UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Aleksander Nikolajev

Lighting Design in Virtual Reality

Master's Thesis (30 ECTS)

Supervisor(s): Ulrich Norbisrath, PhD

Tartu 2022

Lighting Design in Virtual Reality

Abstract:
With the growing advancements in technological prowess, it is important to consider an alternative to current-gen traditional solutions. One such case is the way architects visualize their projects using traditional software, such as Radiance and Rhino. Architects have been looking into ways to simulate light in VR. With that said, this paper provides an alternative view of visualizing such projects via the use of VR and game engines. The paper goes into detail on what benefits VR can offer over traditional software, as well as provide a baseline application for creating, altering, and experiencing created 3D buildings.

Keywords:
VR, Computer Graphics, Illumination

CERCS: T120

Valguskujundus virtuaalses reaalsuses

Lühikokkuvõte:
Seoses tehnogoligaalse võimekuse kasvavate edusammudega on oluline kaaluda alternatiivi praeguse põlvkonna traditsioonilistele lahendustele. Üks selline juhtum on kuidas arhitektid visualiseerivad oma kavandatud projektid traditsioonilise tarkvaraga, nagu Radiance ja Rhino. See paber annab alternatiivse võimaluse selliste projektide visualiseerimiseks VR-i ja mängu- mootorite abil. Dokumendis käsitletakse üksikasjalikult, milliseid eeliseid VR võib pakkuda võrreldes traditsioonilise tarkvaraga, ning pakkuda baasrakendust loodud 3D-hoonete loomiseks, muutmiseks ja kogemiseks.

Võtmesõnad:
VR, Arvutigraafika, Valgustus

CERCS: T120

Contents

1 Introduction

Technological advancements have been changing the way architects and engineers have approached their practices. Consequently, technology has become a key factor in project creation and completion [4]. Notably, architects have become to rely on software to visualize their building designs before their construction. This is done for various reasons - shortening the time of making designs, seeing the resemblance of the original shape and environment, and finally allowing the client to see how the finished product would look [7].

In addition to the aforementioned features, architects have also been using software to simulate daytime lighting. Some of those tools include Radiance[1] and Rhino[2], popular software that allow for 3D modeling and daytime simulation. Radiance, in particular, creates a visualization of lighting in design that may be displayed as color images, numerical values, and contour plots. These tools, however, do not allow the user to immersive themselves in their constructed environment.

A recent technological advancement includes Virtual Reality (VR) - a computer technology used to create simulated environments, allowing users to experience the three-dimensional space immersively and interactively[3]. Architects have been looking at VR to combat the problem of user immersion in concept building [1]. In addition, prior research has shown that using VR is beneficial in the designing phase of the structural system, as well as increasing user awareness of created design and its structure [1].

One of the big challenges in the space of simulating natural lighting comes from the hardware limitations of VR, as developing an application with real-time raytracing [6] is challenging. Real-time raytracing being a method to simulate the physical behaviour of light. Tools such as Radiance and Rhino work well for computationally hard simulations due to their usual output being a static image or data. With VR, providing the user with a physically correct rendering of shadows and daylight at a constant rate requires graphical computation power, which the current-gen hardware of Graphics processing units[4] (GPUs) cannot provide. For this reason, VR applications are developed via the use of game engines, which use a biased rendering method to achieve faster scene processing. In addition to providing an alternative to daylight simulation, game engines have the tools necessary to allow the user to interact with the simulated world in VR.

The aim of this thesis is to create a baseline VR application, that would allow users to create a building, place it anywhere in a simulated realistic location, and experience its location at different daytime intervals. In addition, the user will have the opportunity to alter the created design after its placement, seeing how the alternations would affect daytime illumination. This goal will be achieved by researching created solutions for

[1] https://www.radiance-online.org/

[2] https://www.rhino3d.com/

[3] https://www.marxentlabs.com/what-is-virtual-reality/

[4] https://www.intel.com/content/www/us/en/products/docs/processors/what-is-a-gpu.html

a similar problem, which can be seen in the Related Works chapter. Furthermore, the concept of the VR application and its use case will be described in the Design chapter. In addition, the VR application will be created via a game engine of a particular choice, the context of which can be read in the Implementation chapter. Lastly, the design analysis of the created VR application is examined, showcasing and explaining the design decisions - this can be seen in the chapter Analysis.

2 Related Work

The goal of this chapter is to provide with the necessary topics and research analysis for the reader. This allows for an easier grasp of the topics described throughout the paper, as well as educate the reader on various topics. Chapters 2.1 - 2.7 provide necessary background on the topic of VR and its uses. Chapter 2.8 describes the significant description of real-world daytime simulation and how it can be done. The last chapter, Requirements, gives an overview of what objectives should be fulfilled and focused on for the application portion of this research paper.

2.1 Virtual reality

In the definition given by the Encyclopedia Britannica, Virtual Reality (VR) is "the use of computer modeling and simulation that enables a person to interact with an artificial three-dimensional (3D) visual or other sensory environments"[5]. The said environment includes visual, auditory, and kinaesthetic modalities, allowing the users to visualize realistic environments [14]. The use of VR has been studied in relation to user task performance, psychological benefit, and building performance.
Research into understanding the reason behind the effectiveness of VR has been mostly focused on the concept of presence [16]. Bob G. Witmer et. al [19] defines presence as "the subjective experience of being in one place or environment, even when one is physically situated in another". In Virtual Environment (VE), presence refers to experiencing the computer-generated environment rather than the actual physical space [19]. To add to the previous statement, Martin J. Schuemie et. al [12] notes that both involvement and immersion are necessary for experiencing presence. Immersion is a psychological state characterized by perceiving oneself to be enveloped by, included in, and interacting with an environment that provides a continuous stream of experiences [19]. Involvement - psychological state experienced as a consequence of focusing one's attention on a coherent set of stimuli or related activities and events [12].
Various fields of study, including education, healthcare, entertainment, have given empirical evidence to demonstrate that VR experience leads to positive attitudinal and

[5]https://www.britannica.com/technology/virtual-reality

behavioral outcomes, brand recognition, product recall, and more [16]. All of these outcomes are argued to be a result of presence in the VE [12]. As already mentioned in the Introduction, architects have been looking at VR to combat the problem of user immersion in concept building [1]. With the reviewed research, it can be seen that traditional software tools, such as Radiance and Rhino, do not have the positive effects of VE.

2.2 Open Source and Education

The term "open source" refers to something people can modify and share because its design is publicly accessible [18]. In terms of software development, open-source software development is where other developers can participate in the developing process of the software [20]. In an article by OpenSource [18], the writers give multiple reasons why some people prefer using open-source software. The most relevant reasons for this thesis are the following - training and stability. First, training refers to the ability to help people become better programmers when using open-source software. This is a result of making the source code public, allowing users to easily study it and learn to make better software. Second, stability refers to users using open-source software for important, long-term projects. This is a consequence of the public distribution of the source code, creating insurance that developed tools and features do not disappear.

Open Source (OS) has been researched as an innovative solution in computer science education. Specifically, Nada Alasbali et. al. [2] made a systematic literature review on the advantages and challenges of using open source in computer science education. In their results, they highlight that the use of OS provides numerous benefits, such as better support for contextualized learning, acquisition of a wider range of skills, and increased student motivation.

Using OS in teaching also alleviates the problem of licensing. In an essay by Martin Zeilinger et. al [21], they mention how using non-OS software requires subscription models, which can be prohibitively expensive, increasing the need for the provision by universities. In their performed survey, respondents also reported a wide range of workarounds concerning direct experiences and potential copyright issues.

Due to the presented issues and evident advantages of OS, this thesis will be focusing its work on using only OS software and game engine for its development process. In detail, this thesis will be using the Godot Engine[6], as it is free and open-source software released under the permissive MIT license[7]. Due to Godot not having a subscription-based model and being OS, this thesis and its developed application will be available to the academic community for future research and expansion.

[6] https://godotengine.org/license

[7] https://github.com/godotengine/godot/blob/master/LICENSE.txt

2.3 VR Lighting and Luminance

Over the past years, virtual reality has seen many applications in multiple different industries. For example, equipping soldiers with virtual helmets to move around in simulated battlefields for training purposes, to designing interiors and furnishings in architecture-based solutions [11]. Recently, VR has been used in medicine as a method for improving the doctors' qualifications in performing surgeries using created 3D models from radiologic images [3].

On the topic of virtual reality lighting, Michelangelo Scorpio et. al. in their paper focused on the use of virtual reality for outdoor lighting design, with the motive to examine the benefit of using VR in lighting design [13]. With the use of the Unreal game engine and establishing a literature review, it was concluded that researchers agree on considering virtual reality as a promising methodology for investigating people's visual perception. In addition, the use of the Unreal game engine showed an effective result in their investigation of the evaluation of lighting in outdoor design.

Rafal Krupinski [11] researched ways to use VR to evaluate created 3D designs in terms of lighting and luminance. In his study, the goal was to develop a system that would allow to carry out calculations on luminance distribution on photometric and geometric data, as well as fulfill a survey to see whether users could distinguish real-life from a virtual one. In order to achieve their goal, the proposed method is to create a geometric model of a real object, taking into consideration the color component and object illumination from the light. With the created geometric model, it would be built on the system as a virtual scene, which is then rendered into a 180x360 degree spherical mode, as can be seen in Figure 1. The rendered image is then presented alongside the real photograph in VR on which the rendered image was based, with the objective to observe whether users could the rendered image from the real photo. The reason VR was used is to give the user the impression of full 3D and the depth of field in the image.

On the topic of luminance distribution on photometric and geometric data (the second objective of their paper), the authors note that the built virtual scene takes into account the spectral composition of the light, which illuminates the real object, measured with the spectroradiometer. This is done with computer software, which is adapted to convert the photometric data of luminaires into the luminance and color components assigned to the virtual model.

While no metrics on how complicated the computer simulation is or how long it took to render a real-life room was given, the author notes that lighting calculations can get quite complex based on the environment. With this in mind, the conclusion was two-fold - the users could distinguish the rendered image from the real one, but the simulation would fulfill its task of calculating the necessary statistics.

The proposed solution, however, is only limited to luminance calculations and limited user visualization. While this research presents some evidence that rendered photos could, with enough manpower and effort, look near identical to real objects, it can be

(a)

(b)

Figure 1. Panoramic rendering of (a) virtual and (b) photograph of a real room [11].

argued that just using photos is not enough in today's technological advancement. To add to the argument, users would want to see different perspectives and angles of the rendered room, not just one singular angle. In addition, with the current technological advancement in VR, other functionality can be added - such as walking in the created environment and its alteration, providing users with more tools for luminance statistics at different alterations. To summarize, the proposed solution to evaluate 3D designs in terms of lighting and luminance were successful, showcasing the possibility that VR can be used as a tool for data gathering in architectural works. Consequently, this reinforces the point that VR could be a suitable solution for architects when designing and modeling buildings.

2.4 VR Immersion

Belen Jimenez Fernandez-Palacios et al. [8] developed a visualization pipeline to exploit and access large heritage 3D contents in a user-friendly way with the help of VR. The goal of this solution is to provide an immersive experience for digitally reconstructed heritage scenarios, and then analyze if the use of VR adds any technological advantages. The authors have done that with the Unity game engine and natural interface devices, which allowed them to create detailed 3D models without losing quality in terms of geometry and textures. In particular, Unity and OpenNI were used for this project. For their VR tool of choice, they used Oculus Rift - a VR headset allowing users to replicate

the natural movements of the user's head, and giving an increased perception of the user within the VR environment. To assure their interface is user-friendly when using Oculus Rift, the authors did tests with other people from different age groups. With the tools selected and tested, the implementation was thus - the proposed method exploits the point centered in the main camera and moves according to the head movement. In addition, if the user looks at a specific part of the virtual scene, they access information about the said specific scene.
In their evaluation of the solution, the authors note that VR technology can offer scholars visitations to digital archaeological areas, enabling a non-destructive way of observation and studying. The only limitation they found at the submission of the paper was the following - upload time of large 3D datasets into Unity. In addition, they note that this issue will be solved with the developments in VR. With this in mind, current technological advancements can offer more than just walking and gathering information about specific heritage sites. These technological advancements can include real-to-life lighting visualizations, which as was already established in the Introduction, could provide a positive effect on people's visual perception of objects. To conclude this particular solution, VR technology has advanced to the point where the upload time of large 3D datasets can be improved. Their proposed solution achieves the goal of allowing users to immerse themselves in 3D environments and experience real-world architecture, but this type of application can be improved with modern technological advances to be more beneficial to users' immersion. Related to this thesis, this solution shows that VR has the capability to immerse the user, in addition to allowing for non-traditional educational methods in architectural observation.

2.5 VR Daylight Simulation

Mohammad Keshavarzi et. al [9] presented in their paper a VR tool for daylight analysis and 3D model visualization, using Radiance and Unity in conjunction - RadVR. Radiance is a suite of programs for the analysis and visualization of lighting in design[8]. It allows users to create realistic lighting given specific inputs. These inputs include geometry, materials, luminaires, time, data, and sky conditions. Given these inputs, users are able to see simulations of realistic lighting. That are works about using Radiance as an engine for realistic daytime lighting. As for Unity, Unity is a game engine that allows for the development of video games and simulations.
The goal of their paper was to create a virtual reality tool for daylighting analysis that combines qualitative assessments through immersive real-time renders with physically correct daylighting simulations. In order to achieve this goal, the authors created a system architecture that takes semantic 3D geometry as input and converts it to a Radiance geometry description with appropriate material properties. Once that is done, the RadVR

[8]https://www.radiance-online.org/about

project runs the Radiance engine in the background to prepare the primary scene within VR. For their game engine of choice, they use Unity3D, providing authors with many useful libraries and the ability to output high-frequency renderings in an efficient manner. In their solution, the user can change the direct sunlight position in a limited way, ie. with predefined positions. In addition, the user is allowed to spatially map the daylighting performance of the building with its geometrical properties. With their created implementation, they decided to test it with other students to conduct a survey. In brief, the authors note that some students saw the potential of RadVR as being helpful in understanding the variation of direct light patterns through time. Moreover, the participants found it useful to be able to move around in the building. In their conclusion of the survey, the authors mention that RadVR enabled a more enjoyable experience than other architectural daylight simulators, like DIVA for Rhino.
While the proposed solution looks impressive, the developed tool does not allow the user to interact with materials, altering the scene in question. The said problem, however, is not the most troubling one. The troubling factor is the lack of availability to use the tool. After extensive searching to find this tool on the internet, searches resulted in RadVR being inaccessible and impossible to test. The system architecture and the inclusion of Radiance for daylight simulation are important, and as was already discussed, could help the previous related works with some of their missing aspects. To conclude, the proposed application showcases that VR can be used in conjunction with modern architectural tools, like Radiance, to allow for precise daylight simulation. With the benefits argued in the Introduction for the use of VR, this solution showcases that VR can be a suitable alternative to traditional architectural tools used today.

