design and Strategies for Immersive Technologies*. Apress Berkeley, 2021. DOI: 10.1007/978-1-4842-7020-2.

Appendix

User Study Questions

Part 1 - VR-VRT

Q1: How intuitive (easy) to use are the controllers?
R1: 1-5, with 1 being “Not intuitive at all” and 5 being “Feels like an extension of my arm”.

Q2: Is there anything about the controls/controllers that you would like to see changed (mapping, visuals, etc)?

Q3: Does the layout of the User Interface (UI) make sense?

Q4: How easy is it to navigate the UI?
R4: 1-5, with 1 being “Very difficult” and 5 being “Very easy”.

Q5: Is there anything about the UI that you would like to see changed (element sizing, layout, style, etc)?

Q6: How is translation (object movement), rotation and scaling for you? Does it feel natural?

Q7: How does the player movement system feel?
R7: 1-5, with 1 being “Very bad” and 5 being “Very good”.

Q8: Is there anything about the player movement that you would like to see changed (teleportation movement, snap turning instead of continuous turning, etc)?

Q9: In general, is there anything else that you would like to see changed?

Part 2 - Comparison With PC-VRT

Q1: How is translation (object movement), rotation and scaling of objects in VR compared to using the gizmos on PC?

Q2: Is there anything about the translation (object movement), rotation and scaling system that you would like to see changed in the VR version?

Q3: Which version has the most natural player movement? R3: 1-5, with 1 being “PC feels best” and 5 being “VR feels best”.

Q4: Is there anything about the player movement system that you would like to see changed in the VR version?

Q5: Which version is the most fun? R5: 1-5, with 1 being “PC” and 5 being “VR”.

Q6: Can you think of why you feel one of the versions is (not) better than the other?

Q7: Do you have any other remarks or feedback?

Part 3 - General Questions

Q1: What is your age?

Q2: What gender do you identify with?
R2: Male, Female, Prefer not to say, Other

Q3: What is the highest level of education that you have completed?
R3: University/University of Applied Sciences/WO/HBO/MBO/equivalent or higher, High School/Upper Secondary School/VWO/HAVO/VMBO or equivalent, Elementary School/equivalent or lower, Other

Q4: How familiar are you with computers?
R4: 1-5, with 1 being “I have never used it before” and 5 being “I am very interested in this and use it a lot”.

Q5: How familiar are you with XR (VR, AR, MR, etc) devices?
R5: 1-5, with 1 being “I have never used it before” and 5 being “I am very interested in this and use it a lot”.

Q6: How familiar are you with 3D modelling software or similar?
R6: 1-5, with 1 being “I have never used it before” and 5 being “I am very interested in this and use it a lot”.