university of groningen

faculty of science and engineering

Adapting Virtual Ray Tracer to a Web and Mobile Application

Bachelor Thesis

July 14, 2022

Author:
Roan Rosema

Primary supervisor:
Jiří Kosinka

Secondary supervisor:
Steffen Frey

Abstract

With the quick advancements of modern technology, ray tracing in computer graphics has become key in creating computer-generated images. To get a better understanding of how ray tracing works, the Virtual Ray Tracer desktop application was created. Rays are traced in scenes and the user can observe the paths these rays take. Virtual Ray Tracer is made with the intent of educating students and computer graphics enthusiasts about the art of ray tracing.

In this thesis, we adapt this existing application to both a functional web application and a fully functional and redesigned mobile device application. Instead of being required to download a desktop application, Virtual Ray Tracer can now be downloaded or used on different platforms. The application has become more accessible and this way a broader audience is reached.

A user study was conducted to retrieve feedback for the mobile application. The reported bugs and the retrieved feedback were used to improve the mobile application.

Contents

1 Introduction

In the recent decades, the use of computer graphics has become essential in many areas, ranging from the world of entertainment to the studies of scientific visualization [13]. Movies, games, but also weather depictions and the visualization of distant planets have all improved by a huge margin thanks to computer graphics. With the quick advancements in modern technology, graphics and visualizations have and will become more realistic than ever. As scenes often include various light sources, have many objects that are made of different materials and might have reflections to deal with, the use of ray tracing is a must in contemporary technology involving computer graphics.

Rendering is the backbone of many computer graphics images. For the last 40 years, rendering has been limited to the capabilities of hardware in computers. With recent performance improvements thanks to more processors and new programming methods, the interest in ray tracing technology has increased [11]. Dating back all the way to the middle ages, ray tracing is used to simulate the physical behavior of light [11]. The technique, although not necessarily the fastest, can create accurate and photo-realistic images [9]. Ray tracing is heavily used in real-time rendering, where scenes and objects change, making the rendering of these computationally complex. The challenges of accuracy, complexity and performance while also keeping the speed and quality high, is what makes ray-tracing in real-time rendering a big feat. Ray tracing is one of the few graphics rendering algorithms that is also easy to visualize, explain and code [9]. A small example of ray tracing can be seen in Figure 1.

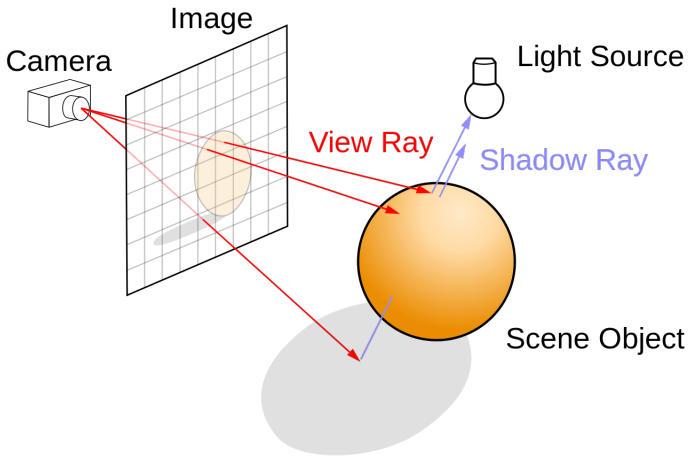

Figure 1: Ray Tracing example, By Henrik - Own work, CC BY-SA 4.0, `https://commons.wikimedia.org/w/index.php?curid=3869326`

The visualization of ray tracing is the main concept of Virtual Ray Tracer [15, 6], a desktop application made for anyone interested in computer graphics. The application is made with the intention to educate the user about the ray tracing algorithm. The application contains numerous scenes where the user can see the ray tracing algorithm visualized. Rays are traced in the scene, where the user can see the new paths the rays take after an interception with anything in the scene. Visualizing ray tracing in an application

will ultimately result in the user understanding the main concept of ray tracing better, without needing to understand the possible many difficult calculations.

Virtual Ray Tracer was only deployed as a desktop application. This thesis discusses the process in which Virtual Ray Tracer is adapted to a web application and mobile application. This way, the accessibility of Virtual Ray Tracer becomes broader and a wider audience is reached, resulting in more students getting to learn about the ray tracing process.

In Chapter 2 the existing applications and inspirations are discussed. In Chapter 3, the process of adapting Virtual Ray Tracer to a web application is discussed. Then in Chapter 4, the process of adapting Virtual Ray Tracer to a mobile application is discussed. Our user study is covered in Chapter 5. Finally, in Chapter 6 a conclusion is given and in Chapter 7 ideas for future work are mentioned.

2 Background

In this chapter, we explore the existing applications, inspirations and information that have been used to shape the results of this thesis.

2.1 Virtual Ray Tracer

The visualization of rays can be beneficial in the understanding of the way ray tracing works. The Virtual Ray Tracer application [15, 6, 16] is made with this exact thought in mind. With the ability to move objects, change their properties, move the virtual camera and much more, the application uses various features to educate the user on the ray tracing process. The application is made in C# with the Unity engine (2020.3 LTS Release), with the initial ability to deploy for Windows, MacOS and Linux.

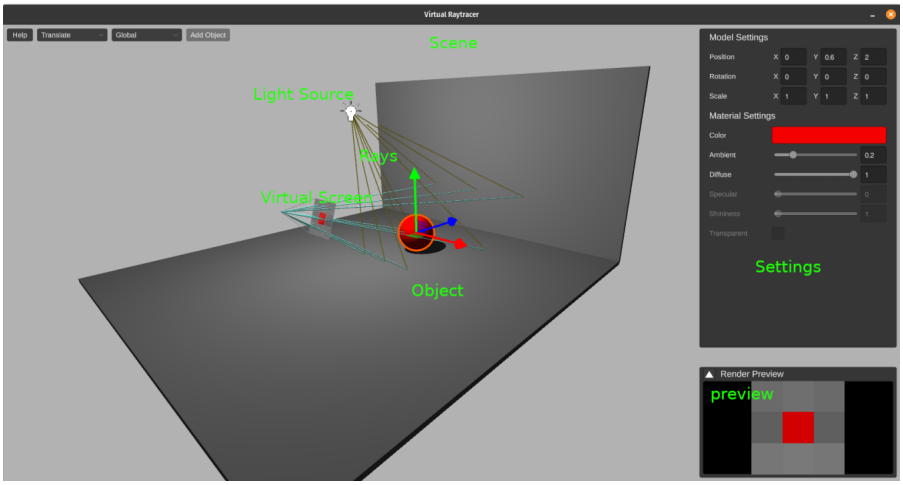

Figure 2: Virtual Ray Tracer desktop application [15, 6, 16]

Virtual Ray Tracer (see Figure 2) is implemented with three main classes at its core, the first of these being the Scene Manager. All scene data, such as the camera, lights and objects, are handled in this class. Ray tracing components may be attached to the objects in the scene, indicating that this object is either a camera, light or mesh. At the start of the scene, the Scene Manager will collect all instances of the ray tracing components and store it in a ray tracing scene component, after which the scene sends this data to the ray tracer.