2.6 VR applications on the market

The last related work, when it comes to VR application, that will be discussed is Arkio[9] - a collaborative spatial designer that enables anyone to create, collaborate and review using VR. The features that Arkio promises are designing interiors, virtual spaces, or game environments.
As can be seen in Figure 2, Arkio allows multiple users to connect to a singular environment, giving them the opportunity to alter the visualization, place objects, etc. While this is not a research paper, this is a developed application that is similar to the goals of this paper and because of that, there are certain lessons that can be learned from said application. Primarily, the fact that this application is not open-sourced. The ability to seamlessly alter buildings, objects, and colors, in addition to the placement of objects, would be a great benefit for future researchers who wish to study architectural applications in VR. Due to it being closed source, scholars have to use resources and manpower to create said features from scratch, or ask the company formally to get

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

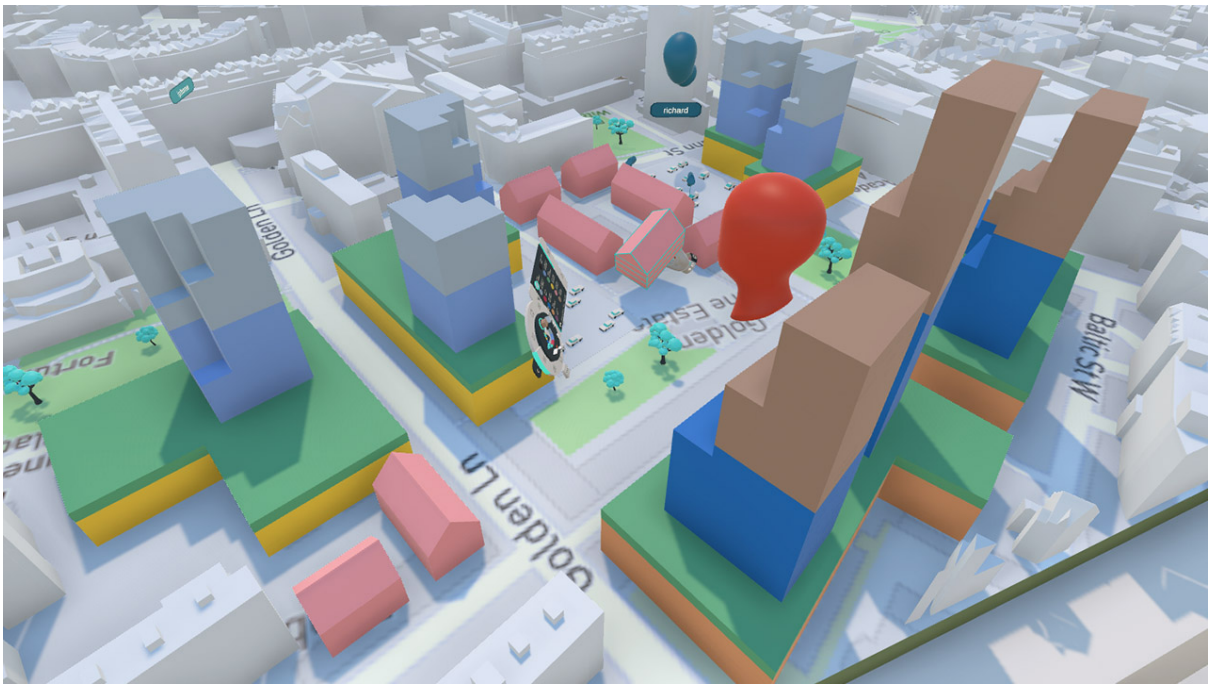

Figure 2. Arkio's environment, with multiple users placing houses and altering the environment. Note the two floating heads signifying that multiple users are located in the same session.

access to the application, which could involve constraints and/or the requirement to pay. As discussed in the Introduction, these deals could be prohibitively expensive to the educational faculty [21], and having a limited budget - not being able to afford it. To conclude, the application includes features that architects could find useful - the ability to create, shape, and alter a 3D environment. In relation to this thesis, this application showcases VR's ability to allow architects to use VR for their 3D design evaluation and alteration.

2.7 VR uses in different industries

VR can be and has been used in a variety of fields. For example, in medicinal education, VR has been used for surgical training in laparoscopic procedures. In addition, using VR to combine the teaching of radiology and anatomy demonstrated an easier understanding of the subject [3]. Other than in medicinal practices, VR has seen use in tourism. In a paper by Myung Ja Kim et. al. [10], they note that VR is an interactive digital-generated medium that enables partakers to create simulated experiences and an opportunity for tourists to experience spaces, attractions, etc., from their homes before making a decision to visit. Also, in their experiment, they noted that the use of VR had a positive experience and increase visit intention to their presented destination in VR.
In terms of the architectural field, it is claimed in an article called "The use of Virtual Reality in Architecture" [5] that VR could be used early as a conceptual stage to explore the potential impact of light on a room. They also note that there is still room for improvement before VR technology replaces the traditional architectural approach in the field of architecture.
Since the publication of that article, multiple research papers, some of which are reviewed in this chapter[3][10], have been done to showcase how VR technology progressed over the years. Featured innovations include simulated daytime lighting and illumination, as well as showcasing the design in 3D. The aforementioned features allow architects to measure the amount of diffuse, insolation, glare, and other metrics for the building. While there is much that has been made to measure different daylight-related metrics, little has been achieved to help architects experience their design in a 3D environment in an immersive and interactive way.

2.8 Sky Illumination

One of the objectives of this thesis, as was stated in the Introduction, is to allow users to experience a simulated real-world location at different daytime intervals. For this reason, it was researched on how best to simulate real-world environments when it comes to the illumination of the sky and/or sun. In a paper by Spitschan et al. [15], they note that illumination of the environment undergoes multiple different changes during a 24-hour cycle, both in terms of intensity and spectral changes. More importantly for this thesis,

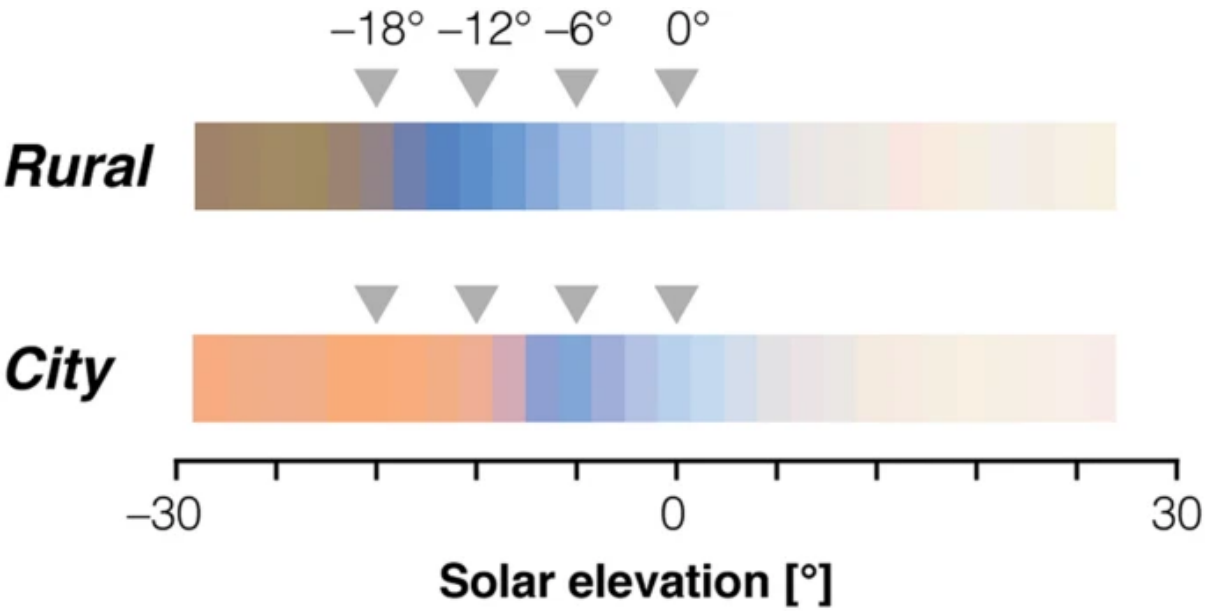

Figure 3. Pseudo-color rendering of illuminant spectra, depending on the solar elevation. Two bars are provided - Rural and City. Rural signifies the rural environment of the illuminant spectra. City - the city environment. [15]

they provide color renderings of illuminant spectra based on different solar elevations. For further context, the authors examined the chromatic variation of solar elevation, allowing a color rendering to be made.
As can be seen in Figure 3, a color rendering is provided depending on the position of the sun. Using the provided graph, one can attempt to simulate a real-world environment more accurately, altering the lighting environment based on different sun elevations.
Another way to interpret sky color is via the use of Kelvin. Kelvin defines a specific property of light that is emitted by a star or some other astronomical object. Using Kelvin, one can interpret the degree to which a body would have to heat to emit a certain light. In Figure 4, an interpretation of common color temperatures of light can be seen. Furthermore, Kelvin values can be attributed to each color temperature. In addition, Figure 4 provides some context to what color temperature corresponds to a certain part of the day. For example, based on Figure 4, it can be seen that the overcast sky is somewhere around 8000 Kelvin, giving a light blue coloring. Intuitively, these Kelvin values can be used to better simulate the real-world environment and give a more immersive feel.

2.9 Requirements

Throughout this chapter, multiple research topics were showcased and reviewed. Their potential use-cases were presented and their drawbacks were noted. In addition, relevance

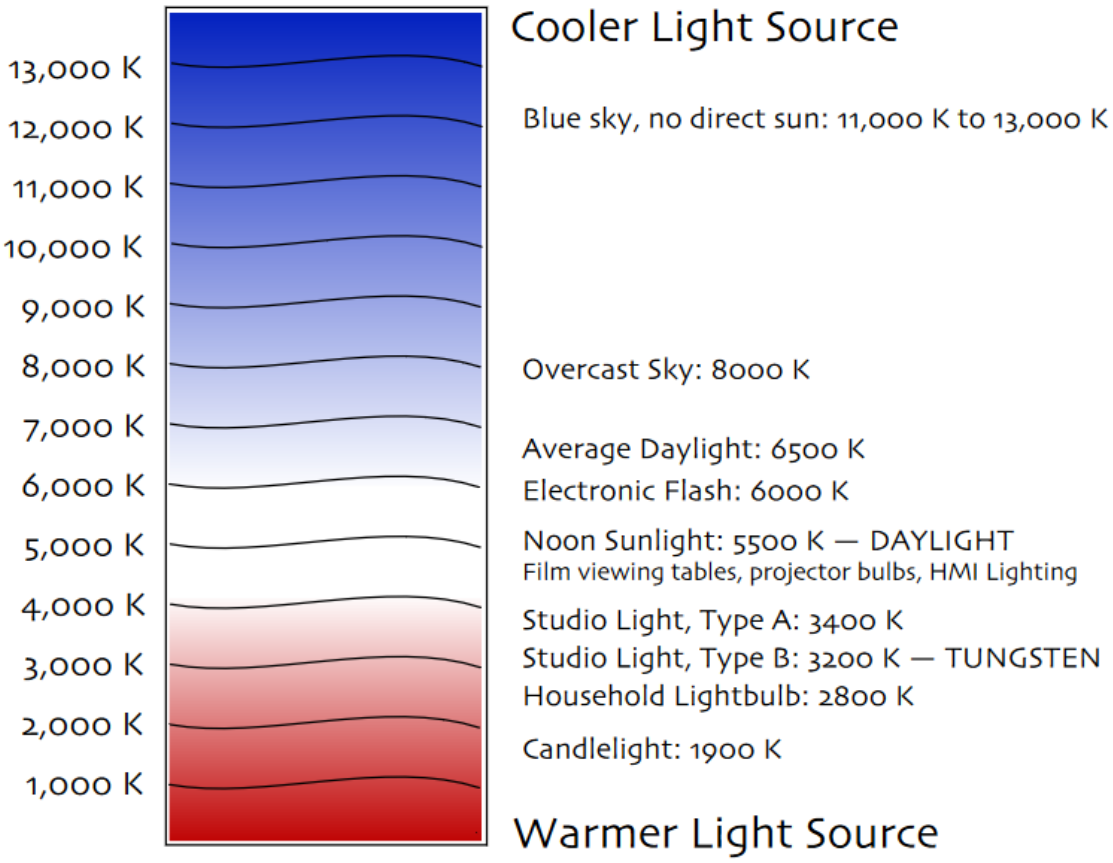

Figure 4. An interpretation of commonly used color temperatures of light based on Kelvin values.

to this thesis was also explained. In order to conclude this chapter, a set of requirements are made from the learned papers that have to be achieved by the end of this thesis. These requirements provide the reader an easier understanding of what is being accomplished and reminders of the relevance of a mentioned implementation.

2.9.1 Requirement 1 (RQ1) - Open-sourced

As was discussed in the Introduction, Open Source and Education, having an application be open-sourced provides tangible benefits as opposed to being closed-source. These benefits include better support for contextualized learning, acquisition of a wider range of skills, and increased student motivation. In addition, from the Related Work application called "Arkio", which was showcased in the chapter Related Work, VR applications in the market, having a closed-sourced application can lead to potential prohibitively expensive deals, which an education system might not be able to afford. Creating an open-source application alone with the capabilities of replacing traditional architectural approaches is out of the scope of this thesis. Consequently, it would be appropriate to allow scholars and researchers to continue the development for architectural benefit.

For the aforementioned reasons, the first requirement is to make an open-sourced application.

2.9.2 Requirement 2 (RQ2) - VR capabilities

It is important to note that the application is expected to run on an Oculus Quest 2[10] without Airlink[11]. This prerequisite is made for the following reason - it is assumed that the user does not have a high-end computer to allow for GPU and/or CPU-heavy tasks.

Requirement 2a (RQ2a) - Intuitive menus

In addition to the assumption of hardware limitations, it is also assumed that the user should easily grasp the controls of the application. This is done to alleviate the frustration that comes from badly implemented user controls, possibly improving the user experience as a result.

