EFFECTIVENESS OF INDOOR DAYLIGHT REPLICATION IN VIRTUAL REALITY

An Undergraduate Research Scholars Thesis

by

LIAM BESSELL

Submitted to the LAUNCH: Undergraduate Research office at
Texas A&M University
in partial fulfillment of the requirements for the designation as an

UNDERGRADUATE RESEARCH SCHOLAR

Approved by
Faculty Research Advisor: John Keyser

May 2021

Major: Computer Science

RESEARCH COMPLIANCE CERTIFICATION

Research activities involving the use of human subjects, vertebrate animals, and/or biohazards must be reviewed and approved by the appropriate Texas A&M University regulatory research committee (i.e., IRB, IACUC, IBC) before the activity can commence. This requirement applies to activities conducted at Texas A&M and to activities conducted at non-Texas A&M facilities or institutions. In both cases, students are responsible for working with the relevant Texas A&M research compliance program to ensure and document that all Texas A&M compliance obligations are met before the study begins.

I, Liam Bessell, certify that all research compliance requirements related to this Undergraduate Research Scholars thesis have been addressed with my Research Faculty Advisor prior to the collection of any data used in this final thesis submission.

This project did not require approval from the Texas A&M University Research Compliance & Biosafety office.

TABLE OF CONTENTS

ABSTRACT

Effectiveness of Indoor Daylight Replication in Virtual Reality

Liam A. Bessell
Department of Computer Science and Engineering
Texas A&M University

Research Faculty Advisor: John Keyser
Department of Computer Science and Engineering
Texas A&M University

Texas A&M University

This paper explores ways of letting users interact with daylighting spaces in a virtual reality (VR) environment. Two methods for viewing daylighting in VR are presented, both of which use a physically-based raytracer to generate daylighting images and a game engine for viewing them. The first method creates a 360° panorama of the space at a particular point. This is then extended to generate multiple renderings from different locations in the scene, allowing users to view the space from different positions. The second method creates a texture for each polygon face in the scene. This approach allows users to freely move around the scene at the cost of losing the specular component of the textures. Finally, a user study is proposed to compare the two methods.

DEDICATION

This thesis is dedicated to the many family and friends who have supported me throughout my journey.

ACKNOWLEDGMENTS

I would first like to thank my faculty advisor, Dr. Keyser, for his incredible guidance and hours of support throughout the research process. I would also like to express my gratitude to Dr. Beltran for her feedback and knowledge on architectural topics. My thanks also go to the Department of Computer Science and Engineering and the Undergraduate Research Scholars program for supporting this research. Finally, a fond thank you to my family and friends for their continuous encouragement.

No funding was provided for this project, but it was developed under the *Human Centric Daylighting based on circadian stimulus, view, visual comfort, and lighting needs* grant from the Texas A&M University Triad Program.

NOMENCLATURE

VR	Virtual reality
HMD	Head mounted display
HDR	High dynamic range

1. INTRODUCTION

Daylighting is the use of windows, skylights, and other methods to bring sunlight into an indoor space, be that direct or indirect. For example, a room with large windows and a skylight will have large amounts of daylighting, whereas a basement with only a small window will have little. Good daylighting is essential for spaces people spend a lot of time in, as it has been shown to increase overall well-being as well as save energy [1].

1.1 Daylighting simulations

The first methods for daylighting simulation involved constructing miniature models of the building. These models could then be moved and examined in the real world to simulate the daylight conditions of the final construction. With advancements in computing power and computer graphics, standard simulations became virtual. Most commonly, physically-based ray tracers are used to create renderings which can then be viewed on a display device. Radiance is the standard tool used to simulate daylighting [2]. It gives users immense control over the simulation and output while creating physically accurate results.

The goal of daylighting design is generally to optimize the amount of daylight in the room while minimizing glare. Illuminance metrics are used to determine daylight sufficiency. However, illuminance is not accurate with regards to measuring glare, because it quantifies the amount of light incident on a surface [3]. Instead the standard metric for quantifying glare is luminance, which quantifies the amount of light perceived by the eyes. For computer simulations this requires an image format that can convey large differences in luminance.

High dynamic range (HDR) images are rendered such that the luminance values are mapped to each pixel, thereby giving the user an understanding of the visual discomfort in the scene. HDR images are necessary because standard dynamic range (SDR) images can't convey glare as well, they are limited to a smaller range of values. HDR renderings allow for a greater contrast between

the dark and bright areas of an image.