The Ray Tracer class takes this scene data sent from the Scene Manager and produces a set of rays. Here, object intersections are determined and collisions are taken into account when producing the rays. The rays are stored as objects that store the origin, direction and length of the ray. One ray could potentially result in many more rays as ray tracing follows a recursive pattern, so the rays are stored in a tree. A new object is added as a child of the caller's tree after each recursive call. The output of the render function is a list of every tree node of the ray, with each tree node representing a pixel.

Finally, these rays are visualized with the use of the Ray Visualization class. The class

takes the list made in the Ray Tracer class and draws them in the scene. It takes the position, orientation and scales a cylinder to match the stored objects' origin, direction and length. As the rays are computed in recursive fashion, the length of the ray gets updated recursively as well, resulting in an animation achieving the effect of the rays shooting out of the camera into the scene.

2.2 Web Applications

Web applications are programs that are stored on servers and distributed through browsers on the Internet. They do not require to be downloaded and can easily be accessed through a network. Web applications are generally designed to increase accessibility. This is the main reason why lots of educational programs are programmed as web applications as opposed to desktop applications.

The use of web applications has its advantages when compared to desktop applications. The main advantages are listed below:

- More accessible than desktop applications;
- Updates are done via deployments on a web server, so no client-sided downloads are required;
- Most web applications can be used in any browser and are not dependant on the machine the application is run on.

However, there are also some drawbacks. They are listed below:

- Web applications cannot be started without an internet connection;
- Web applications are not installed on a client's device, but are distributed via the Internet. This also means that the application is accessible to practically all of the Internet. This, in turn, could result in a web application being less secure than a desktop application;
- In general, desktop applications are faster than web applications.

There are a few web applications where generated images produced with the ray tracing technique are shown, but none of these are focused on actually visualizing the rays' traces. A good web-based example is the application known as Rayground (see Figure 3) [17]. In Rayground, one can run their own ray tracing implementation and can test their own created scenes.

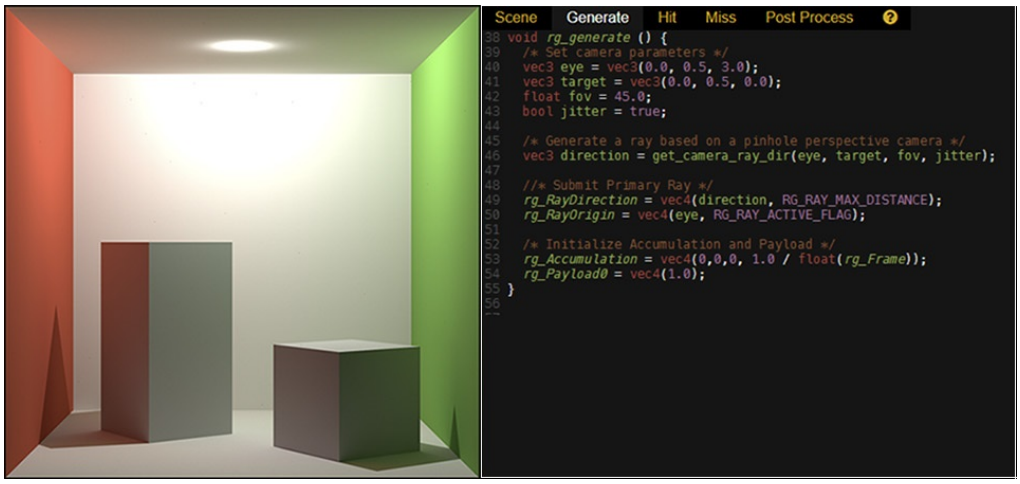

Figure 3: Rayground Web Application with a scene preview (left) and a code editor (right) [17]

Two other web based applications that are worth mentioning are Ray Optics Simulation [14] and the Light Tracer rendering tool [4]. Ray Optics Simulation simulates the ray tracing process in a 2D scene. The Light Tracer rendering tool allows users to create photo-real interactive 3D visuals right inside their web browser.

2.3 Mobile Applications

Mobile applications are applications that are run on mobile devices. They do not differ much from desktop applications, besides the screen size, memory capacity and processing power typically being smaller. As desktop applications, they are platform-dependant. Updates of mobile applications (as well as desktop applications) are known as patches and can be downloaded to the client's device whenever a new update is available. Most mobile applications do not require an internet connection to run and are usually highly secure.

As of today, there is a very small number of mobile applications where the ray tracing process is explained and there are no applications where the ray tracing process is visualized in 3D. Bringing Virtual Ray Tracer to mobile is therefore a nice addition to application stores, so Computer Graphics students and enthusiasts that would like a mobile way to learn about ray tracing can do so.

3 Web Adaptation

The first part of the project was to adapt the Virtual Ray Tracer desktop application to a functional Virtual Ray Tracer web application.

3.1 Unity WebGL Build

Unity supports porting an application to WebGL [7]. WebGL is a 3D graphics library/API, which allows browsers to efficiently render 3D scenes. The rendering is client-based; the scene is usually downloaded from a server, after which the processing of the scene is done locally using the client's hardware [5]. WebGL is developed with JavaScript [5, 18]. The well-known programming language for web development allows for easy communication between elements as opposed to using an applet. As WebGL is programmed in JavaScript, the applications are easily integrated with other JavaScript libraries and other HTML5 technologies [5].

JavaScript with the usage of WebGL is the logical option for porting a Unity project to a web-based application. A relatively recent new technology known as WebAssembly has the capability to run various programming languages, such as C/C++/C#, in the browser [12]. With the performance tested on multiple perks, among which loading times, handling HTTP requests and sorting a list, it can be concluded that WebAssembly has better execution times than a JavaScript applet with the same functionality when performing computations [10, 2].

Having done this research, the application was then built as a WebGL application. The realization then came that Unity also uses WebAssembly in its WebGL builds and not much else had to be done. It was, however, nice to see that it worked well. Complex scenes also ran fairly well (see Figure 4).

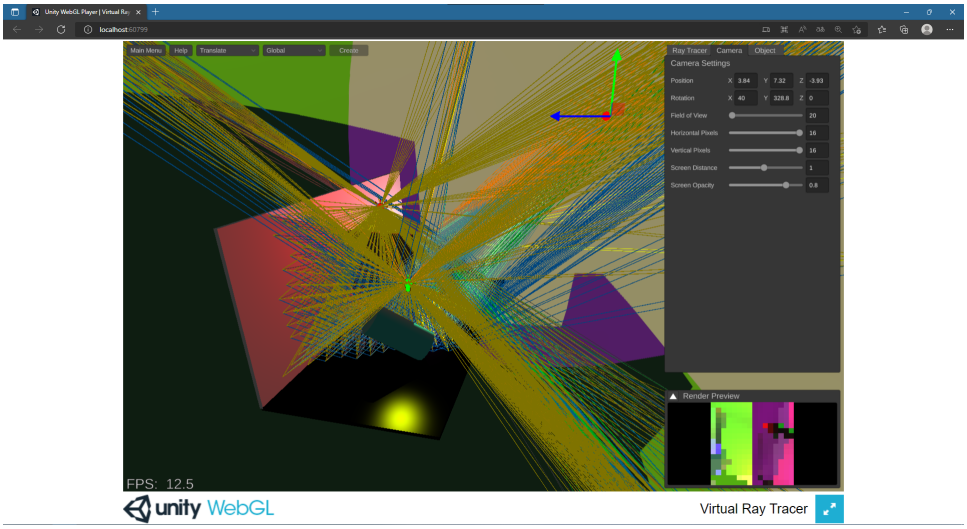

Figure 4: Virtual Ray Tracer as a web application

The main thought was that deploying the tool as a web application would come with

various optimizations needed. Fortunately, the application ran well enough to not need any optimizations in the original version. When any future work is added, optimization might still be needed.