2.9.3 Requirement 3 (RQ3) - Functionalities of Application

As was discussed throughout this chapter, VR allows users to experience a 3D environment as if it were real. Intuitively, it can be speculated that VR technology can be used to create applications for architects with the intent to experience concept buildings in an immersive fashion. In addition, said application can also provide a way for architects to modify their design in real-time to see how their modifications would alter the building. Finally,

[10] https://tinyurl.com/y89nznda

[11] https://www.oculus.com/blog/introducing-oculus-air-link-a-wireless-way-to-play-pc-vr-games-on-oculus-quest-2-plus-infinite-office-updates-support-for-120-hz-on-quest-2-and-more/

the design could also be placed in a simulated real-world environment, such as a city, to see how the concept design would fit that location. With everything developed, architects would have a tool that would allow them to visualize their design in an immersive 3D environment, in addition to allowing modifications of the building to see the consequence of their alterations.
With this in mind, functionalities will be defined in sub-chapters that are expected to be seen in the developed application of this thesis.
Requirement 3a (RQ3a) - 3D model construction
The developed application must allow users to create a custom 3D model. It is expected that the user can place 3D objects, such as walls, roofs, and flooring in some concept. In addition, the 3D model's object should be alterable, such as changing its material, shape, and position.
Requirement 3b (RQ3b) - 3D model placement in a simulated real-world environment
The developed application must allow users to place the created custom 3D model in RQ3a in a simulated 3D environment, such as a city. The user should be allowed to place the 3D model anywhere in the simulated world.
Requirement 3c (RQ3c) - 3D model visualization in the simulated real-world environment
The developed application must allow the user to visualize the 3D model in a simulated 3D environment after placement in RQ3b. This should include the user being in the created 3D model, such as being inside of it. In addition, the user should be able to alter world environments, such as the position of the simulated sun.

2.9.4 Requirement 4 (RQ4) - Addition of Radiance

In Related Works, VR Daylight Simulation, RadVR was mentioned for having used radiance as its lighting engine. Having a separate lighting engine would give the benefit of having realistic lighting simulations in the application.

3 Design

The goal of this paper is to create a VR application to allow the user to create a 3D building, place it in a simulated real-world environment, then experience the building from said environment. In order to achieve this goal, a renderer is required to output high-frequency renderings in an efficient manner. In addition to just rendering, tools and libraries are needed to facilitate the creation of the said application. And as mentioned in the Related Work, Requirements, the application is required to be open-sourced (RQ1). For the above-mentioned reasons, the Godot[12] game engine was used.

[12]https://godotengine.org/

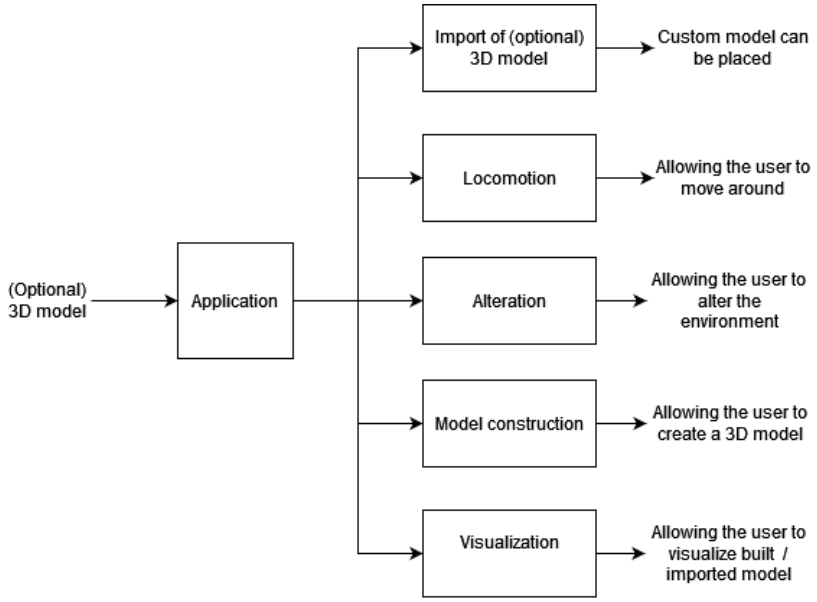

Figure 5. The system overview of the developed application. The system takes an optional 3D model, that can later be placed in the simulated world environment. The application has five main functionalities that it must perform.

The design of the application can be seen in Figure 5. The import of the model assumes that the user has *.obj* files in one of the User folders in the Oculus Quest 2, allowing the user to select them from the applications menu. The Locomotion feature allows the user to look and move around the environment. The Alteration feature allows the user to alter the environment to a certain degree, such as changing the sun's position and placing new objects. The Model construction feature allows the user to create a 3D model of its choice to a limited degree. The visualization portion allows the placement of the created/imported model into the simulated real-world environment, which the user can then immersively experience.
The following chapters go into more detail about the design choices that were initially made and why they have been made. Note that not all portions of the above-mentioned features are covered in the following chapters, refer to the chapter Implementation for more details.

3.1 Use Case

The goal of this chapter is to create a use case on how this application could be used by the user. It will provide a descriptive list of actions and event steps necessary in order to fulfill the requirements mentioned in Related Works, Requirements. This is done in order to give the reader of this thesis a walk through of what the capabilities of this application

Table 1. Use case for the first branching path going to the application box Figure 6. The actor, in this case, is presumed to be an architect without significant background in computer science.

Goal	Visualize the created model in a VR environment
Actor	Architect

Table 2. Use case for the second branching path going to the node outside of the application box in Figure 6. The actor, in this case, is presumed to be an architect with knowledge in the computer science field.

Goal	Visualize the custom-made model in a VR environment
Actor	Experienced architect

are. In addition, a use case would give a descriptive reason for the possibilities of this application. Table 1 showcases a simple case where the assumed actor does not have significant knowledge of computer science and technology. Table 2 assumed a more complicated scenario.
As can be seen in Figure 6, the path starting from the User branches into two. One path goes to the application box, the second - outside of it. For this reason, two use cases will be given to explain the difference.

3.1.1 Use Case 1 - Simple

The architect wishes to create a 3D model in a VR environment to get an immersive feeling for it in a simulated real-world environment. The architect puts on his VR headset of choice and opens the application. The application puts him in an environment, where the architect can put blocks and change their material. After having built his desired 3D model, the architect presses the button to proceed with the application. The architect then gets a view of the simulated real-world environment. The architect puts his recently created 3D model onto the simulated world. Having done so, the architect is put in the 3D model, where it can see the computer-generated lighting. Finally, the architect can put furniture items around the 3D model to see how they would affect the model, and additionally, change the material of the walls the architect recently placed.

3.1.2 Use Case 2 - Advanced

The architect has recently built a 3D model using some software of choice and wishes to see it in a VR environment. The architect opens the documentation of this application and notices the instruction on how to import custom 3D models. Having done so, the architect runs the application again. In order to verify that the application was imported correctly,

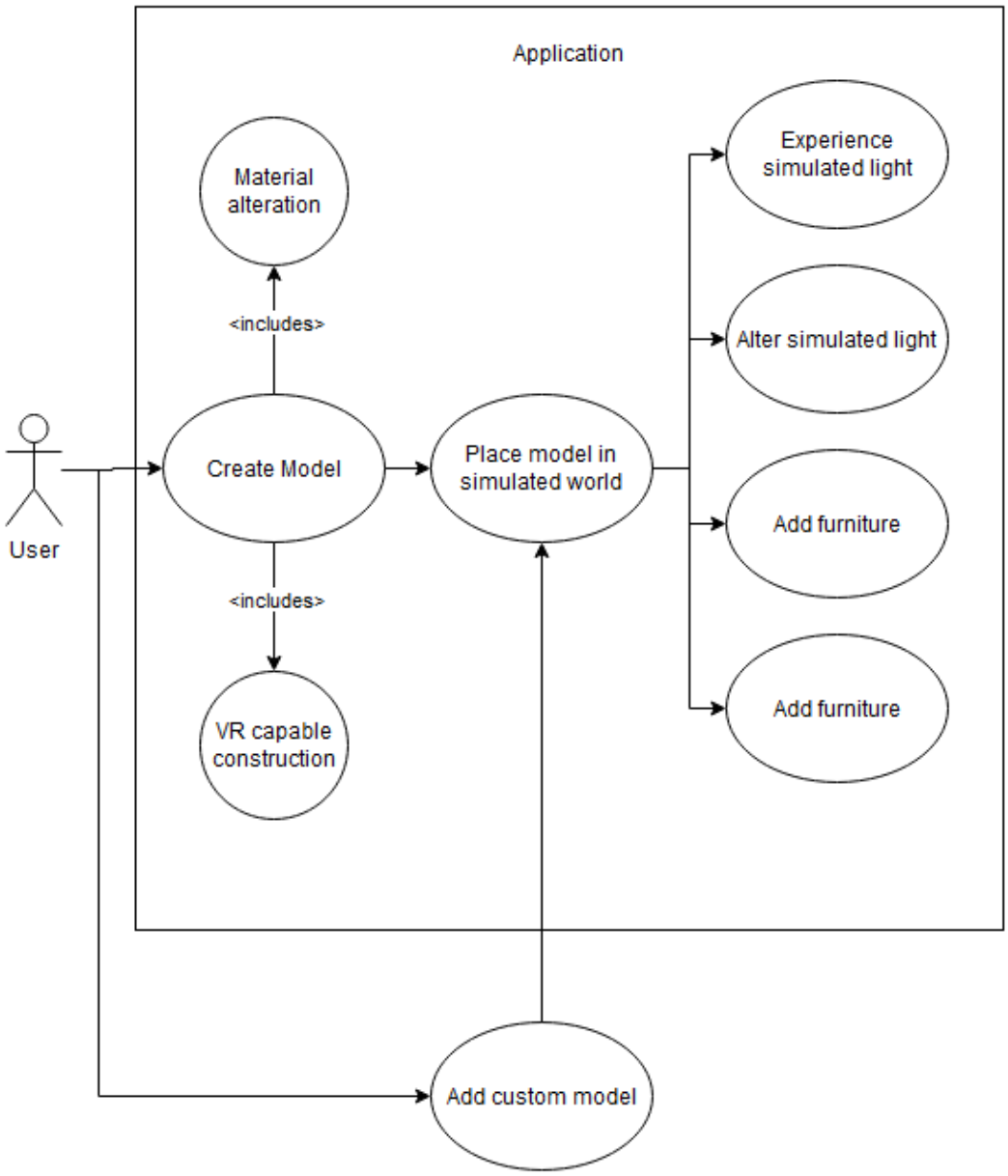

Figure 6. A use case diagram showcasing the possibilities the user has in terms of application functionalities. The nodes outside of the application box represent activities done outside the application.

the architect opens the menu allowing to see all custom imported 3D models. Having verified that the architect's model is there, it chooses it and proceeds to the next step of the application. The architect then gets a view of the simulated real-world environment. The architect puts his recently created 3D model onto the simulated world. Having done so, the architect is put in the 3D model, where it can see the computer-generated lighting. Finally, the architect can put furniture items around the 3D model to see how they would affect the model.

3.2 Model Construction

This part focuses on allowing the user to create a 3D building to be later placed in a simulated environment. Having looked at a lot of previous work related to architecture, the creation of a model in VR is a somewhat new concept. There have been multiple games created in VR that allow the creation of custom-made concepts and objects, such as Minecraft[13] and the already mentioned Arkio. From the available concepts and previous solutions, it was concluded that the model construction feature should be easy to use and allow the objects to be modified.
As the VR controllers allow for limited input, the user should have an interface to see all the available features that can be used. Features should include the ability to place walls in an intuitive fashion and change their appearance. In this portion of the application, the user should also have the opportunity to choose an imported 3D model instead of creating a new one in the same interface.
Godot provides many features for object manipulation and placement, which can be found in the next chapter Implementation, Model Construction.

3.3 Placement

In Related Works, VR Immersion, it was discussed how VR allows users to experience a non-destructive way of observation [8]. From this statement alone, it was decided that the placement of the custom-made object should be as clear and concise as possible, without forcing the user to make any major alterations.
After having built the design, the user should be allowed to place the building anywhere in the simulated real-world environment. The placement should signal to the user when the building can be placed signaled by a colorful indicator. This is done to inform the user when the location is available. In order to make it clear to the user where it is placing the building, the user should have a top-down view of the world environment. Being high and above the world, the user fully visualizes where it is and how it can place the object. In addition, the user should have an indicator as to where it is pointing, to give even more details on the placement of the building.

[13] https://www.minecraft.net/en-us/vr

This feature should be the quickest to make, as the only difficulty of this feature is remembering which coordinates the user has selected in the game world. If the time permits, the placement of the object should be blocked if the user has selected a place where other objects would collide with the created/custom model. Due to mentioned features being mostly quality-of-life, it is not regarded as a necessity.

3.4 Altering simulation

Once the designed concept has been placed in the real-world environment, the application should allow the user to move around the created building for an immersive experience. As mentioned in the related works, allowing the user to move around the built model is seen as useful. For the mentioned reason, locomotion will be added so the user can move and look around.
Also, the user should is allowed to switch the time of day, consequently altering the direction of the sun. The aforementioned feature should properly alter the shadows based on the direction of the sun, as well as provide adequate bounces of the light. Lastly, this part of the application allows for further alteration of the building, allowing the user to place furniture, and lights and modify existing walls with different materials. These changes should also update the shadows and light bounces correctly.
Having some experience with Godot, altering the materials and objects of custom imported scenes could be time-consuming. For this reason, it was decided that the alteration of walls and objects should only be limited to created 3D models that were done in the first stage of the application - Model Construction. if the time allows, the ability to alter even custom imported objects would be a great benefit to the thesis, but not the main priority.

4 Implementation

This application was developed in Godot, a cross-platform, free, and open-source game engine. Godot allows for the development of VR applications and has the tools necessary to create the design parts mentioned in the Design Chapter. The vast amount of tools that Godot has can be found in the documentation[14]. A description of the used tools will be given throughout this chapter.
In addition to using Godot, a Godot Oculus Quest Toolkit[15] was used for basic VR functionality. In addition to just being a toolkit, the Godot Oculus Quest Toolkit's GitHub page provides a wide array of showcased projects and prototypes that users can take a look at and learn from. Relevant to this thesis, one of the prototypes, Voxel Works

[14]https://docs.godotengine.org/en/stable/index.html
[15]https://github.com/NeoSpark314/godot_oculus_quest_toolkit

Quest[16], is a Jog-In-Place locomotion prototype, allowing users to experience an infinite procedurally generated world. Additionally, Voxel Works Quest allows users to place 3D objects in the world. Voxel Works Quest provided ideas for the developed application and possible solutions to the stated Requirements RQ3a and RQ3b, such as how to allow for the placement of 3D objects in a created world environment. No explicit code was taken, unless it is a part of the Godot Oculus Quest Toolkit, meaning that the prototype was primarily used for reference as to how RQ3a and RQ3b could be solved. More can be read on the influence of Voxel Works Quest in Evaluation, Block-Based Building
The following Chapter is split into multiple segments. The first segment, Model Construction, explains how a user would use the developed application to create a construct of its choice in Godot. The next part, Placement, goes into detail on how the constructed building is placed in the real-world environment. The third segment, Altering Simulation, gives an overview of how the real-world environment can be altered once the building has been placed.