1.2 VR simulations

Advances in computing power have led to exciting technologies like virtual reality (VR) devices becoming more accessible. This has brought about a surge of research and development into applications using VR systems, which allow for enhanced presence while viewing a space as opposed to a traditional display. Presence is the feeling of being somewhere else than where you physically are; it is a condition of total immersion [4]. Greater presence means that designers can have a better understanding of the daylighting conditions of the space they are working on, and thus create a design that increases human well-being and save energy.

1.3 Limitations of current VR tools

Current consumer VR systems do not support HDR. As such, HDR renderings are transformed into SDR for viewing in VR. In practice this limits the ability to convey glare to a user, as well as contrasting very dark areas from very light areas. This is overcome for visualization purposes with falsecolor renderings, which map the luminance or illuminance values for each pixel in the image to a color spectrum.

Previous work has been done on viewing daylit spaces in VR via 360° panoramas. This approach is helpful in several ways, but it is limited in that renderings must be done for each position in a scene. Hence, users can only cycle between predefined positions, rather than freely moving around the scene. This problem was partially overcome in recent work that allowed users to walk around the space, but it was still constrained to predefined locations for viewing the daylighting renderings [5].

2. METHODS

The goal of this thesis is to evaluate the effectiveness of daylighting simulations in VR. With this in mind, I first present a panoramic method that allows the user to look around the scene in all directions and move to other predefined positions. Next, I introduce a new approach that allows the user to freely move around the scene without being confined to predefined locations. Finally, I lay out a user study that compares the effectiveness of these two methods.

2.1 Panorama simulation

In 3D applications a panorama can be created in a number of ways. Panoramas are a popular method used in a variety of VR applications because they are easy to implement and allow the user to look around at their environment in all directions. For this work I generated panoramas in two ways - a cube map and an equirectangular map [6]. The principle for both of these methods is the same, namely to take a 2D image and map it to a 3D sphere.

First I implemented the cube map approach. A cube map is created by stitching together six different square images into a cube (Figure 2.1). This method produces a high quality panorama around the user, with the only artifacts being seams where the squares meet each other.

The second approach I took for the panorama method uses an equirectangular projection (Figure 2.2). Instead of requiring six distinct textures from the ray tracer the equirectangular projection just requires one. It takes the shape of a rectangle with a width to height ratio of 2:1. The major downside is that the poles on the texture get distorted laterally when projected onto a sphere. This approach was eventually chosen as it was better supported by the game engine.

My implementation takes inspiration from previous work with a couple key differences [5]. In my case, the user can move between different positions in the room by selecting their respective marker (Figure 3.3). Though still limited to predefined positions, this at least allows the user some freedom to move around the scene.