Besides WebGL and WebAssembly, another web API is in the making, known as WebGPU. The API is currently specified as being a Working Draft, denoting that the API is still a work in progress [1]. WebGPU behaves quite similarly like WebGL, but enables developers to take full advantage of the capabilities of modern GPUs [3]. For now, Virtual Ray Tracer is only built as a WebGL application.

3.2 Github Pages

Four other students have been working on the application with their own projects; Jesper van der Zwaag on adding distributed ray tracing; Peter Jan Blok on the Gamification of VRT; Bora Yilmaz on adding support for acceleration data structures; Anton Bredenbals on adding support for ray marching.

With now having the knowledge that web builds can be built pretty easily, it was a good idea to make use of this to distribute the progress of these different projects. With the use of Github Pages, they were able to upload their WebGL builds and keep updating them whenever changes had been made. This way, it was prevented to having to download a new desktop application any time new changes had been made.

4 Mobile Adaptation

Besides deploying Virtual Ray Tracer as a web application, the application developer wished the tool to be deployable for mobile as well. Sadly, WebGL does not work particularly well (or not at all) on mobile devices [7]. Features like keyboard input also do not work, as a mobile device uses touches instead of mouse clicks and keyboard keys. One of the main concerns of deploying the application as a mobile application was performance, but although the performance is a little worse on mobile than on a computer, there are no big problems. Lots of visual changes were needed to create a functional replica of the existing application. Fortunately, this could be done in Unity.

4.1 User Interface

The main visual changes that had to be made in the application had to do with the User Interface (UI). All of the components had to be adapted to make the application readable and usable for mobile users. In the following subsections, most of the changes are explained. Besides tooltips, all of the UI elements within Virtual Ray Tracer are held in the Main Canvas object. Unity uses Prefabs, which are practically templates. If a change is made to a Prefab, all the scenes that have the Prefab as an object will also change. The Main Canvas is created as a Prefab. Most of the UI elements within the Main Canvas Prefabs are also Prefabs themselves. Changes made to the UI elements depended on which depth level the changes had to be made on.

4.1.1 Home Screen

The very first change that was made to the mobile version was the home screen. The home screen of the original Virtual Ray Tracer (VRT) desktop application looks good (see Figure 5), but was too small to nicely use on a mobile device.

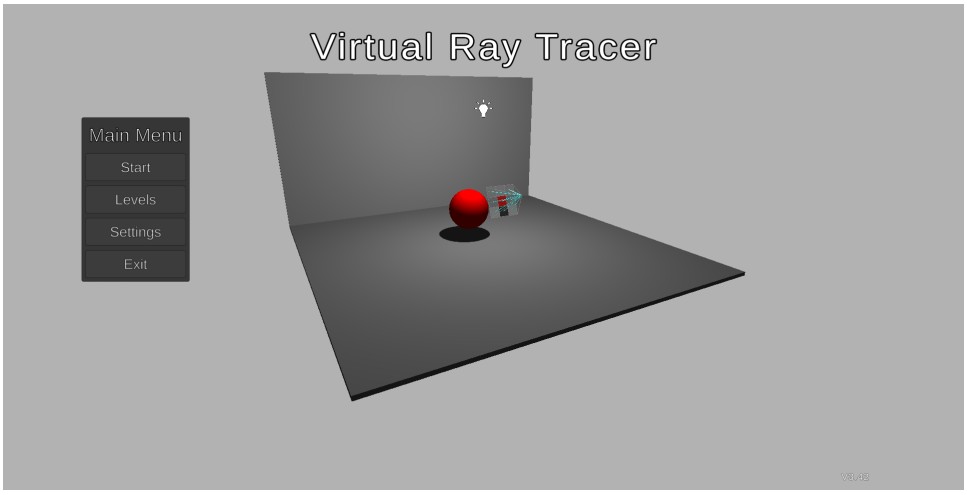

Figure 5: Home screen of the Virtual Ray Tracer desktop application

To have a good idea on how the home screen should look, inspiration was taken from various other mobile applications. A good example of this was Geometry Dash, a game that is played holding the device horizontally that also has a nice home screen for mobile users to use (see Figure 6).

Figure 6: Home screen of the mobile game Geometry Dash, used for inspiration

With the use of some inspiration, the home screen for the mobile version of Virtual Ray Tracer was made. The design was kept as simple as possible, to have a clear and concise home screen (see Figure 7).

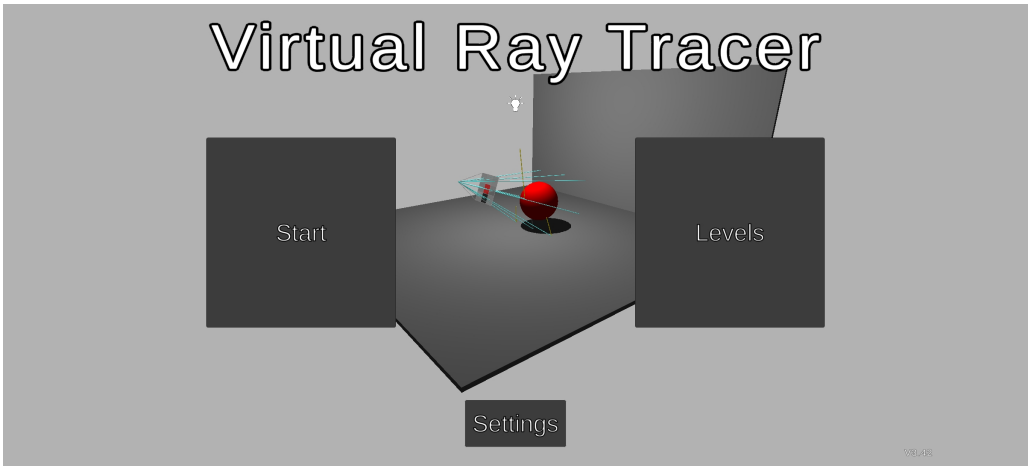

Figure 7: Home screen of the Virtual Ray Tracer mobile application

4.1.2 Main Menu and Help Menu

The two clear changes that can be seen in the mobile version of VRT are the Main Menu (see Figure 8) and Help Menu (see Figure 9). These menus and their texts within are made bigger. Just as the desktop application, these menus are relative to screen size.