4.1 Model Construction

The main goal is to allow the user to create a 3D building in a simple-to-grasp fashion. As described in the Design and Analysis chapter, the creation of the model should include the intuitive placement of walls and alteration of their appearance. Before having implemented said features, the user should first be allowed to look around and move in the VR environment. Due to being constrained to two handheld controllers, it was decided that one controller should be responsible for the user's locomotion, while the other - the rotation of the view.
In order to accomplish the above-mentioned feature, a physical location of the center of the tracking space needs to be designated in the game world in Godot. This can be done using the ARVROrigin[17] node. The ARVROrigin node is the position of the node that is updated when the user moves around the game world.
In addition to the ARVROrigin node, Godot requires three more nodes as the children of the ARVRORigin node - ARVRCamera[18] and two ARVRController[19] nodes. A camera in-game engine usually displays what is visible from its current location. In terms of ARVRControllers, these are spatial nodes that are linked to the tracking of controllers. Godot includes the addition of pass-through for buttons that are integrated into the controllers.
However, to give more functionality to these nodes, a helpful set of tools is used in this thesis - Godot Oculus Quest toolkit[20]. It is an in-development toolkit for basic VR

[16]https://sidequestvr.com/app/431
[17]https://docs.godotengine.org/en/stable/classes/class_arvrorigin.html
[18]https://docs.godotengine.org/en/stable/classes/class_arvrcamera.html
[19]https://docs.godotengine.org/en/stable/classes/class_arvrcontroller.html
[20]https://github.com/NeoSpark314/godot$_o culus_q uest_t oolkit$

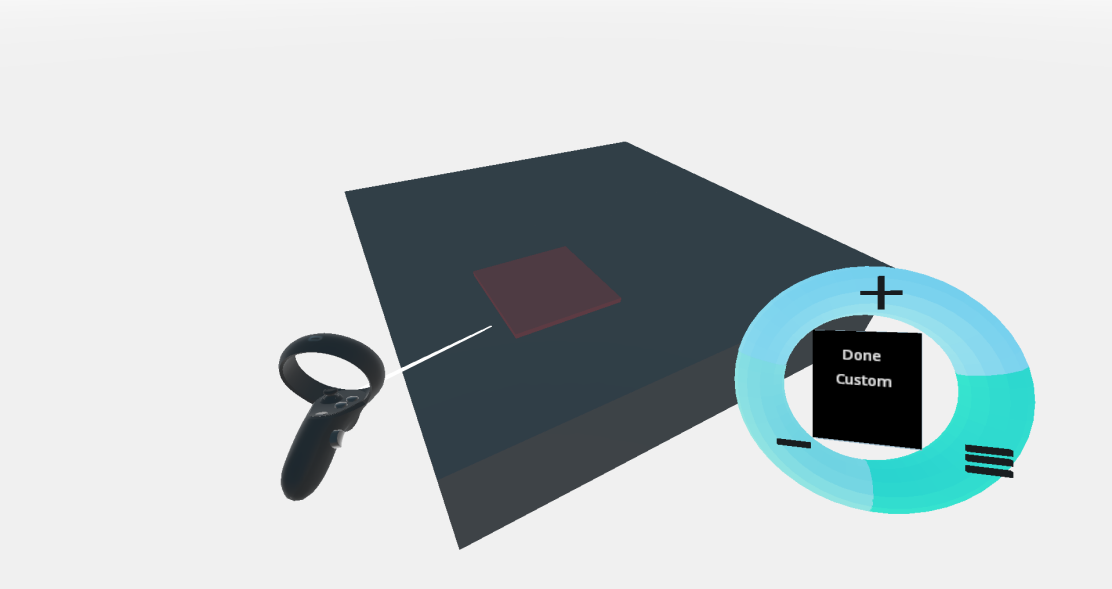

Figure 7. Showcase of the UI canvas on the right controller. The features visible on the canvas include the placement of a new cube, deletion of mesh, and different materials. These features are represented by the plus icon, the minus icon and the hamburger icon.

interactions in Godot. For this step of the application, it is important to note the UI (User Interface) canvas feature of the toolkit. In brief, it allows the user to see a 2D canvas in the game world in order to interact with functions on the canvas. In this particular case, the right controller is being used as a UI item with all the functionalities available to the user (Figure 7). Another important feature of the Godot Oculus Quest toolkit is the already made implementation of locomotion. In brief, the toolkit has a script that allows the user to move around in the VR environment - the Locomotion_Stick. With the Locomotion_Stick in mind, adding the ARVROrigin, ARVRControllers the ARVRCamera is enough to give the user the ability to move around the simulated world.

As per requirement of RQ2a, it is important to make the UI menu as intuitive as possible. A few experiments have been done, including using the Godot's Control Nodes for menuing, however, they were deemed unintuitive for this part of the application. For this reason, the author looked to already made solutions discussed in the Related Work chapter. Specifically, inspiration was taken from an already discussed tool - Arkio, from Related Works, VR applications on the market. Their technique involves replacing the right controller with an intuitive menu which has all the features. With this in mind, an UI interface was created using Blender[21]. Blender is a free and open source 3D computer graphics software tool that allows for creation of 3D objects, which can then be imported

[21] https://www.blender.org/

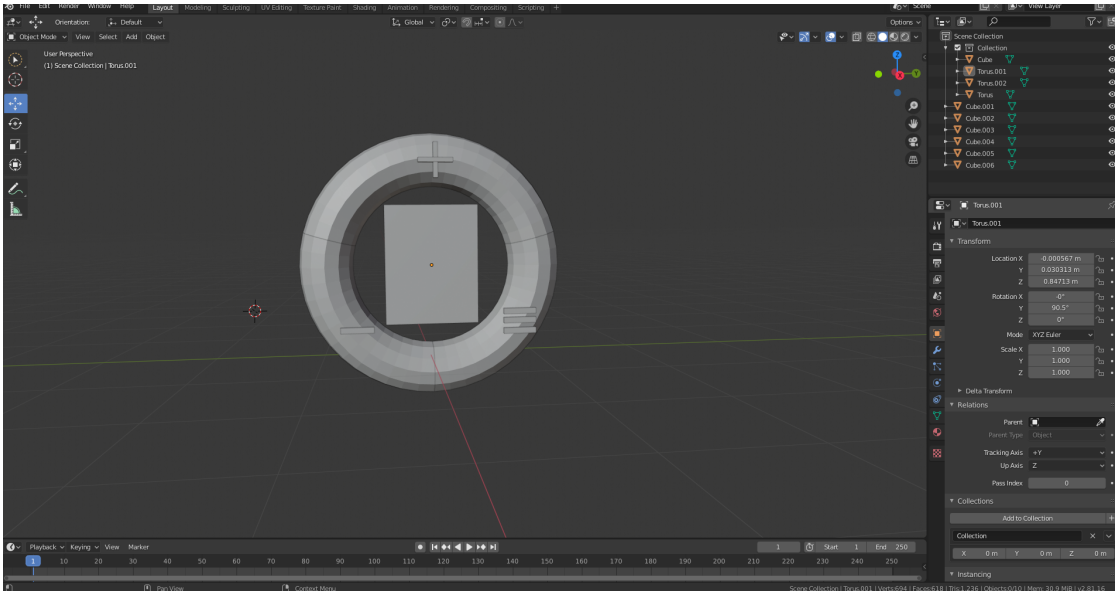

Figure 8. The Blender window showcasing the completed 3D model that will be used in Godot for UI menuing.

onto Godot. As can be seen in Figure 7, the right controller is replaced with a torus-like UI menu with three separate sections. The three separate sections have slightly different colors for better distinguishment. In addition, the three sections have their own unique icons - the plus, the minus and the hamburger icon. The process of creating the UI can be seen in Figure 8. In detail, Blender allows or creation of 3D objects with great manipulation and features. The object first started as a 3D torus object, which was then cut using the Knife[22] tool. The Knife tool allows for separation of the object; it was used to separate the torus into three evenly shaped objects. The three separate sections would be used later to fulfill requirements in RQ3 - allowing the placement of objects and their alteration. Furthermore, there is a rectangle object in the middle of the three separate sections, which is used for Godot's in-built UI interface, visible in Figure 7.

Having implemented locomotion, the UI interface and camera rotation using Godot, the next step is to design the Model Construction itself. Starting with the placement of walls, it was decided that the user can add new "cubes" to the world with the help of the UI canvas described above. Pressing the "Add" button (signified by the block with a plus sign) allows the user can start placing new cubes on the foundation. In order to correctly place them, the Godot Oculus Quest toolkit is used once again for its raycasting[23] feature, which allows the precise direction of where the user is pointing its controller. The pointed

[22]https://docs.blender.org/manual/en/latest/modeling/meshes/tools/knife.html
[23]https://docs.godotengine.org/en/stable/tutorials/physics/ray-casting.html

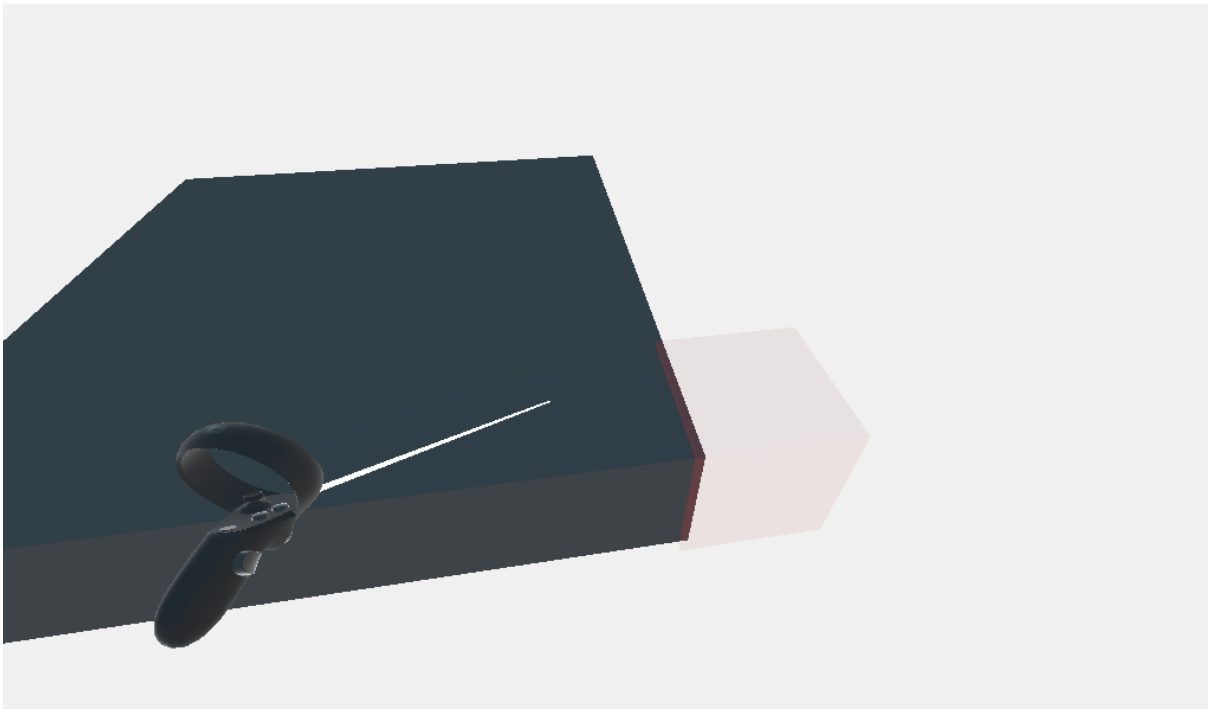

Figure 9. An example of the user pointing the ray at an occupied object and pressing the joystick right to shift the red cube. This allows the user to move the red cube to an unoccupied area.

direction of the ray will be signified by the red transparent box. If there is an object in the red box, the user will not be able to place a new cube on it. The user can manipulate the joystick on the left controller to shift the position of the red transparent box, allowing for the placement of the cube. This can be done by pressing the [INDEX] button on the left controller. Note that for the purposes of readability, ray casting will not be described in this chapter, and more information can be found in the Individual Object Manipulation sub-chapter.
In Figure 9 an example of the necessary steps is showcased in order to start moving the red cube to an unoccupied area. The way the application interprets an unoccupied space is via the use of Arrays - a data structure consisting of a collection of elements. The collection of elements in this particular case is a set of coordinates of all placed cubes. The code iterates over the specific "line" of cubes based on the direction of the joystick from the starting cube. It iterates until it finds coordinates with an unoccupied space. As for the creation of a new cube, this application uses Godot's Instancing[24] feature. In brief, Instancing allows for reproduction of "blueprints". In this case, the "blueprint" is a 3D object in the form of a cube, waiting to be instanced onto the scene the user is

[24]https://docs.godotengine.org/en/stable/getting$_started/step_by_step/instancing.html$

currently in.
In addition to placing the cubes, this part of the application allows for the deletion of the cubes. This feature allows the user to correct mistakes or add alterations if necessary. In Figure 7, the "Delete" feature is represented by the minus icon. For the deletion of the cube itself, it is done via Godot's "queue_free()" command [25]. In short, it deletes a node and all of its children from the scene. Due to the previously mentioned array system for saving coordinates - once a cube is deleted, its coordinates are also deleted from the array.
Deleting the cube follows the same principle as moving it. If there is an object in the red cube, pressing the [GRIP] button on the controller will remove the cube from the 3D world.
The last section of the button is represented via the hamburger icon. This section allows the user to change materials. Once pressed, as can be seen in Figure 10, a different UI shows up in the middle. The UI has three buttons - Wood, Brick, and Glass. They only appear after having pressed the hamburger icon section of the UI. Pressing these buttons allows the user to change the material of the cube to any of these three materials. Following the pattern, changing the material of the object is done by pointing the ray at any of the cubes. Figure 10 demonstrates the changes to the material of the cube. Every placed cube is a MeshInstance[26] - Godot's property for creating a resource that can be added to a scene. For this reason, each cube has a Material property used for coloring and shading geometry. With the same said property, Godot allows for alteration of the cube's material at any time with the right methods. In this particular case for material alterations, the method "Material Override"[27] was used to change the materials to whatever the user wishes. With all of these features explained, it can be concluded that RQ2, RQ2a and RQ3a were fulfilled.

Once the user is satisfied with the building, pressing the Done button visible in Figure 7 will store the building as a resource and allow its placement in the next stage of the application, as per requirement RQ3b. Note that the Done button will only be visible if either the Add (Placing of cubes) or Delete (Removal of cubes) features are enabled. Storing the created building is done via the use of two Godot's tools - ResourceSaver[28] and PackedScene[29]. In short, all of the placed objects are saved in a PackedScene, which can then be saved as a file using ResourceSaver. With both of those actions done, using Oculus Quest toolkit allows for convenient switching of scenes via the use of the "switch$scene()$"$method$.
It is important to note that the user may skip the step of having to build a concept building.