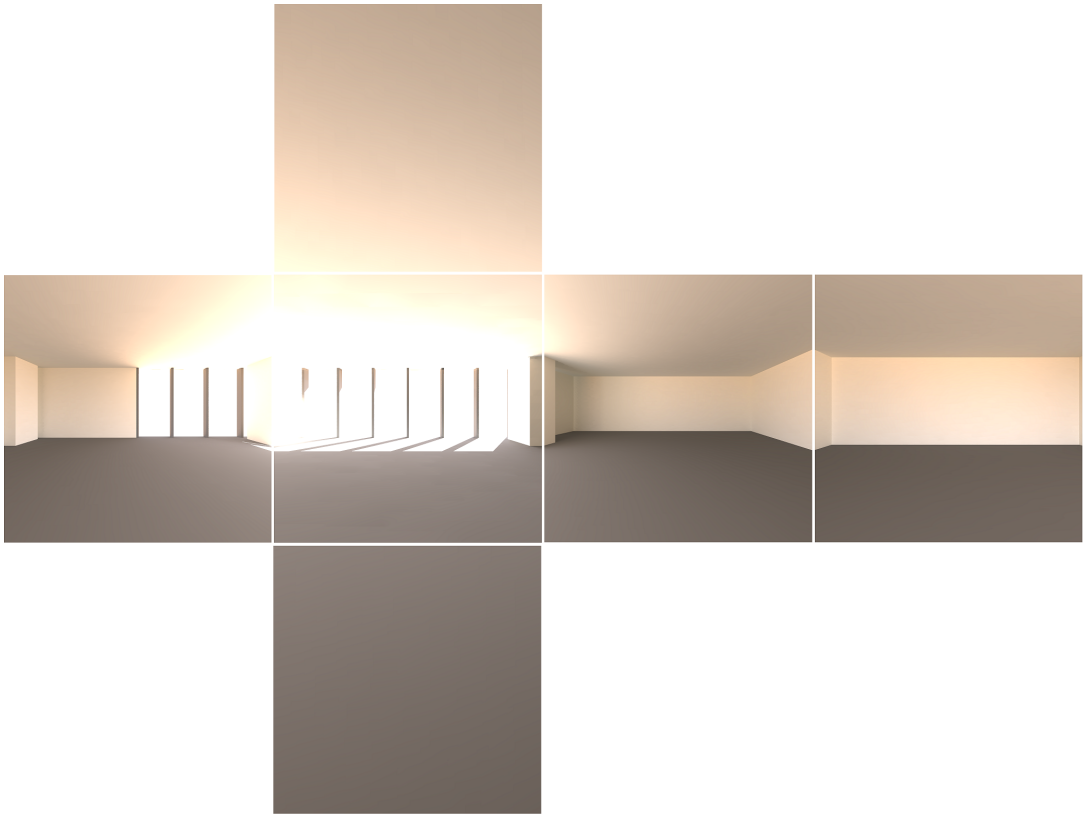

Figure 2.1: Schematic of an unwrapped cube map. In the game engine the six textures are stitched together to form a skybox around the user. This lets the user look around the scene in all directions

2.2 Free roam simulation

As discussed previously one of the appeals of VR is the enhanced presence it gives users. One way of improving presence is enabling users to move around the environment and interact with it. The panorama method thus hinders presence because users are restricted to the predefined positions in the room. Moreover, there is no true interaction with the environment, such as colliding with walls. With this in mind I developed a novel method that allows users to move to any position in the scene and interact with the environment.

The main drawback of this simulation is that the textures do not have a specular component. Thus, it is not ideal in scenes with a large amount of glossy materials, or in cases where the user is focused on specularity. The reason for this is because the specular component of renderings is

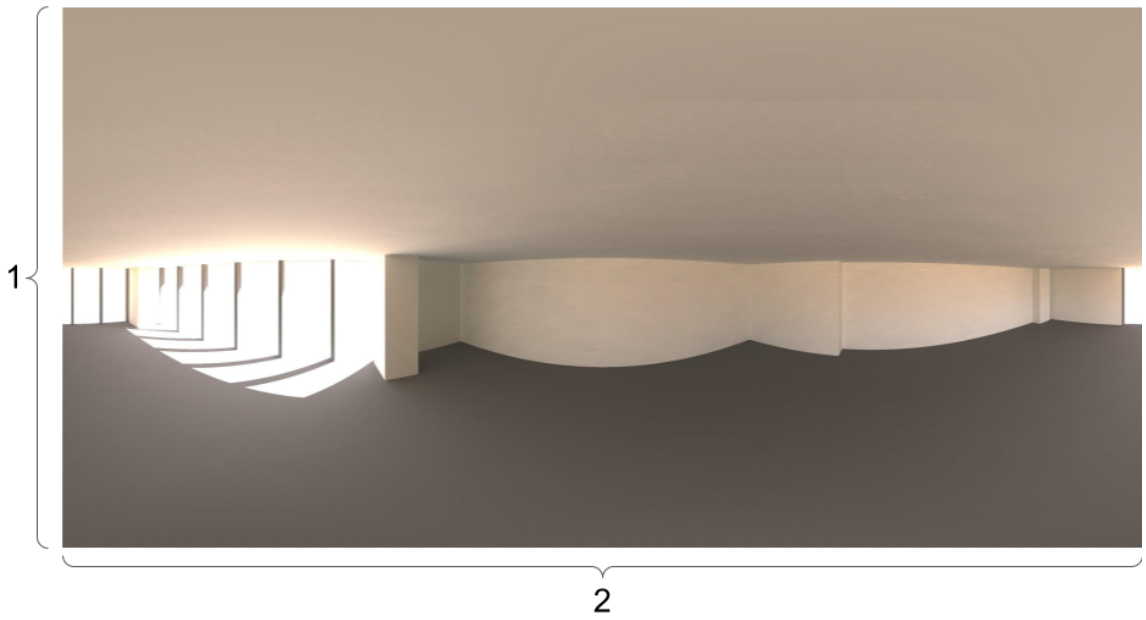

Figure 2.2: Equirectangular map rendering used in the panorama method, marked by its 2:1 ratio between the width and height. The texture is projected onto a sphere around the user which allows them to look around in all directions.

viewpoint dependent, while the diffuse component is viewpoint independent.