Figure 8: Main Menu of the Virtual Ray Tracer mobile application

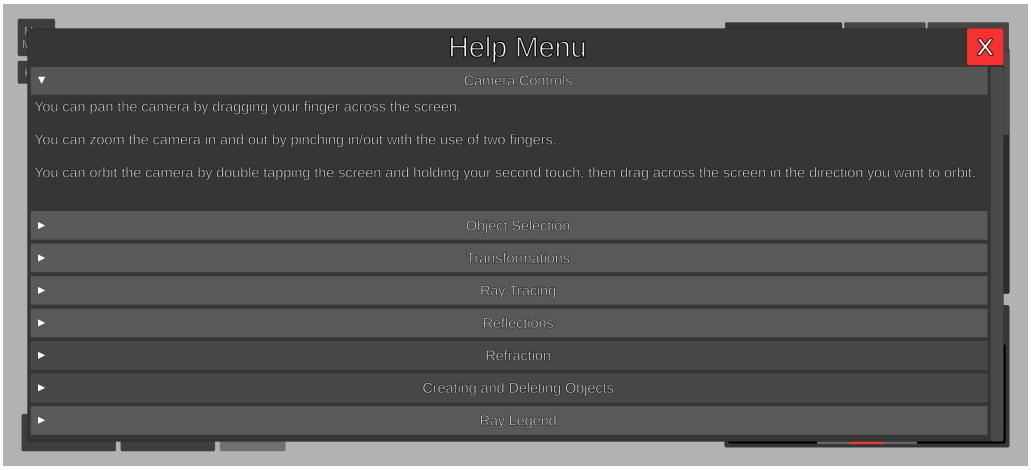

Figure 9: Help Menu of the Virtual Ray Tracer mobile application

4.1.3 Settings Menu

Just as the Main Menu and Help Menu, the Settings Menu has also been made bigger (see Figure 10). It includes the same buttons as the original desktop application.

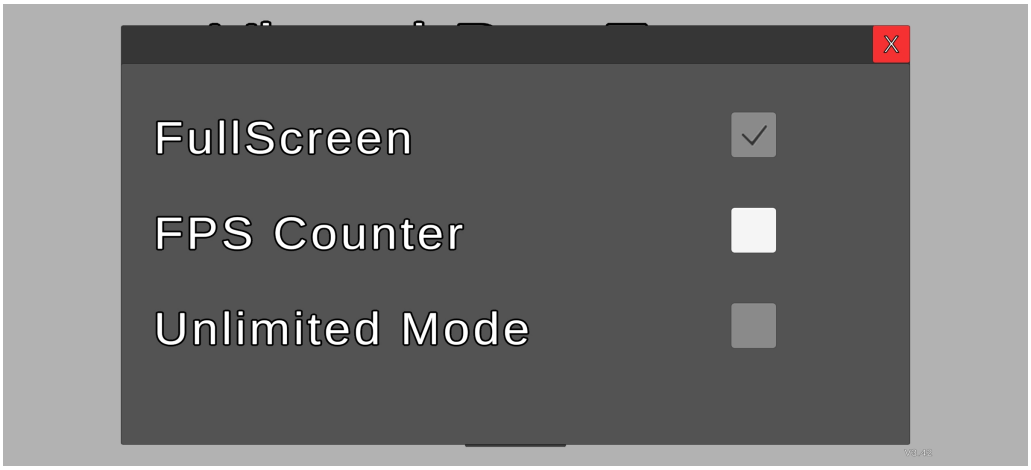

Figure 10: Settings Menu of the Virtual Ray Tracer mobile application

4.1.4 Levels Menu

The Levels Menu is also changed trivially, but notable to show (see Figure 11). As the other menus, the Levels Menu has been enlarged.

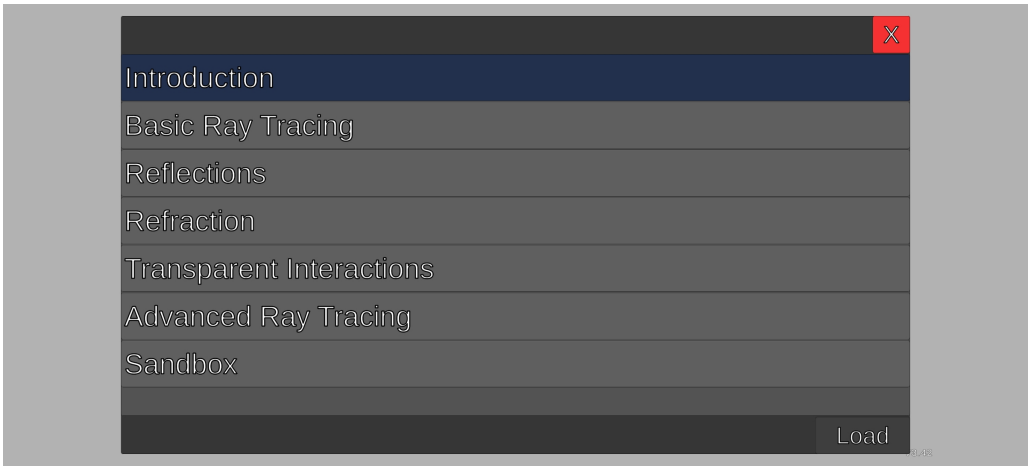

Figure 11: Levels Menu of the Virtual Ray Tracer mobile application

4.1.5 Menu Bar (Bottom Bar)

In the mobile version of VRT, the menu bar is split up into two new Prefabs; the Menu Bar (Bottom Bar) and the Top Bar. For nearly all levels, the Bottom Bar consists of three buttons: the Translate/Rotate/Scale button, the Local/Global button and the Create

button. All of these buttons now open an upward dropdown menu. In the Sandbox level, an additional Delete button is added. This is to delete any objects within the scene. In the desktop application this could be done with the delete button on a keyboard, but this is not possible on mobile devices. The Bottom Bar can be seen in Figure 12.

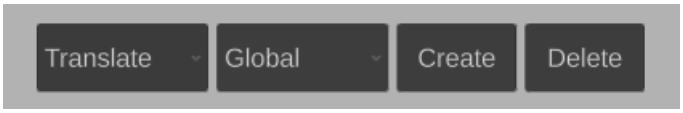

Figure 12: Bottom Bar of the Virtual Ray Tracer mobile application

4.1.6 Top Bar

The Top Bar consists of the Main Menu button and the Help button. They have been made smaller than the buttons in the Bottom Bar. Their purpose is self-explanatory. The Top Bar can be seen in Figure 13

Figure 13: Top Bar of the Virtual Ray Tracer desktop application

4.1.7 Control Panel

As mentioned before, some Prefabs have objects that come from other Prefabs. The Control Panel consists of various of these Prefabs. In the Control Panel, there are three tabs; Ray Tracer, Camera and Object. In reality, there are four different screens that can be shown within the Control Panel: Ray Tracer Properties, Camera Properties, Mesh Properties (for objects) and Light Properties. Each of these properties have settings that can change the scene.

There are four different types of setting changes. The types have been separated as Prefabs: `BoolEdit`, `FloatEdit`, `Vector3Edit` and `ColorEdit`.

Every setting that has been made from the `BoolEdit` Prefab is shown as a checkbox within the Control Panel. As the name suggests, these settings are attached to a boolean variable and can either be switched on or off. Visually, it looks nearly the same as the desktop application (see Figure 14).

Figure 14: Example of a setting made from the BoolEdit Prefab

In the mobile version of VRT, the settings that have been created from the `FloatEdit`

Prefab are shown as their desired name, an input field to the right and a slider at the bottom (see Figure 15). These settings are attached to a float variable.

Figure 15: Example of a setting made from the FloatEdit Prefab

The settings that have been created from the `Vector3Edit` Prefab are shown the same as on the desktop version, except they are a little larger (see Figure 16). These settings are attached to a Vector3 variable, such as position and scale.