[25] https://docs.godotengine.org/en/stable/classes/class$_node.html class-node-method-queue-free$
[26] https://docs.godotengine.org/en/stable/classes/class$_meshinstance.html$
[27] https://docs.godotengine.org/en/stable/tutorials/3d/spatial$_material.html$
[28] https://docs.godotengine.org/en/stable/classes/class$_resourcesaver.html$
[29] https://docs.godotengine.org/en/stable/classes/class$_packedscene.html$

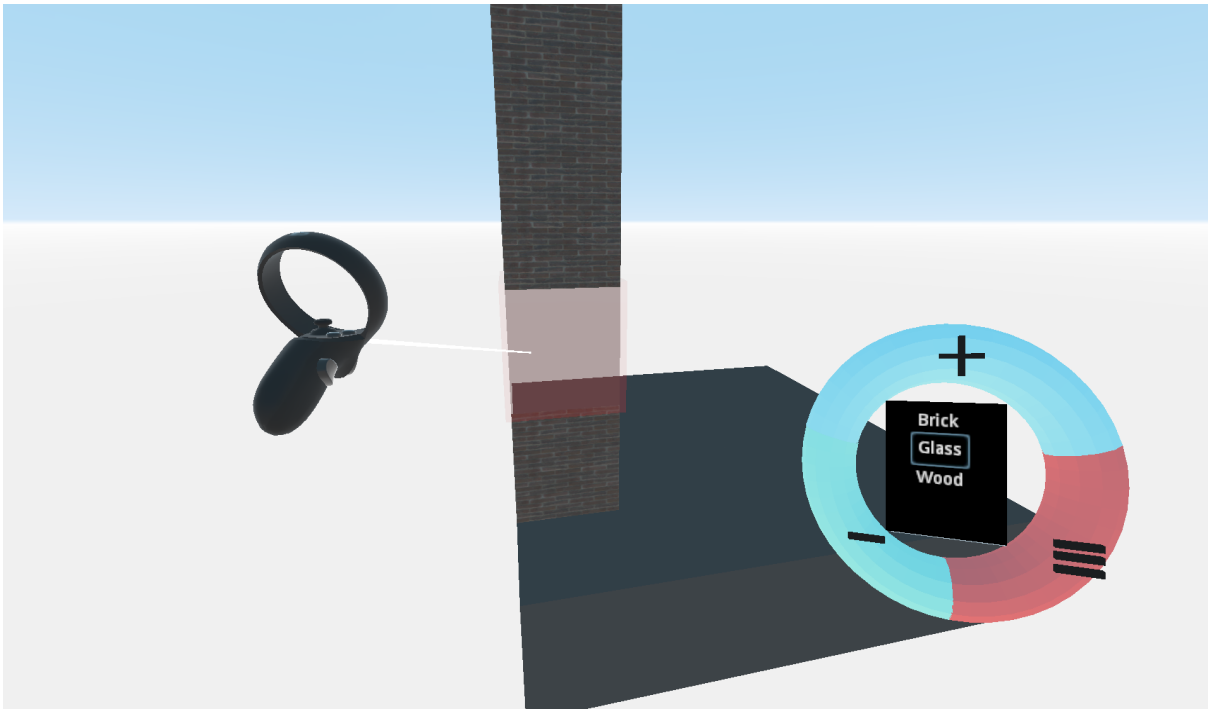

Figure 10. Changing the material of the cube from brick to wood via the use of the UI canvas. Notice that the menu on the right has the hamburger icon section in red, meaning the user is currently using that feature.

In addition, the user may add its own 3D building of choice into the application. This process, however, cannot be done in the developed *.apk* file and requires considerable steps from the user. Consequently, the steps to import custom buildings will not be provided in this paper but will be added to the authors Github page of this thesis [30]. The reason for the existence of this feature is due to the mentioned use case in Design, Use Case, as well as requirement RQ3b.

4.2 Placement

Having chosen or built the desired building, the next step of the application is to allow the user to place it anywhere in the designated real-world environment, as per requirement RQ3b. This is also done with the help of the Oculus Quest toolkit, as it allows for precise placement of objects based on where the user points one of the controllers.
Once in the scene, the user is placed above the real-world environment, like a city, to have the full view. This gives the user the most options and precision as to where to place the building. As was explained in Model Construction, the user is allowed to move and rotate its orientation with the help of the two controllers. The left controller is responsible for locomotion, the right - rotation of the camera.
As can be seen in Figure 10, the user has a top-down view of the 3D world environment. In addition, if the user is pointing to a valid location in the 3D world, a green rectangle will appear, signaling a valid location. Pressing the [INDEX] button on the left controller will place the created/custom object and move the user to the placed location. With all of the process explained, it can be concluded that RQ3c was implemented.

4.3 Altering Simulation

The final stage of the application involves the user experiencing the insides of the concept building while being in the simulated real-world environment. After having placed the object, the user will appear inside the building in the desired location. From there, the user can move around, experience the simulated lighting, and place predefined objects.
All of the aforementioned functionalities can be done via a menu that can be opened by pressing the [Y] button on the left VR controller, which reveals a menu with buttons. This is different from the previously showcased menu UI that can be seen in Figure 7 for the following reason:

- Two menus do not share the same functionality and purpose. In detail, the new menu requires the user to be able to alter the direction of the sun, its color and place new objects. While placement of the objects can be seen as similar functions, the other aforementioned tools would not fit the previous UI. This is further reinforced

[30] https://github.com/aleks96n/MasterThesisAleksander

Figure 11. Top-down view of the custom 3D world, with the user being in charge of placing the developed/custom object. Notice the green rectangle on the center of the screen, signaling to the user the location the developed/custom building will go to.

by RQ2a, which requires the menu to be intuitive. Because of this requirement, it was decided that the menu should look the way it does in Figure 12.

The Sun position slide visible in Figure 12 can be manipulated using the VR controllers to change the direction of the sun based on the y-axis. This will update the shadows according to the direction of the sun. This is done using the WorldEnvironment node in Godot[31], as it allows for the creation of a custom world environment. The world environment includes the adjustment of the position of the sun, as well as altering the color of both the sun and sky. The WorldEnvironment node is explained in further detail in Chapter Evaluation, Sunlight color.
The WorldEnvironment node, however, does not change the direction of the shadows - that is the work of the DirectionLight[32] node.
The [PLACE OBJECT] button in the menu allows the placement of predefined objects onto the concept building. This is done to furnish the building with different objects, allowing the user to visualize how daylight would affect them and how the light would bounce. In addition to furniture items, the user can place light in the building to further

[31]https://docs.godotengine.org/en/stable/classes/class_worldenvironment.html
[32]https://www.google.com/search?client=firefox-b-d&q=godot+directionlight

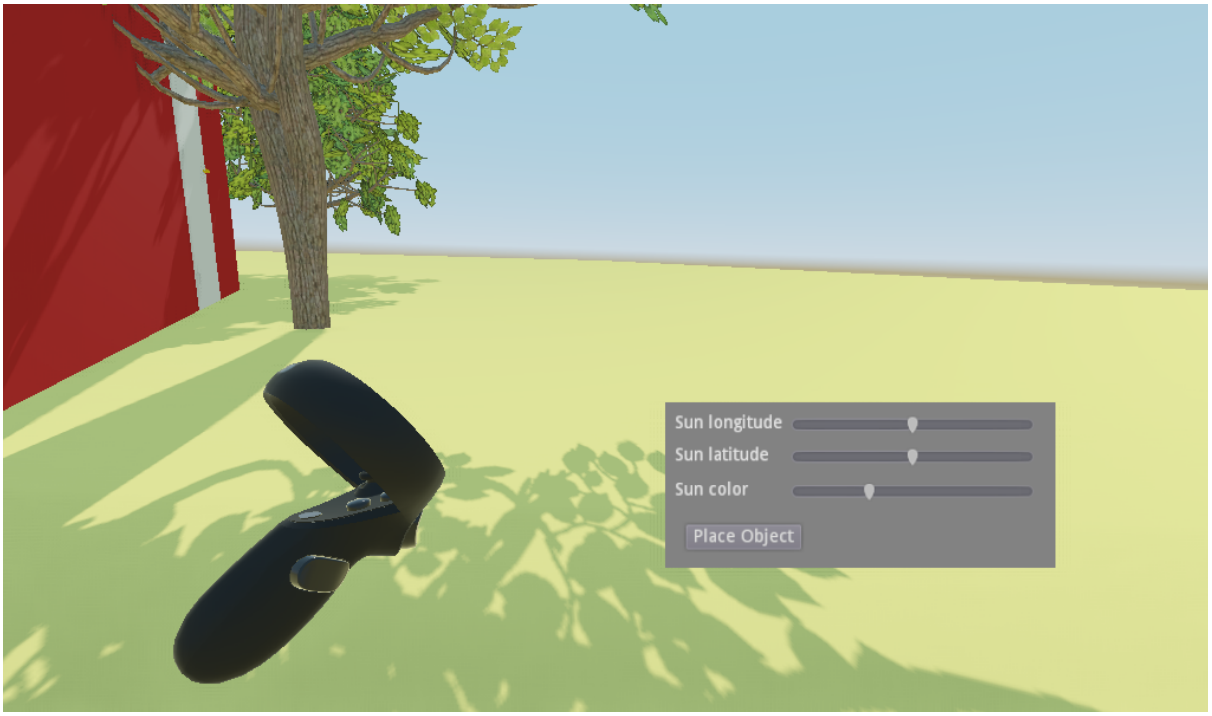

Figure 12. The user is teleported to the custom building in the desired location. The UI canvas is being opened and showcasing all the functionalities.

alter the lighting of the building.
Lastly, the ability to change the color of the light rays can also be done in the same menu. As of requirement RQ3c and Related Works, Sky Illumination, alteration of the light rays was included. This is done by manipulating the slider found in Figure 9 titled Sun Illumination. More on Sunlight Color can be found in Analysis, Sunlight color.

4.4 Sun position

One of the features that were attempted was to simulate how the sun would emit different colors based on its position in VR. For example, in the real world, humans perceive sunlight in the morning as "orange"[33], even though scientifically it is known that the color of the sun is "white"[34]. There are many reasons why humans see different colors of the sun at changing positions, such as the swirling air that surrounds the planet. Due to the many factors that come into play when dealing with sunlight colors, perfectly recreating the way the sun behaves in VR was deemed out of scope for this baseline application.

The concept was developed in Godot VR with the help of the following features - WorldEnvironment and DirectionalLight. The WorldEnvironment node in Godot allows the developer to alter many features, but only the following settings were deemed relevant for this application:

- Manipulating the sun's position, color, and intensity
- Manipulating the sky's color and its intensity
- Manipulating the ambient light of the world, such as color and intensity

As previously mentioned, the DirectionLight node was also used, primarily for two reasons - altering the direction of the light and changing the color of the light rays.
With the features explained, the following Figure 13 shows how the thought of the concept works. A user can be seen being in the middle of the simulated world environment, having the ability to change the sun's rotation. This is done in Godot via the use of both WorldEnvironment and DirectionLight. The WorldEnvironment node allows the developer to change the position of the sun in the game environment. However, the simulated sun does not provide correct bounces of light rays. That is done via the use of the DirectionLight node, which is used conjunctively with the WorldEnvironment node.

[33]https://theconversation.com/curious-kids-why-is-the-sun-orange-when-white-stars-are-the-hottest-120216

[34]https://www.wtamu.edu/~cbaird/sq/2013/07/03/what-is-the-color-of-the-sun/

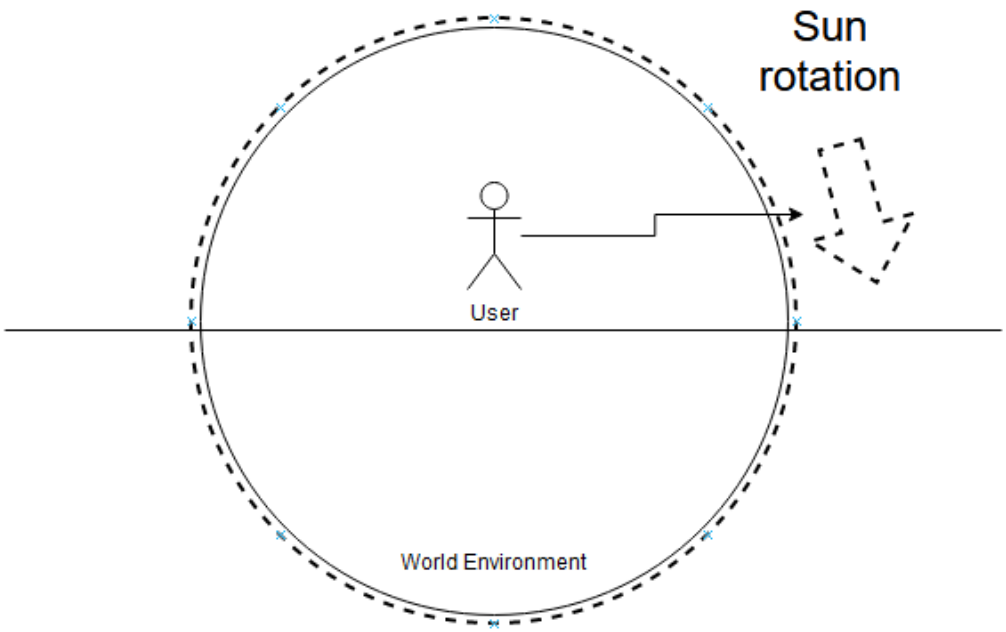

Figure 13. Sketch of the sun rotation concept used in the application. The dotted line signifies how the simulated sun could rotate in the world environment. The user has the capability of altering the rotation of the sun.

4.5 Illumination tactics in Godot

In the Introduction, it was discussed how simulated lighting in-game engines could have a positive effect on the user. The question arises as to what kind of illumination tactics can be used, what were used, and why some approaches were disregarded. In this following sub-chapter, some lighting techniques will be explained, and then their relevance to this thesis will be discussed.

4.5.1 Global Illumination

Global Illumination describes several algorithms used to calculate non-direct lights in game engines[35]. In Godot, this is done via the use of a GIProbe node, which calculates emissive lights and secondary reflections. This technique gives accurate lighting, at the cost of significant performance. As a reminder, while the objective of this particular thesis is to create a baseline application for future researchers to work on, the hopeful future is to develop a tool for architects that allows them to use VR technology, replacing the traditional modern tools they use today. Having accurate realistic lighting can be attributed to that cause. With that said, the question arises as to why this feature is not developed.
As mentioned in the chapter Design and Analysis, it is assumed that the user does not have his computer wired to the VR headset. This means that the only hardware the headset can use is whatever is integrated into the headset itself.
The addition of the GIProbe has caused severe performance drops for the Oculus Quest 2, making the application unusable. Advice was asked at the official Godot Forum regarding this issue. To give a brief answer from one of the official Godot Developers - "GIProve is indeed too demanding to be used for standalone VR"[36]. As a result of this answer, Global Illumination techniques were not implemented in the developed application.

4.5.2 Baked Lightmaps

As a consequence of the question on the use of Global Illumination techniques in Godot, a suggestion was given by the same developer - Baked Lightmaps[37]. In brief, baked lightmaps are an alternative technique for adding indirect lighting to a scene, which is usually better for low-end computers and mobile devices. The problem with Baked Lightmaps is the following - they are completely static, meaning they cannot be altered once created. As has been showcased in the chapter Implementation, the user is allowed to constantly alter the object of its choice, either via moving it, reshaping it, or even

[35] https://gamefromscratch.com/global-illumination-in-godot/

[36] https://godotforums.org/discussion/29774/giprobe-destroys-performance-in-vr-oculus-quest-2#latest

[37] https://docs.godotengine.org/en/stable/tutorials/3d/baked_lightmaps.html

completely removing it. As a consequence of the possible alterations, Baked Lightmaps will need to be rebaked every time a small adjustment is made.
It can be argued that allowing the user to bake the lightmap whenever it sees fit is a possible solution to the problem. That statement, however, could raise the following counterpoint - would the addition of baked lightmaps help replace traditional architectural applications? Traditional architectural applications use lighting techniques that almost perfectly recreate real-life daylighting, while baked lightmaps are a low-end technique that is primarily used because it has a low cost on performance. The low cost of performance is a consequence of not having the most technologically advanced algorithms that allow for it to recreate real-life lighting. Another argument against having baked lightmaps is the way the technique would be implemented. Allowing the user to bake the world at any given time requires input from the user, such as a press of a button. While seemingly simple, this type of interaction has to be made every time an object is modified in any way. This can be argued to be the practice of bad user interface, but without a definite survey and inquiry on user-friendliness in the developed application, is hard to prove.

To conclude these chapters, requirements RQ1, RQ2 and RQ2a, RQ3a, RQ3b, RQ3c were fulfilled in the developed application. However, RQ4 was deemed out of scope for this application. Radiance is a complicated tool that requires a lot of work to fully comprehend. In addition, using it as a separate lighting engine with Godot was deemed out of scope, as other requirements took priority.

5 Analysis

The objective of this thesis is to develop an application to allow users to do the following actions - create a 3D model in a game engine in VR, place said 3D model in a real-world environment, and finally, alter the real-world environment. These steps would allow users to see how their 3D constructs would look in a real setting before their initial construction in the real world. With that in mind, the following Chapter goes into detail on the design decisions that were made throughout the development of this application.
One aspect of this thesis is to test and figure out the Godot engine and its capabilities. Throughout this thesis, these capabilities were tested as to what they can offer for this application. Any design choice that is done will be analyzed and explained in this chapter. The purpose of this analysis is to give the reader more understanding of the Godot engine and explain the design principles that were chosen throughout this thesis.

5.1 Lighting Technique Comparison

The goal of this chapter is to give a basic introduction to how to use the different lighting techniques in Godot. This chapter will only cover techniques done throughout the thesis.

Explaining every detail of Godot's lighting engine is outside the scope of this thesis, and permits an easier grasp of the subject. Lastly, this chapter will not explain the basics of Godot itself, it is expected that the user trying to recreate these techniques knows basic concepts.
If the reader wishes to recreate these techniques, it is first advised to add two nodes to the current scene - WorldEnvironement[38] and DirectionalLight[39]. The WorldEnviromenet node allows for the creation of the background ground and sky colors, as well as alteration of the sun. In addition, SSAO (Screen Space Ambient Occlusion) can be enabled in the WorldEnviromement node, which approximates the ambient occlusion effect in real-time. This allows for a better visual experience when dealing with lighting techniques. For the aforementioned DirectionalLight, it is advised to enable shadows, for the same mentioned reason.

5.1.1 Global Illumination

Having placed objects of your choice around the scene, creating the global illumination technique comes with the following steps:

1. Place the GIProbe node onto the scene. Once placed, locate it on the scene and note the green border that appears. If no green border appears, refer to point *a*.

    a. Having selected the GIProbe in the scene view, select the View menu. In it, scroll to Gizmos and select GIProbe. Figure 14 shows the exact position of the gizmo.

2. Surround the green border with objects you want the Global Illumination technique to work on. Any object outside of the border will not have the technique implemented.

3. Press the Bake GI Probe to bake the Global Illumination technique.

5.1.2 Lightmap

Unlike the Global Illumination technique, Lightmap requires objects to have a UV2 layer and a texture size[40]. Consequently, default objects in Godot, such as common MeshInstances[41], will only work if they are of the ArrayMesh type. This consequence

[38] https://docs.godotengine.org/en/stable/classes/class_worldenvironment.html
[39] https://docs.godotengine.org/en/stable/classes/class_directionallight.html
[40] https://docs.godotengine.org/en/stable/tutorials/3d/baked_lightmaps.html
[41] https://docs.godotengine.org/en/stable/classes/class_meshinstance.html

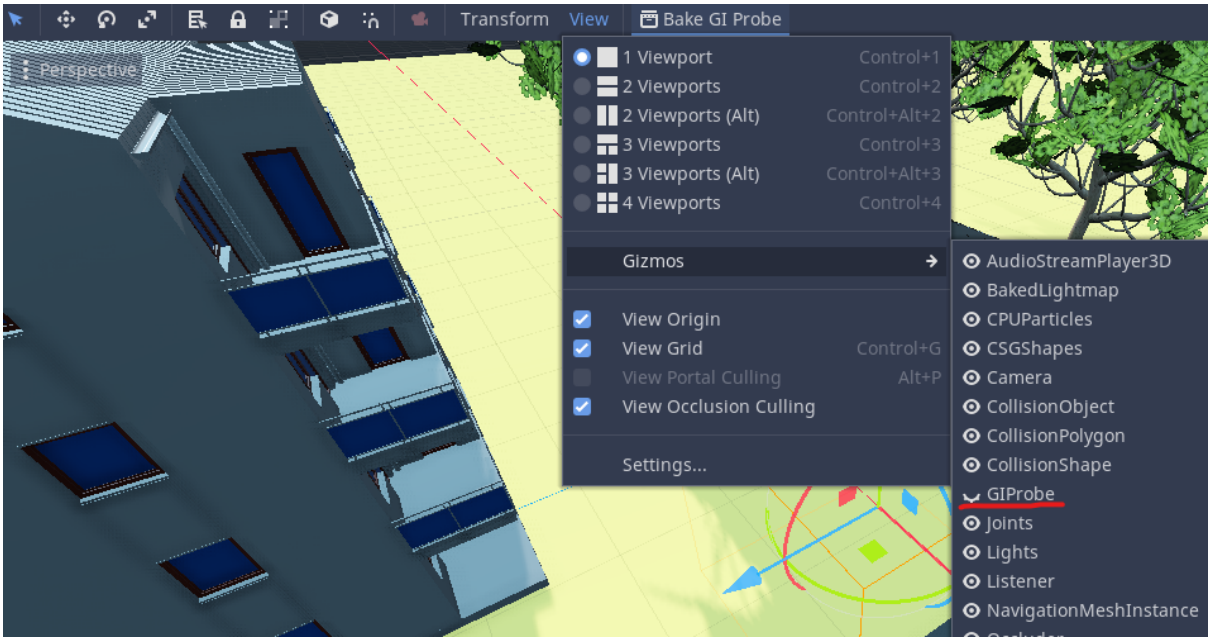

Figure 14. Showcasing how to enable the green border in Godot for the GIProbe node, also noting the Bake GI Probe button in the top middle of the figure.

can be avoided by using objects only with an ArrayMesh type, such as custom imports from Blender[42] and other 3D model software tools.
Having made note of the objects in the scene, said objects need to have a UV2 layer and texture size as mentioned previously. The easiest way to do that is by clicking the Mesh menu for every individual Mesh and Unwrapping the UV2 for Lightmaps. Having done so for every object that is of ArrayMesh type, the Lightmap technique works in the following way:

1. Place the BakedLightmap node onto the scene. Note the blue border around it.

2. Cover the border with every object your want to be Lightmapped.

3. Press the Bake Lightmaps button. Note that this process takes time (based on Quality and Bounces that can be selected from the Tweaks menu in the Inspector).

[42]https://www.blender.org/

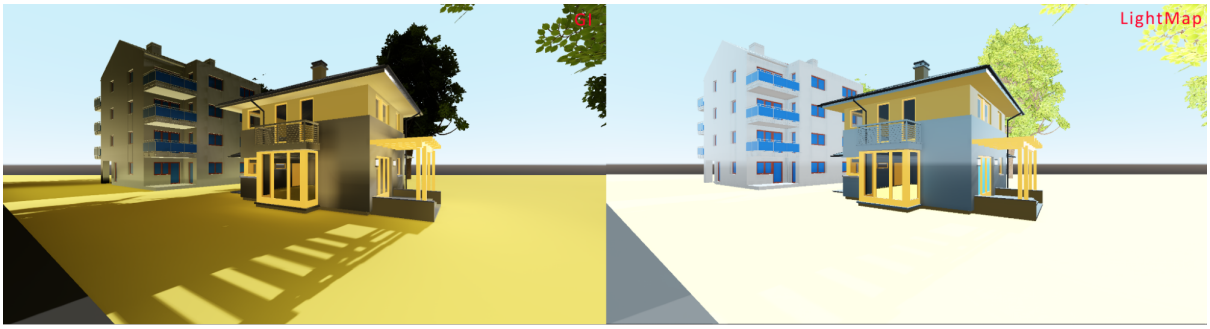

Figure 15. Comparison of houses in the VR environment given different lighting techniques. The image on the left is done using the Global Illumination technique. The image on the right is done using LightMaps.

5.1.3 Comparison

Below are figures made for comparison of the two techniques. Both Figure 15 and Figure 16 were made with the same camera perspective for a fair comparison. Based on these two figures, a few conclusions can be made:

1. The shadows on both figures are more distinct when using the GI technique.

2. Using the GI technique results in a darker image when compared with the LightMap technique.

While more distinctions can be made, the critical part is the performance of these techniques - something the figures alone cannot showcase. Both of these techniques had the same problem - when running these lighting techniques in VR, the performance would significantly decrease, making the application unusable. In addition, baking the Lightmap techniques takes a significant amount of time, even when lowering the settings for it. During the baking time, the application in VR starts to decrease in performance, reducing the quality of user experience.

5.2 Sunlight color

The implemented Sun position scheme was illustrated and explained in Implementation, Sun position. In this subchapter, the design decision behind said implementation will be discussed and analyzed.
As was mentioned in Related Works, Sky Illumination, two sky illumination concepts were brought one. One was the use of a color rendering graph that was provided after extensive research on outdoor illumination [15], and the other - was the use of

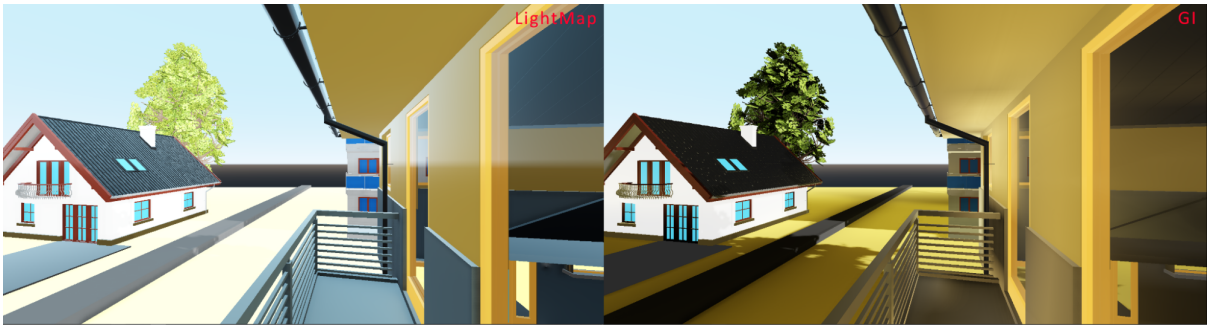

Figure 16. Comparison of houses in the VR environment given different lighting techniques. The image on the left is done using the LightMap technique. The image on the right is done using Global Illumination.

Kelvin temperatures [17]. However, following the design decision in Implementation, Sub position, it can be witnessed that there was no mention of any color changes to illumination. The reason for this is the following - both implementations of the concepts discussed in Related Works, Sky Illumination, were deemed unfit for the application.

5.2.1 The Kelvin Technique

The Kelvin method relies on a Kelvin scale, which was shown in Figure 4. The Kelvin method then can be seen as the following - based on the position of the sun, assign the corresponding Kelvin value to a color temperature and alter the light of the environment. For example, if the sun is in a position that would be considered Noon, the Kelvin scale in Figure 4 would put the color temperature at 5500 Kelvin, making the color itself - white. Consequently, the sun's rays would be changed to white.
Figure 18 shows the main idea behind the implementation - the sun would rotate on a 360-degree circle around the world environment. Then, based on the rotation of the sun, the appropriate Kelvin temperature color would be given to the Directional Light node and World Environment's Sky node.

This concept worked in practice, however, it was deemed unfit for the following reasons:

1. The colors made the game world too colorful and unlike the real world.
2. The colors would not show an appropriate simulation of how the constructed 3D model would look in the real-world environment. In detail, there was no guarantee that the colors chosen would fit the designated simulated real-world environment.
3. Kelvin's temperature color did not reflect what the environmental lighting would look like in a real-world environment.

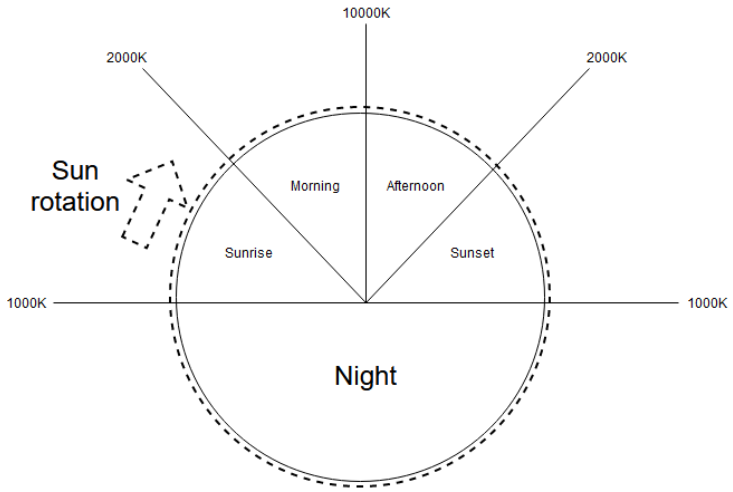

Figure 17. Kelvin temperature values based on the position of the sun. The dotted line shows how the sun would rotate, while the solid lines signify what the Kelvin temperature would be based on the sun's rotation.

5.2.2 Sun Elevation Method

Similar to the Kelvin Method, a useful graph is provided in Figure 3 as to how the colour of the light should look based on the elevation of the sun. Noticeably, the elevation of the sun in Figure 3 is given at an angle, which can intuitively be used in Godot with the WorldEnvironment node, which also has the elevation of the sun at a positional angle. Figure 17 shows how the rotation of the sun works in Godot with the use of the WorldEnvironment node. Intuitively, using Figure 3 as a reference for the color of the sun, a technique can be applied that, given a certain position, changes the color of the light rays to their appropriate color. The multiple reasons why this idea is not in the final build are the following:

1. It was assumed that the user might not want to change the color of the rays at all. This is further reinforced by the aforementioned discussion in Illumination tactis in Godot - the inability to simulate real-life light bounces.