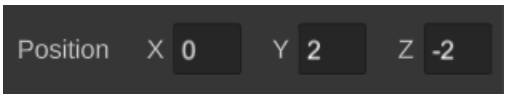

Figure 16: Example of a setting made from the Vector3Edit Prefab

The biggest change of these Prefabs is the `ColorEdit` Prefab. In the desktop application, the Prefab was relatively small. The color picker canvas was also shown below the clickable Color field. In the mobile version, this is changed. The color picker canvas is now shown to the left of the clickable Color field and is made relatively larger. Each slider and its text/values have been manually changed so that the entire color picker canvas is easily usable on mobile devices. After many versions and attempts to create a nice color picker canvas, the final one can be seen in Figure 17.

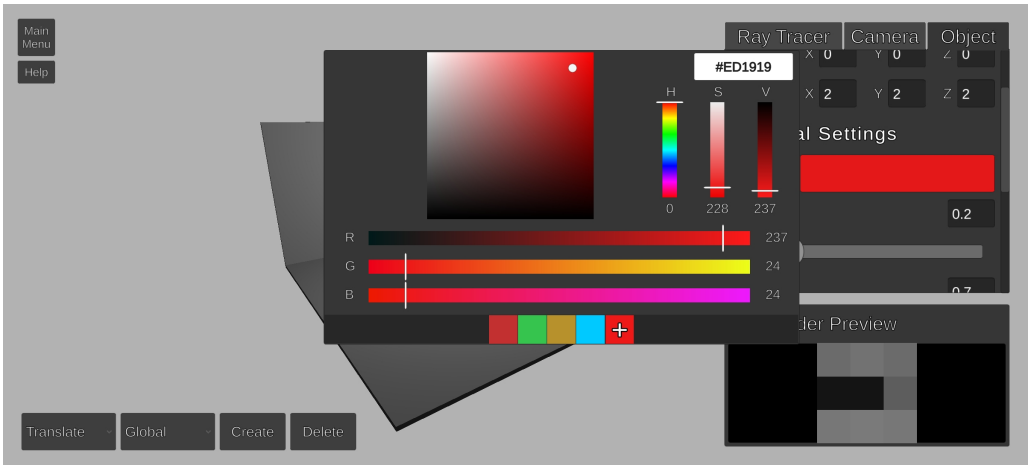

Figure 17: Example of a setting made from the ColorEdit Prefab

4.1.8 Rendered Image Window

The Rendered Image Window has also been made a bit bigger (see Figure 18). All functionality stayed the same. Within the Rendered Image Window it is still possible to zoom in on the image and pan around when zoomed in. More is explained in Section 4.2.

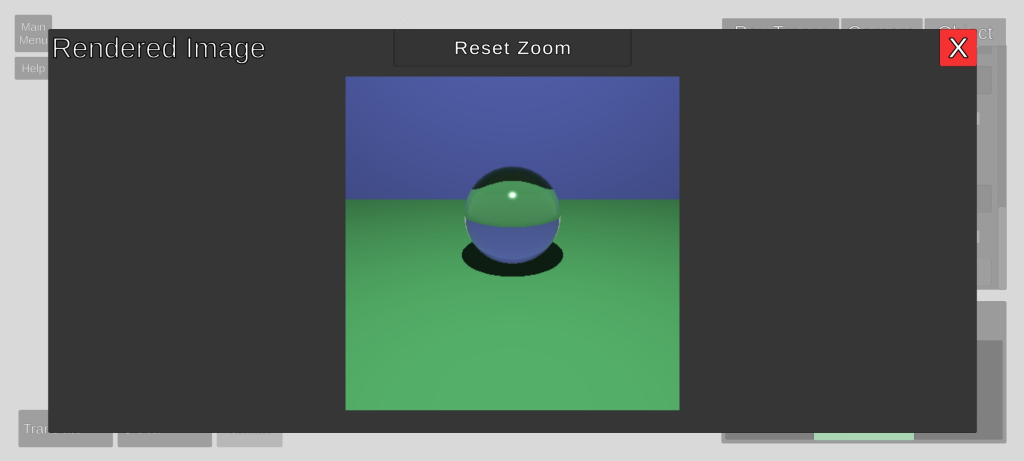

Figure 18: Rendered Image Window within the Virtual Ray Tracer mobile application

4.1.9 Gizmos

Gizmos are used to translate, rotate and scale any objects within the scene. Visually, the gizmos have not changed much. They have become larger and are in turn easier to use on mobile than before. The gizmos, however, each have their own hitbox which is not correlated to how they look visually. The hitboxes have been made a little wider and a little higher than what is shown on screen. As seen in Figure 19, when translating there are three planes. These had to be re-positioned when enlarged, as overlap would occur when not done so.