2. The color did not reflect what the environmental lighting would look like in a real-world environment when presented in the game world.

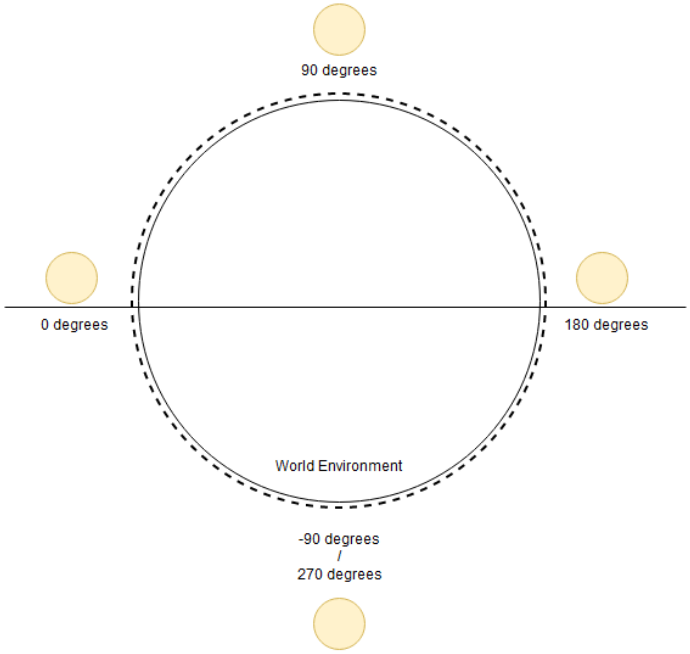

Figure 18. Sun Behaviour in Godot based on the WorldEnvironment node.

5.3 Model Building

In this chapter, some of the experiments for tackling the model building aspect of the application will be analyzed. In the first chapter, a block-based approach will be showcased and explained. In the second chapter, a more complicated solution will be examined and thoroughly analyzed.

5.3.1 Block-Based Building

As explained at the start of this Chapter and Background, Requirements, one of the core ideas was to create an application that would allow the user to create a 3D model of their choice (RQ3a). During the development of this particular feature, it was speculated what would be the best approach to implement RQ3a. Some ground rules were formed before the development of this feature:

1. The creation of the 3D model should be simple and intuitive to the user.
2. The implementation allows the user to create the 3D model of his choice.
3. The 3D model can later be placed in the real-world environment.

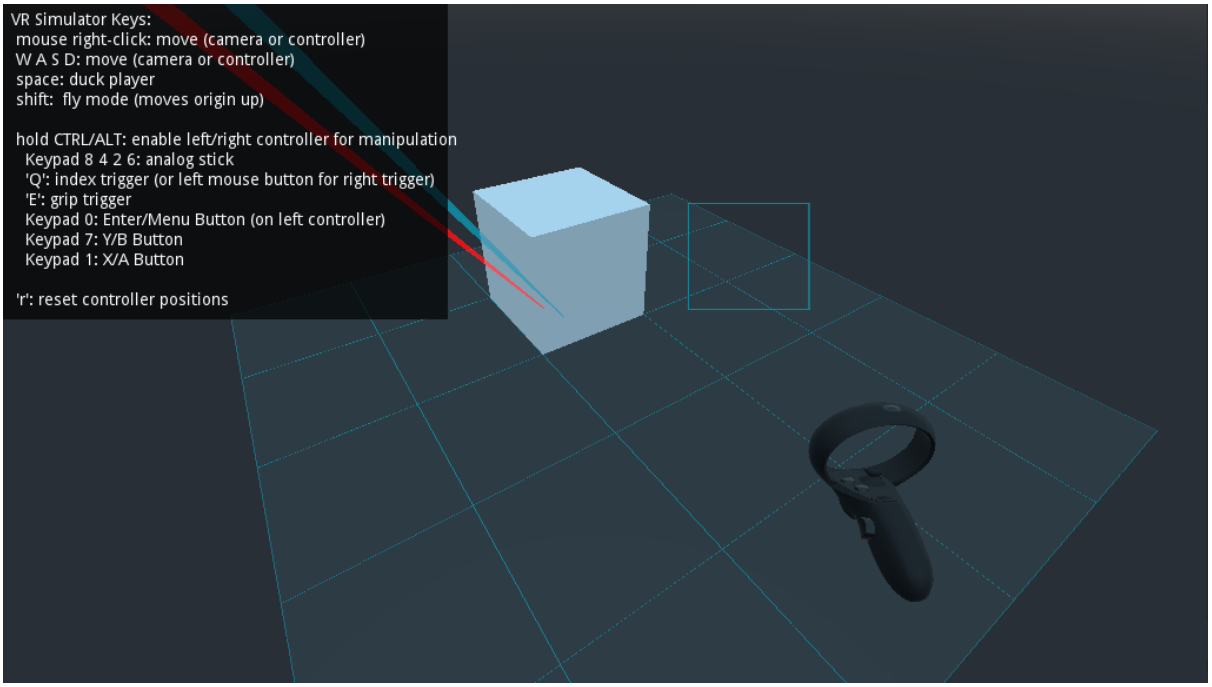

Figure 19. Simulated VR environment in which the user is placed in a scene with a black foundation. The black foundation is the starting grid in which the user can then place blocks of his choosing on the said grid.

With the ground rules formed, the following concept has been thought of - block-based building. This concept was inspired by Voxel Works Quest[43], a prototype for the Oculus Quest in which users can jog in place and place "cubes" in the game world. The initial thought of concept was the following:

1. The user is placed in an environment with an invisible grid. The grid would be used to properly position "blocks".

2. The user is only allowed to put "blocks" of the same size as every other "block". The blocks themselves are predesignated Meshes, forming a Cube.

3. The user is allowed to delete already placed blocks or alter their material.

The thought of the concept was developed in the exact same fashion. Figure 19 shows the concept in practice.

[43]https://sidequestvr.com/app/431

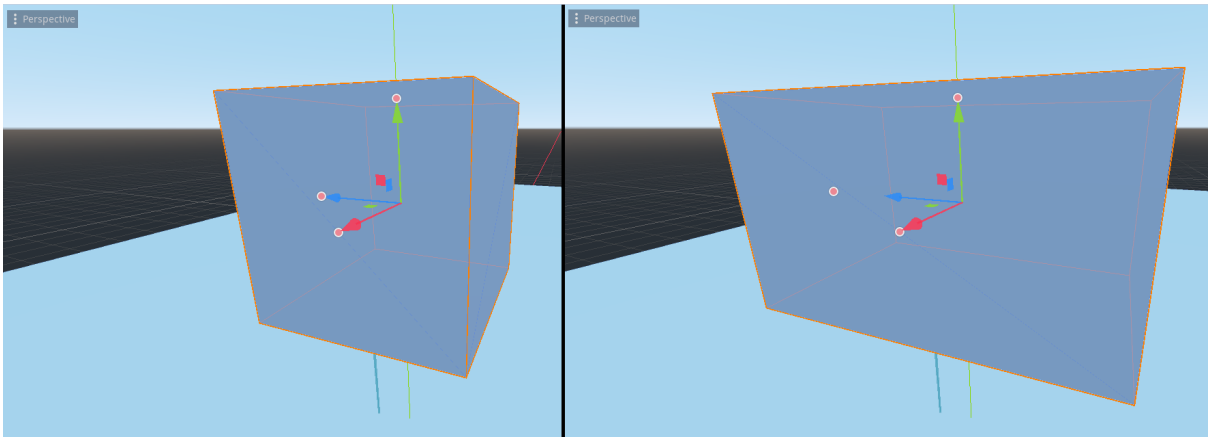

Figure 20. The box on the left is the default CSGBox that is provided by Godot. The box on the right is the box being scaled on one side. Notice that the origin point (represented by the three-axis arrows) remains in the center of the box.

As Figure 19 shows, the user is placed in a scene with an invisible grid with black blocks. The grid is shown in the figure because of the enabled debugging option, in which the grid is shown if that option is enabled.

While the concept worked in practice, another idea came to mind which was tried and tested.

5.3.2 Individual Object Manipulation

As a reminder, the goal was to create an application to allow users to build their 3D models and place them in the environment. To facilitate this idea, appropriate building blocks were required. The initial idea was to create a singular 3D cube, which allows for its manipulation. The CSGMesh[44] node comes in handy to achieve said goal. It allows the use of any mesh resource, allowing the creation of a custom-shaped mesh. In this particular application, the custom-shaped mesh resembles an even cube. While the Godot by default allows choosing a Cube as a starting mesh, the problem comes with its manipulation.

Figure 20 demonstrates the problem with default object manipulation in Godot. When an object is scaled, the origin point stays the same, meaning that if one wishes to extrude one of the sides of the box, the opposite side will also be affected.

For this reason, a custom-shaped mesh is required. This creation of such a mesh requires specific coordinates for each point of the mesh.

Figure 20 is provided to help understand the meaning behind each point requiring a specific coordinate. For example, based on Figure 20, it can be seen that point D has

[44]https://docs.godotengine.org/en/stable/classes/class_csgmesh.html

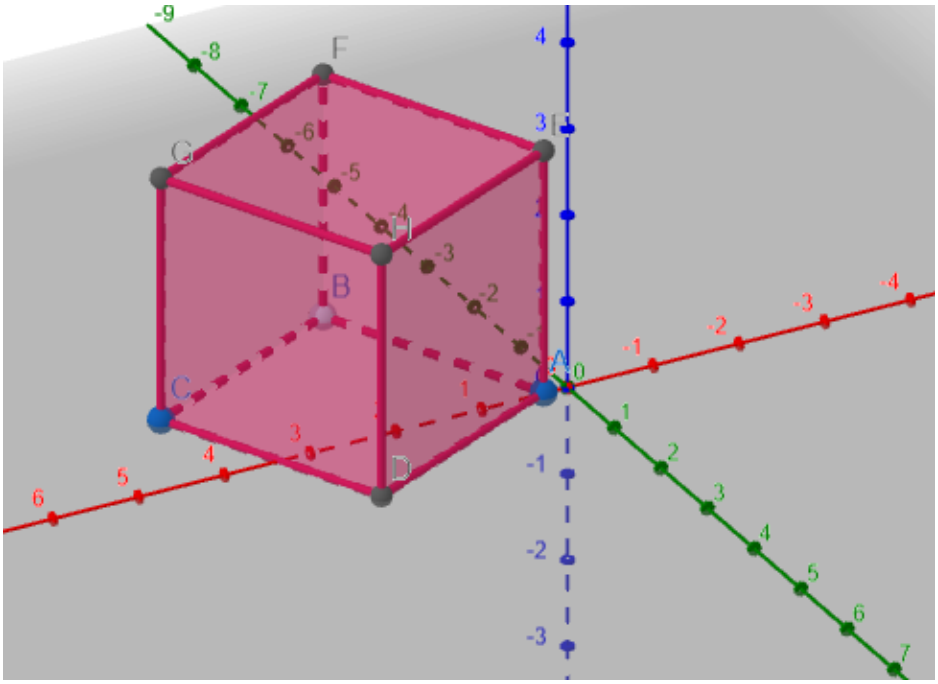

Figure 21. Helper graph to help explain the requirement of each point having a coordinate. This particular graph was created in GeoGebra[45] **and does not correctly illustrate the coordinates used in Godot.**

its own coordinates on the XYZ-axis. This is done for every point of the mesh until an even cube is created. These coordinates can then be connected to form vertices, and later faces, to shape the mesh into a 3D cube. In Godot, this particular style of mesh would be called an ArrayMesh. For the duration of this chapter, this mesh will be referred to as a cube.
Having specified coordinates, it can be interpreted which vertices are connected to each other to form a face. This is important for object manipulation, as instead of scaling both sides of the cube-like it was described previously, the coordinates are altered instead, moving the specified face. This avoids the problem of scaling both sides because instead of scaling, coordinated vertices are moved to one specified direction.
With everything described, the next step was to allow the user to manipulate the cube. First, moving the object will be described. CollisionShape[46] and RigidBody[47] were used to help implement the feature. CollisionShape, as the name would apply, allows for the creation and editing of collision shapes in a 3D space. A more important feature of the CollisionShape node is the ability to give an area of detection, which facilities signaling when something enters its area. In order to define this area, the object has to have the RigidBody node attached to it. In brief, the RigidBody node implements full 3D physics. Intuitively, a collision shape can be formed to perfectly match the dimensions of the cube, meaning everything that enters the area of the cube will be detected. This is important for the briefly mentioned feature in Model Construction - raycast. The RayCast[48] node is a line with a predefined length, which is used to query the 3D space in order to find the closest object along its path. It calculates the intersection of any collision area, which is what was used in the thesis to see whether the user is pointing and colliding with any of the cubes. The cooperation of all of these nodes is what allows the user to move the cube around.
Second, mesh manipulation or scaling will be described. Model Construction hinted at borders surrounding the cube, which the user is allowed to interact with to scale the cube as it would seem fit. The borders themselves are meshes with CollisionShapes and RigidBodies. The key difference is that each border knows which vertice coordinates it needs to adjust to facilitate proper scaling. Once a ray is intersecting one of the borders, it signals to the cube which vertices need to be adjusted and in which direction. In addition to the cube moving its vertices, it also needs to correctly adjust the other borders. This is done by defining neighboring borders and how each border should manipulate itself when one of its neighbors is being moved.
While the concept of Individual Object Manipulation using CSGMeshes worked and was given a lot of thought, it was decided that the block-based building would be a better approach for the following reasons:

[46] https://docs.godotengine.org/en/stable/classes/class_collisionshape.html
[47] https://docs.godotengine.org/en/stable/classes/class_rigidbody.html?highlight=rigidbody
[48] https://docs.godotengine.org/en/stable/classes/class_raycast.html

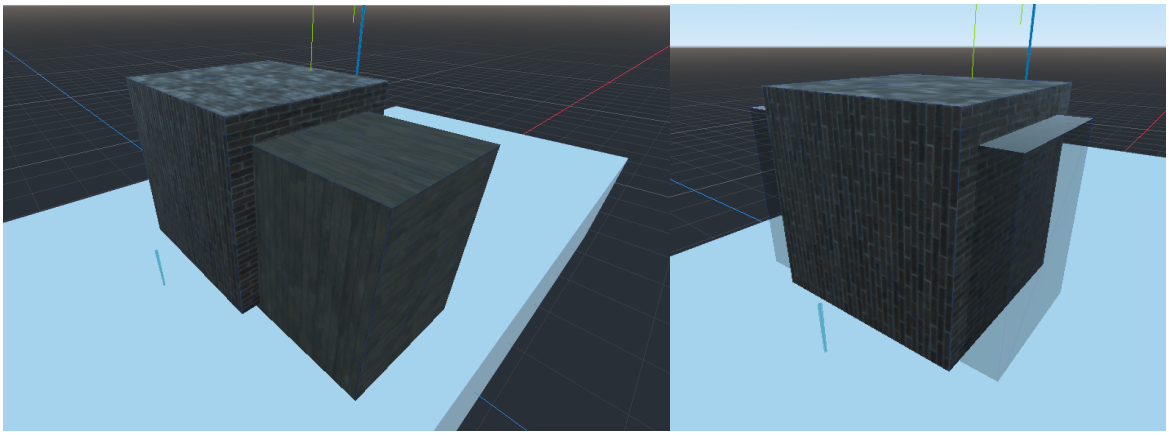

Figure 22. The intersection of two cubes with solid material on the left. The intersection of two cubes with one being solid material and the other being a transparent material.