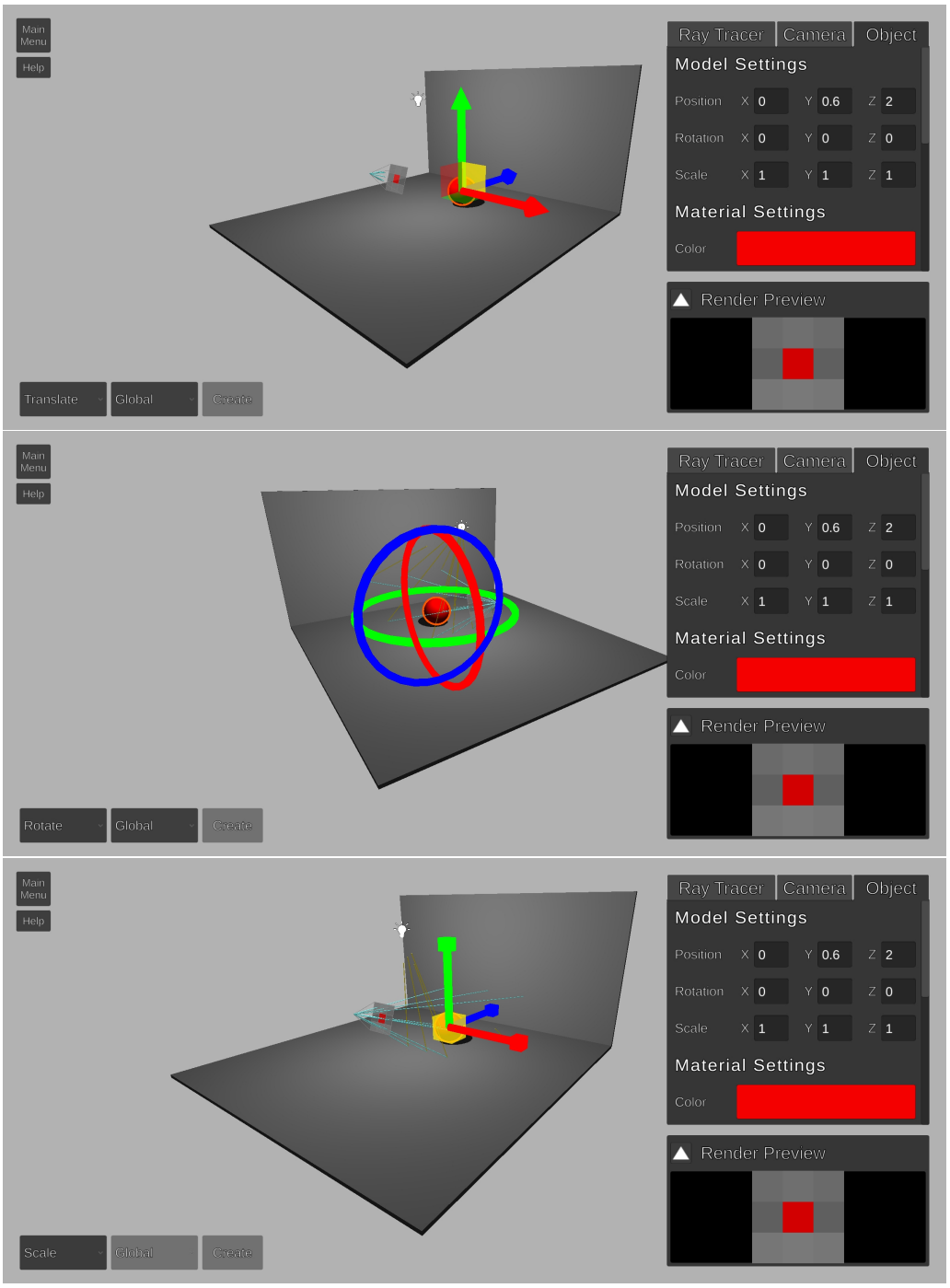

Figure 19: Gizmos of the Virtual Ray Tracer mobile application

4.1.10 Additional changes

Besides everything mentioned previously, there are some changes that have not yet been mentioned, ensuring the mobile application looks good. They are:

- Introduction panels of levels and its texts have been enlarged
- Render Preview proportions changed a little
- Changed text that did not apply to the mobile version of VRT
- Padding on mobile for devices with rounded screens
- Light hitbox was made bigger to make it easier to select
- Tooltip made bigger and to the left of touch
- FPS Counter position changed to not overlap with UI for each level

For some time, the UI was unusable as whenever a user tried to do anything within the UI, the touch would go 'through' the UI and select anything that was behind this UI component. A small check had to be added to check for a touch being inside of UI.

4.2 User Experience

Besides all of the UI changes, some changes for the User Experience had to be made. Some parts are not portable to a mobile application without change; the most obvious being, as mentioned, input methods [8]. Mobile devices make use of touchscreens as opposed to keyboard or mouse click inputs. The application has three ways to move within a scene: panning, zooming and orbiting.

The way the code is written, all original functionality and usages with respect to movement that the desktop version has, is still intact. If someone wants to use the mobile version as a desktop application, they would be able to do so. Keeping the original functionality intact was mainly used to find any bugs related to the mobile version.

4.2.1 Panning

In the desktop version of VRT, panning was done with the use of the middle mouse button on a mouse or with the use of the arrow keys on a keyboard. Obviously, these methods cannot be used on mobile devices.

Panning on the mobile version of VRT is done using one touch and dragging this touch across the screen. Equal to the method of the middle mouse button dragging, it keeps track of the starting position of the touch and determines its delta position when the touch has moved. It then pans the camera accordingly. When the user drags across the screen, any selected objects are deselected.

4.2.2 Zooming

Zooming within the desktop application could be done in two ways. A user can use the mouse's scroll wheel, or hold the Ctrl button on their keyboard, hold the right mouse

button and drag forwards or backwards. Again, both of these methods are unusable on mobile devices.

Zooming within the mobile version of VRT is done with the use of two touches. Whenever two touches are detected, it prevents the user from panning. This could otherwise result in some unwanted movement within the scene. The application keeps track of the two touches' starting positions and whenever they get closer or further away from their starting positions, they zoom in or out.

4.2.3 Orbiting

In the original desktop application, orbiting was done by either holding the Ctrl button, holding the left mouse button and dragging, or by holding the Ctrl button and using the arrow keys. These are also not possible on mobile.

As the panning uses one touch and zooming uses two touches, some experimenting had to be done on how to make the orbiting functional for mobile devices. At first, the use of two touches was tried but it was hard to differentiate between wanting to zoom and to orbit. Secondly, orbiting was implemented with the use of three touches. This, however, did not feel intuitive and was not nice to use.

Per a suggestion by my supervisor, it was best to try double tapping the screen to enable the orbiting. Currently, users are able to orbit by double tapping the screen and holding their second tap, and then dragging that touch in the desired direction.

4.2.4 Rendered Image Window Movement

Panning and zooming is not only used within scenes, but can also be used to zoom into rendered images. Fortunately, with some small changes, the method used to zoom within the scene could also be used as a method to zoom into rendered images. When zoomed in, the user can pan around the image. This was already built in. Unfortunately, the zooming in/out is sometimes a bit buggy, but a fix could not be found.

4.3 Google Play Store

The idea behind the creation of the mobile application was to reach a broader audience. The best way to do this was by uploading the application to the Google Play Store.

To be able to upload the application to the Google Play Store, various steps had to be taken. First, a Developer account had to be made for Google Play Console. Necessary information had to be filled in and after that, it was required to pay a one-time registration fee. After the registration had been processed, the app was created in the console.

First, all info about the application had to be filled in, such as the title and descriptions of various lengths. After that, screenshots of the application and nicely made thumbnails to get users hooked were added.

After all of the information was filled in, the app bundle including the .apk file was added to the release. Then some more mandatory info, like pricing and content rating, had to be filled in.

The final step was to review everything and check if all of the information was correct. The application was then rolled out to production. The application was 'In review' for about five days, after which the application became available for everyone in the selected target countries.

As most logos, the Virtual Ray Tracer application logo and feature graphic have been kept very simple, as can be seen in Figure 20. This was done to make the application recognizable.

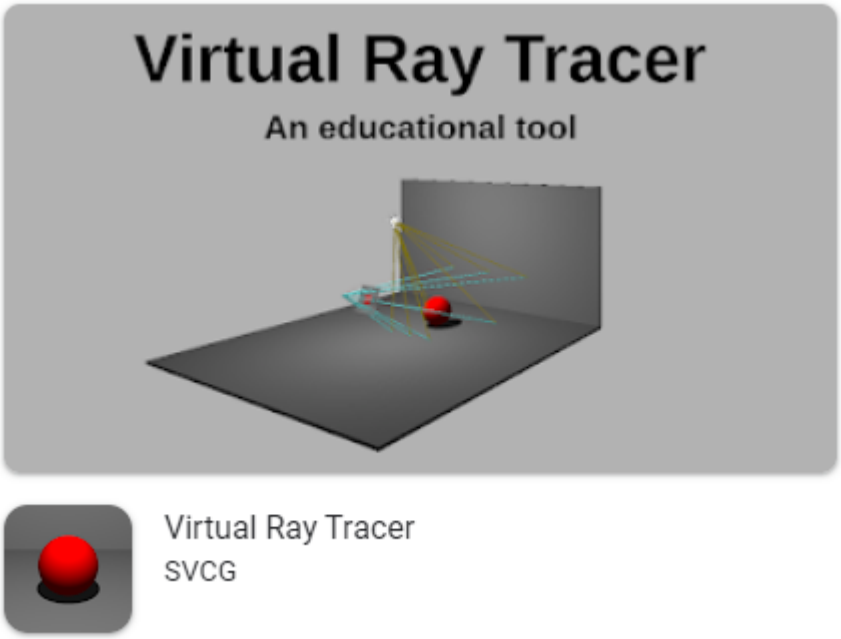

Figure 20: Virtual Ray Tracer application as seen in the Google Play Store

5 User Study

In this chapter the user study is explained. In this user study, the focus is fully on the mobile version of Virtual Ray Tracer. The questions that were asked in the study are mentioned and the notable results are discussed. The full list of questions can be found in the Appendix.

5.1 Questions

The participants of the user study were asked to answer a series of questions about the mobile version of Virtual Ray Tracer. The goal of this study was to get feedback from participants, such that changes can be made to ensure a pleasurable experience for participants of all backgrounds. The usage of the mobile version of Virtual Ray Tracer has to be as intuitive as possible. Unfortunately, this user study was only taken on participants that use an Android device, as the iOS build had not yet been compiled and distributed. The user study had a total of 15 participants; some with a background in Computing Science/Computer Graphics and others with other backgrounds.

5.1.1 User Interface Questions

In this section of the user study, participants are asked about the various User Interface changes that were made compared to the original version. These questions had to do with the functionality of the UI components, as well as the participant's opinion about the styling of these components. Questions about the components that contain text also ask about readability. The UI components that are mentioned:

- Introduction panel
- The Main Menu button and Help button
- Main Menu
- Help Menu
- Ray Tracer Settings
- Camera/Object/Light Settings

Another question is asked about the usage of the gizmos within the application. These gizmos are used to translate/rotate/scale objects.

The responses to these questions were used to alter the components whenever a great amount of people did not like a component, the component was buggy/not working or had good feedback.

5.1.2 Movement Questions

In this section, participants are asked two questions about the movement within the application. As users are now required to use their fingers/touches to navigate around the scene instead of using a mouse or keyboard input, it is required for it to work as expected.

The movement within the application should be as intuitive as possible, so whenever a participant had any difficulties with any of the movement features, I would look at the possibilities to change these features.

5.1.3 General Remarks

In this section, participants are offered the option to leave any general remarks or feedback. These remarks could then be used to tweak any functionality in the application to further improve the user experience.

5.1.4 General Questions

In the last section of the form, participants are asked for general information about themselves. Participants were allowed to skip any of these questions if they preferred to do so, as some questions may be too personal. The questions that are asked are mainly about age, education and how skillful they would consider themselves to be with computers. This way we can differentiate among the participants and figure out what group of participants had more difficulties and where they had these difficulties. The user study was not that extensive, so great assumptions about a certain group cannot be concluded for certain, but a good indication is given.

5.2 Results

As previously mentioned, a number of 15 participants have partaken in the user study. Most of these were friends and family, some of these were other students that have taken the Computer Graphics course at the University of Groningen. The general questions are discussed first, as they are useful to determine what groups of people had difficulties and where they had these difficulties. Unfortunately, most of the questions are not in multiple choice form, so individual results had to be looked at explicitly. A nice overview of the answers can therefore not be made.

5.2.1 General Questions

The participants of the user study can be split up into two parts considering their age. Eleven of the participants were between 19–22 years old, whereas the other four participants were older than 45. Notably, most of the participants older than 45 were also the participants with less experience with computers or mobile devices, with three of them filling in a 6 or lower out of 10.

Education of the participant was also asked about, but this had no notable effect on the participant's experience within the application compared to participants with different educations. Most of the differences were seen between groups split by age or computer experience.

The two groups will now be referred to as Group 1 and Group 2, where Group 1 is the group of the 19–22 year participants and Group 2 is the group of the participants older than 45.

5.2.2 User Interface Questions

A difference in the answers of the two groups can already be seen in the first question. 100% of the participants in Group 1 found the text in the introduction panel to be clearly readable, whereas only 50% of Group 2 found that to be the case. This difference in groups was useful to adapt the application to be as useful and intuitive for all future users.

Questions about the Main Menu, Help Menu and its corresponding buttons were positively answered by all of the participants. Overall, they are sized well, clear and easy to navigate. Some mentioned the buttons in the top left of the application to be a bit on the smaller side, but it did not bother them too much.

The next question was about the translating, rotating and scaling of an object. No particular differences were found between Group 1 and Group 2. 60% of the total participants found it to be doable and had no further remarks on the matter. The other 40% mentioned they had minor difficulties with either selecting the object, or selecting any of the gizmos. One participant mentioned that they sometimes found it difficult to see objects whenever they tried to make adjustments to it. This, however, can hardly be changed. Making the gizmos smaller would result in the users having a hard time being able to click the gizmos and making the gizmos bigger would only result in the object being hidden even more. No changes had been made as the overall feedback was positive.

The last two questions of the section were about the different Settings screens within the Control Panel. All of the sliders, checkboxes and input fields were received well and none of the participants mentioned any difficulties with these. There were, however, some mentions about the scrolling within the Control Panel. When using the scrollbar on the right, scrolling seems to work well but when dragging within the Control Panel it is a bit more difficult. Weirdly enough, this scrolling problem mainly occurs within the Camera Settings tab. A fix for it has not been implemented/found as of yet.

5.2.3 Movement Questions

The first movement question mentioned the panning, zooming and orbiting within a scene. All of the participants said that this works as expected, although there was some feedback from some of the participants. One of the participants mentioned that the orbiting was not properly explained and the help text was outdated. Furthermore, an object stays selected whenever the user starts panning across the scene. Both of these issues were quickly fixed and some of the other participants tested a version where this had already been fixed.

Four participants mentioned that the orbiting was a bit tricky sometimes. Two of these participants were from Group 1 and the other two from Group 2; 18% of Group 1 and 50% of Group 2. It is to be expected that the group with less experience with mobile devices had a harder time using the application. The orbiting method, however, has not been changed as there is no apparent better intuitive way of handling this.

The second/last question of this section was about the zooming and panning of a rendered image. 27% of the participants said that it does not work well. The other 73% said that it did work well on their devices. Elaborations were not asked in this question, but the zooming problem is known and is unfortunately still present. This small bug is luckily only visual and has no other consequences for the user's experience.

5.2.4 General Remarks

The users were asked to give any other feedback or remarks about the application if they had them. Whereas 73% of the participants either had no feedback or wished good luck for the thesis, the other 27% had found some bugs or had suggestions for the application.

Two of the participants mentioned that they found it difficult to understand most of the text, as all of it is in English and they are only able to speak Dutch well. Both of these participants were of the older generation and although this is not the target audience, it is still a good suggestion.

Another user mentioned that most of the application is very bland and would rather have it a bit more colorful. This feedback is reasonable, but as the Virtual Ray Tracer application is mainly for educational purposes the colorfulness of the application should not have to be changed much and could potentially even be distracting.

Lastly, another user found a bug that had to do with the color picker. When using the color picker to select a color, it could sometimes occur that a gizmo behind the UI would be selected. Users would have been able to accidentally drag this gizmo when they did not intend on doing so. This has now been fixed.

6 Conclusion

The purpose of this thesis was to broaden the accessibility of the educational tool Virtual Ray Tracer by adapting the existing desktop application to a web and mobile application. Fortunately, porting the web application was very simple. With the use of the WebGL build functionality within Unity, a folder of data could be generated which then can be used to add to any website that makes use of HTML. With the use of GitHub pages, the other students working on their projects related to Virtual Ray Tracer were able to distribute their projects more easily as well.

To achieve a functional version of Virtual Ray Tracer for mobile devices, various changes had to be made. These changes include both the User Interface and the User Experience. Currently, all functionality that was available in the original desktop application has been adapted to work on mobile devices. This mobile version with its changes has been evaluated through a user study.

Looking at the overall results of the conducted user study, most of the participants were positive about the mobile adaption of Virtual Ray Tracer. Most of the User Interface was received well and was said to be clear and easy to navigate. Most of the constructive feedback in this section of the user study was given by the older participants of the user study and it was helpful to know what (smaller) problems they had with the application.