1. The Individual Object Manipulation stretches the cubes based on the position of the user. In other words, the direction of the joystick would be used as a direction to where the object should be stretched, but it also had to take into consideration the position of the user's head and rotation. This led to a lot of movement from the user's side, causing dizziness.

2. The block-based approach was deemed simple, hence more understandable to the user. Due to its simplicity and intuitiveness, the flow of building a new model was deemed better.

5.4 Glass Material

One of the challenges of this thesis was the implementation of the glass material. In the real world, glass is a hard, brittle substance that is typically transparent. In a game engine, this material would also be transparent and allow users to see through it. The difficulty presented itself in the Model Building, Individual Object Manipulation portion of this chapter.
As explained in the Implementation, Model Construction sub-chapter, the user is allowed to place objects, and manipulate them as would be desired. Consequently, this allows the user to place two or more objects that would intersect one another. The challenge was to make blocks intersect in an intuitive matter that would look natural to the user.
As can be seen, Figure 22 shows the interaction of solid materials on the left. This interaction did not present any problem and was left as is. The problem comes with the interaction on the right of Figure 22. The user could expect that the glass material would make the inside of the solid material see-through. Even though this assumption is not a

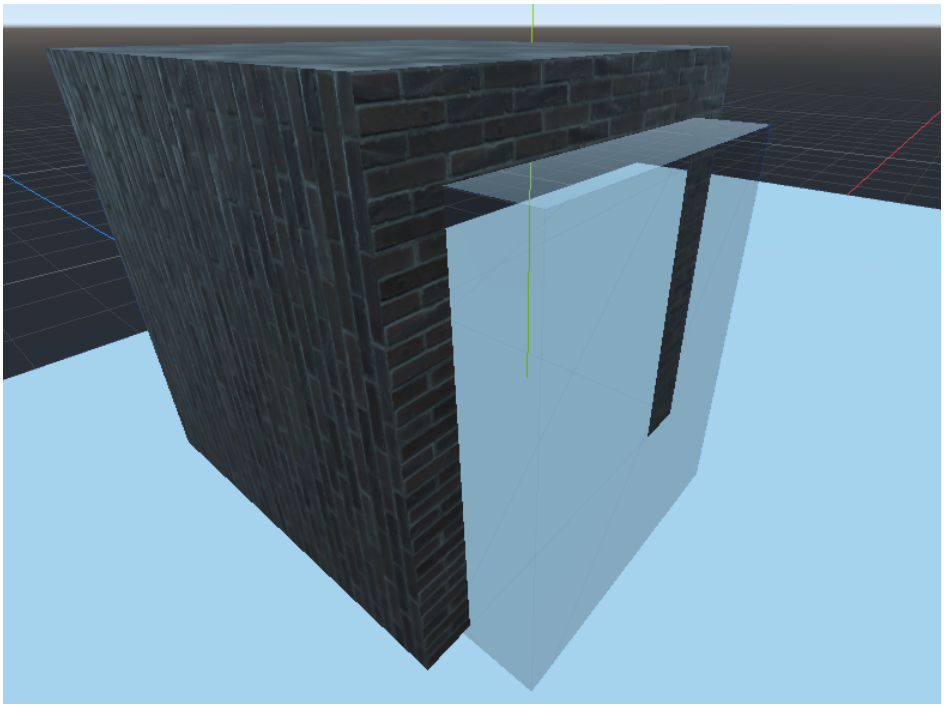

Figure 23. CSGMesh intersection with the Union operation turned on. The glass material makes the inside of the solid material invisible.

guarantee, that is what was speculated would be the most intuitive reaction when placing glass materials inside solid materials.
The Godot game engine allows the manipulation of materials with different parameters. The relevant ones for this case are the following - Transparent and Albedo alpha value. These two parameters allow the material to become transparent, where the Albedo alpha value measures how transparent the material becomes, becoming invisible at max value. This, however, does not fix the problem of intersection.
The Godot game engine also has the following nodes to help alleviate this problem - CSGCombiner and CSGMesh nodes, both of which are explained in Implementation, Object Manipulation. These two nodes allow CSGMeshes to interact with each other with a specified operation once they start intersecting. These operations include - Union, Intersection, and Subtraction. Using these two nodes with the Union operation for both CSGMeshes, we get closer to the desired result, as seen in Figure 23.
As evidenced by Figure 23, the implementations of the CSGCombiner and CSGMesh give one of the desired effects - the intersection of glass and solid material make the insides of the solid material invisible.
However, as already mentioned in Model Building, Individual Object Manipulation, the Glass Material solution provided in this chapter is not present in the final build. Instead, a

simpler solution was thought of - not using the CSGMeshes to avoid the use of the Union operation. Instead, the user retextures each individual cube without extra computational cost.

6 Summary

The research started at an Introduction - the background necessary to understand the motivation behind this paper and explaining its relevancy. It was discussed how the traditional tools used in architecture have missing aspects that they could never provide without newer technology - VR. In said chapter, a goal was defined for the remainder of this paper - create a baseline application, provide background and analyze the results. That, however, is only the goal of this particular paper. The actual goal is to make the application open-source and provide with the tools necessary to achieve an ambition task for the future - providing architects with a tool that they currently do not have.
In Related Works, it was shown how VR is used in different industries. The benefits of VR have also been stated. Due to the potential benefits of using VR and an intuitive use case for architects when it comes to architectural development in conjunction with VR, creating an application that replaces traditional architectural methods was deemed appropriate. In support of the previous statement, multiple projects of similar scope were shown, having success in the field of VR, showcasing that the goal of creating a worthwhile tool for architectural work in the near future is possible. In the aforementioned chapter, requirements were defined, to create a cohesive flow and help the reader understand what this application is capable of.
As was discussed in Implementation, three out of four requirements were fulfilled. The application can create 3D objects in a Virtual Environment, allow for its alternation, place them in a simulated world and allow the user to immersive itself in the world. With the help of Godot, Blender and Godot Oculus Toolkit, this application became a reality. It was discussed how all of these features were made. More importantly, the experiences, even though replaced, were analyzed in the Analysis chapter. The one downside was the inability to fulfill the fourth requirement - using Radiance as a separate lighting engine. On the topic of traditional architectural methods, such as Radiance, the following question can arise - do the traditional architectural methods need to be replaced by VR? The benefits of VR have been described, but it remains a question whether VR can completely replace the creation of 3D models on something like a computer. Current methods involve precise implementation of walls with exact measurements, opening, custom-made furniture, etc. It can be argued that VR cannot recreate all of the functionalities of traditional architectural methods. If that would be the case, a counterpoint can be made that VR can be an alternative solution to visualizing daytime lighting and luminance from a first-person immersive experience. However, to further support the claim that applications in VR can become a complement to traditional architectural tools, all one

needs to do is look at the previously mentioned Related Works and the achievements of this paper. An application was made by a singular person. Combine that statement with the already created projects, such as RadVR and Akrio, and one can see how a competent architectural tool can be made. As of the time of writing, a VR application was made, most of the requirements were fulfilled and significant background has been established.

References

[1] Wael A. Abdelhameed. "Virtual Reality Use in Architectural Design Studios: A Case of Studying Structure and Construction". en. In: *Procedia Computer Science* 25 (2013), pp. 220–230. ISSN: 18770509. DOI: 10.1016/j.procs.2013.11.027. URL: https://linkinghub.elsevier.com/retrieve/pii/S1877050913012325 (visited on 05/01/2022).

[2] Nada Alasbali and Boualem Benatallah. "Open source as an innovative approach in computer science education A systematic review of advantages and challenges". In: *2015 IEEE 3rd International Conference on MOOCs, Innovation and Technology in Education (MITE)*. Oct. 2015, pp. 278–283. DOI: 10.1109/MITE.2015.7375330.

[3] Simon Ammanuel et al. "Creating 3D models from Radiologic Images for Virtual Reality Medical Education Modules". eng. In: *Journal of Medical Systems* 43.6 (May 2019), p. 166. ISSN: 1573-689X. DOI: 10.1007/s10916-019-1308-3.

[4] Hasan Gokberk Bayhan and Ece Karaca. "Technological innovation in architecture and engineering education - an investigation on three generations from Turkey". In: *International Journal of Educational Technology in Higher Education* 17.1 (Aug. 2020), p. 33. ISSN: 2365-9440. DOI: 10.1186/s41239-020-00207-0. URL: https://doi.org/10.1186/s41239-020-00207-0 (visited on 05/01/2022).

[5] Bee Breeders. *The Use of Virtual Reality in Architecture*. en. URL: https://architecturecompetitions.com/the-use-of-virtual-reality-in-architecture/ (visited on 05/10/2022).

[6] Robert L Cook, Thomas Porter, and Loren Carpenter. "Computer Graphics Volume18, Number3 July 1984". en. In: (1984), p. 9.

[7] M Fadjri and Y A Ekawardhani. "The Influence of Technology on Architecture Design 4.0". en. In: *IOP Conference Series: Materials Science and Engineering* 879.1 (July 2020), p. 012149. ISSN: 1757-8981, 1757-899X. DOI: 10.1088/1757-899X/879/1/012149. URL: https://iopscience.iop.org/article/10.1088/1757-899X/879/1/012149 (visited on 05/01/2022).

[8] Belen Jiménez Fernández-Palacios, Daniele Morabito, and Fabio Remondino. "Access to complex reality-based 3D models using virtual reality solutions". en. In: *Journal of Cultural Heritage* 23 (Jan. 2017), pp. 40–48. ISSN: 1296-2074. DOI: 10.1016/j.culher.2016.09.003. URL: https://www.sciencedirect.com/science/article/pii/S1296207416301856 (visited on 04/11/2022).

[9] Mohammad Keshavarzi, Luisa Caldas, and Luis Santos. "RadVR: A 6DOF Virtual Reality Daylighting Analysis Tool". en. In: *Automation in Construction* 125 (May 2021), p. 103623. ISSN: 0926-5805. DOI: 10.1016/j.autcon.2021.103623. URL: https://www.sciencedirect.com/science/article/pii/S0926580521000741 (visited on 05/01/2022).

[10] Myung Ja Kim, Choong-Ki Lee, and Timothy Jung. "Exploring Consumer Behavior in Virtual Reality Tourism Using an Extended Stimulus-Organism-Response Model". en. In: *Journal of Travel Research* 59.1 (Feb. 2020). Publisher: SAGE Publications Inc, pp. 69–89. ISSN: 0047-2875. DOI: 10.1177/0047287518818915. URL: https://doi.org/10.1177/0047287518818915 (visited on 05/19/2022).

[11] Rafa\l Krupiński. "Virtual reality system and scientific visualisation for smart designing and evaluating of lighting". In: *Energies* 13.20 (2020). Publisher: Multidisciplinary Digital Publishing Institute, p. 5518.

[12] Martijn J. Schuemie et al. "Research on Presence in Virtual Reality: A Survey". en. In: *CyberPsychology & Behavior* 4.2 (Apr. 2001), pp. 183–201. ISSN: 1094-9313, 1557-8364. DOI: 10.1089/109493101300117884. URL: http://www.liebertpub.com/doi/10.1089/109493101300117884 (visited on 05/06/2022).

[13] Michelangelo Scorpio et al. "Virtual reality for smart urban lighting design: Review, applications and opportunities". In: *Energies* 13.15 (2020). Publisher: Multidisciplinary Digital Publishing Institute, p. 3809.

[14] Mel Slater and Martin Usoh. "Representations Systems, Perceptual Position, and Presence in Immersive Virtual Environments". en. In: *Presence: Teleoperators and Virtual Environments* 2.3 (Jan. 1993), pp. 221–233. ISSN: 1054-7460. DOI: 10.1162/pres.1993.2.3.221. URL: https://direct.mit.edu/pvar/article/2/3/221-233/58839 (visited on 05/06/2022).

[15] Manuel Spitschan et al. "Variation of outdoor illumination as a function of solar elevation and light pollution". en. In: *Scientific Reports* 6.1 (June 2016). Number: 1 Publisher: Nature Publishing Group, p. 26756. ISSN: 2045-2322. DOI: 10.1038/srep26756. URL: https://www.nature.com/articles/srep26756 (visited on 06/16/2022).

[16] Iis P. Tussyadiah et al. “Virtual reality, presence, and attitude change: Empirical evidence from tourism”. en. In: *Tourism Management* 66 (June 2018), pp. 140–154. ISSN: 0261-5177. DOI: 10.1016/j.tourman.2017.12.003. URL: https://www.sciencedirect.com/science/article/pii/S0261517717302662 (visited on 05/06/2022).

[17] Don Werthmann. “The Kelvin Scale Defines the Color Temperature of Light”. en. In: (), p. 1.

[18] *What is open source? | Opensource.com*. en. URL: https://opensource.com/resources/what-open-source (visited on 05/06/2022).

[19] Bob G. Witmer and Michael J. Singer. “Measuring Presence in Virtual Environments: A Presence Questionnaire”. en. In: *Presence: Teleoperators and Virtual Environments* 7.3 (June 1998), pp. 225–240. ISSN: 1054-7460. DOI: 10.1162/105474698565686. URL: https://direct.mit.edu/pvar/article/7/3/225-240/92643 (visited on 05/06/2022).

[20] Ming-Wei Wu and Ying-Dar Lin. “Open source software development: an overview”. In: *Computer* 34.6 (June 2001). Conference Name: Computer, pp. 33–38. ISSN: 1558-0814. DOI: 10.1109/2.928619.

[21] Martin; Jayemanne Zeilinger. “‘Chaotic Lawful’: Teaching Videogames in a Licensing and Permissions Lacuna”. In: *The Journal of Cinema and Media Studies* 60.7 (2021). ISSN: Online IS. DOI: https://doi.org/10.3998/jcms.18261332.0060.709. URL: http://hdl.handle.net/2027/spo.18261332.0060.709.

Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Aleksander Nikolajev**,

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

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Lighting Design in Virtual Reality,

   supervised by Ulrich Norbisrath.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Aleksander Nikolajev
08/08/2022