The responses of the questions about the movement within the application had some more mentions of difficulties than the User Interface question's responses had. Most of the users giving feedback had mentioned the orbiting to be a bit difficult. It was, however, good to see that most of the users had no particular problems with the application when they had gotten used to it.

The final remarks of the user study also had some great feedback of which most have been fixed or implemented into the Virtual Ray Tracer mobile application.

7 Future Work

In this chapter, some additions/improvements to the Web/Mobile Virtual Ray Tracer versions are discussed:

- **Full integration of projects**
  As previously mentioned, four other students have been working on Virtual Ray Tracer. These other projects include many new subjects that were not present in the original application and having these in the mobile version of Virtual Ray Tracer would be a very good addition. One of the projects has made the Virtual Ray Tracer application more interactive. It would be great to have their changes merged as one and then adapted to be usable on mobile devices. Suggested by the other students, this may be possible with the use of Prefab variants, a functionality within Unity that lets developers differentiate between Prefabs depending on which platform the application is used on. Unfortunately, this does take some time and the time for it has not yet been found.

- **Image saving**
  A nice addition to the application would be the ability to download the rendered images to the mobile device.

- **Rendered Image Window**
  The zooming within the Rendered Image Window is sometimes a bit buggy. It does work, but it is not as smooth as it should be at times. A fix for this would definitely be nice to have.

- **Scrolling issue**
  As previously mentioned, scrolling within the Camera Settings of the Control Panel is a bit difficult as opposed to the other settings. A fix for it has not yet been found and implemented, but this would be very good to have. This issue also occurs on the desktop and web version, but this is not as critical to fix for those versions as most screens are big enough to have the entire Camera Settings on screen without having to scroll.

- **Translated Versions**
  As mentioned in the user study, a nice addition to the application would be the ability to switch languages. Although the main aim of the application is to educate students about ray tracing who are experienced in English, some other users might want to learn the concepts in their own language.

- **Larger user study**
  Due to the in-progress application builds that were used for the current user study, there were some differences in answers. It was also hard for 'random' people to trust someone sending an unknown file, so most participants have been asked in person. In the current user study, the focus was mainly on feedback of the application itself. In further research it would be interesting to set up a larger user study with more questions and a larger group of participants and finding out whether users enjoy the mobile application as opposed to the desktop/web version.

Acknowledgements

I would like to thank my supervisors Jiří Kosinka and Steffen Frey for helping me by answering questions and providing feedback during the project. They responded very quickly which resulted in me being able to work continuously without getting stuck on something that needed feedback. I would like to thank Chris van Wezel for providing the resources and source code of the original Virtual Ray Tracer application. Finally, I want to thank friends, family and the other participants that took part in the user study.

References

[1] WebGPU (Working Draft). https://www.w3.org/TR/webgpu/.

[2] Rick Battagline. *The Art of WebAssembly: Build Secure, Portable, High-Performance Applications*. No Starch Press, New York, USA, 2021.

[3] Francois Beaufort and Corentin Wallez. Access modern GPU features with WebGPU. https://web.dev/gpu/.

[4] Denis Bogolepov and Danila Ulyanov. Light Tracer Render. https://lighttracer.org/app.html,

[5] Diego Cantor and Brandon Jones. *WebGL Beginner's Guide*. Packt Publishing, Birmingham, UK, 2012.

[6] W.A. Verschoore de la Houssaije. A Virtual Ray Tracer, BSc Thesis, University of Groningen, 2021. http://fse.studenttheses.ub.rug.nl/24859.

[7] Unity Documentation. Building and distributing a WebGL project. https://docs.unity3d.com/Manual/webgl-building-distribution.html.

[8] Unity Documentation. Porting a project between platforms. https://docs.unity3d.com/520/Documentation/Manual/HOWTO-PortingBetweenPlatforms.html.

[9] Eric Haines and Tomas Akenine-Möller. *Ray Tracing Gems*. Apress, Berkeley, CA, USA, 2019.

[10] Harvey Ray, Hanspeter Pfister, Deborah Silver and Todd A. Cook. WebAssembly as an alternative solution for JavaScript in developing modern web applications. *JCSI 13 (2019)*, pages 322–328, 2019.

[11] Jon Peddie. *Ray Tracing: A Tool for All*. Springer, Cham, Switzerland, 2019.

[12] Mike Rourke. *Learn WebAssembly: build web applications with native performance using Wasm and C/C++*. Packt Publishing, Birmingham, UK, 2018.

[13] Somnath Sinha and Aditi Paul. *Computer Graphics*. Alpha Science International Ltd, Oxford, UK, 2018.

[14] Yi-Ting Tu. Ray Optics Simulation. https://ricktu288.github.io/ray-optics/,

[15] C. van Wezel. A Virtual Ray Tracer, BSc Thesis, University of Groningen, 2021. http://fse.studenttheses.ub.rug.nl/26455.

[16] Willard A. Verschoore de la Houssaije, Chris S. van Wezel, Steffen Frey, and Jiri Kosinka. Virtual Ray Tracer. In Jean-Jacques Bourdin and Eric Paquette, editors, *Eurographics 2022 - Education Papers*. The Eurographics Association, 2022.

[17] Nick Vitsas, Anastasios Gkaravelis, Andreas-Alexandros Vasilakis, Konstantinos Vardis, and Georgios Papaioannou. Rayground: An Online Educational Tool for Ray Tracing. In Mario Romero and Beatrice Sousa Santos, editors, *Eurographics 2020 - Education Papers*. The Eurographics Association, 2020.

[18] Mitch Williams. *WebGL Hotshot: Create Interactive 3d Content for Web Pages and Mobile Devices*. Packt Publishing, Birmingham, UK, 2014.

Appendix

User Study Questions

User Interface Questions

Q1: After clicking 'Start' on the Home page, you will encounter an introduction panel. Is the text readable and clear?
R1: *Yes, No.*

Q2: After closing the introduction panel, you will see the first level of the app. Try making use of the 'Main Menu' and 'Help' buttons. Are these buttons too small/big or are they sized well? Please explain your answer

Q3: Are the buttons within the Main Menu clear and easy to navigate? If no, why?

Q4: Is everything readable within the Help Menu and is it easy to use? If no, why?

Q5: Go back to the scene and try to translate/rotate/scale the ball within the level. Was this doable? Please explain any difficulties you encountered.

Q6: Please navigate to the next level. Play around with the Ray Tracer Settings. Is everything clear? Does everything work well on your phone or do you have difficulties with any of the sliders/checkboxes/input fields.

Q7: Play around with the Camera/Object/Light Settings. Please let me know any difficulties you have encountered or any suggestions you might have.

Movement Questions

Q8: Try moving around within the scene (moving/panning/zooming). Does everything work as expected? If no, why?

Q9: After rendering an image, you can also zoom in and move around. Does this work well on your phone?
R9: *Yes, No.*

General Remarks

Q10: If you have any other feedback/general remarks, you can leave them here!

General Questions

Q11: What is your age?

Q12: How skillful would you consider yourself to be with computers/phones?
R12: *1–10, 1 being not skillful at all, 10 being very skilled*

Q13: What are you currently studying or what did you study in school?

Q14: What platform are you testing the application on?
R14: *iOS, Android*