2.2.1 Parallel projection view generation

A custom algorithm was implemented in Python to automate the process of generating the textures and model. While it's reasonable to manually render each face for simple scenes, it becomes incredibly time consuming to do so for more complicated ones. The algorithm takes in a geometry file as its input and then generates (1) a Radiance defined parallel projection view for each face and (2) a Wavefront MTL (materials) and OBJ (object) file.

The script was designed to read in a RAD file generated by the Rhino 3D software. The RAD file provides a description of the scene, including the geometry and materials, to Radiance. First every quad - that is, a geometric face in the shape of a rectangle - is read and stored in a list. Quads can either be defined by four vertices - top left, top right, bottom right, bottom left - or six vertices. The latter case is made up of two triangles where their hypotenuse is shared and spans

the diagonal of the quad. During this step the Radiance material assigned to each face is read and stored.

Once the file reading is done the next step is to generate the parallel projection Radiance view for each face. For a parallel projection the projection plane is parallel to the face. The view is then defined as having a length and height equal to that of the face. This means the final texture's size will cover the entire geometric face. Finally the clipping planes are set to just before and after the face. This means that only the face, and not anything in front of it, will appear in the rendering. Each face in the RAD file also has a name assigned to it, which is used to identify the Radiance view. The final Radiance view produces an HDR texture of the entire face after being rendered.

After each view has been defined, the final step is to create the OBJ and MTL files. In the OBJ file, every face is defined by four vertices (unlike the RAD file where faces sometimes have six vertices). Each face is also assigned to a material in the MTL file that directly corresponds to the HDR texture that will be generated by its Radiance view (Figure 2.3). Finally, in the MTL file a material is created for each Radiance view.

2.2.2 Ray tracer

Once the parallel projection views have been generated, the next step is to use them with the physically-based ray tracer (Radiance). This is the most computationally intensive part of the process, because a different rendering has to be done for each face of the model. After the renderings have been completed there will be an HDR file for each face.

2.2.3 FBX model generation

Blender is an open-source 3D software tool set used to generate models, visual effects, and more. In this work it is used to create the final FBX (filmbox) model file of the scene which is imported into the game engine. This process is simple, all that it requires is the user to import the OBJ file into Blender and then export it as an FBX file. Blender automatically applies the textures

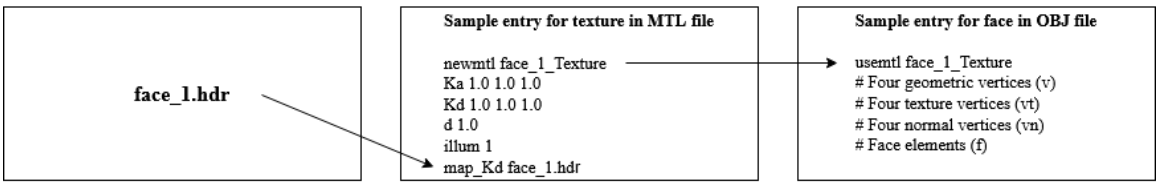

Figure 2.3: Schematic of how the free roam OBJ file entries map to their textures. Each geometric face in the OBJ file has its own material defined in the MTL file. Moreover, every face has a corresponding HDR texture rendered with the parallel projection view in the ray tracer. The HDR texture is used as the material's diffuse map.

generated by Radiance to the model because they are specified as part of each geometric face's material in the MTL file. This step is not necessary if the game engine being used can import the generated OBJ and MTL file.

2.2.4 Game engine

The game engine drives the application and allows the user to interact with the scene in VR. The FBX model is imported into the game engine project, after which it is placed in the scene. Next collision boxes are added around either the imported geometry or set up manually.

A teleportation locomotion method was used in the free roam method to enable users to move around the scene. Users point and hold down the trigger on their controller where they want to travel to in the scene. A marker appears showing where the user will move to, and then when they release the trigger they teleport to the marker. The user is unable to teleport past collision objects like walls or windows.

Both the Unity and Unreal game engines were tested before resolving on the Godot open-source game engine [7], [8]. Unity worked for the cube map panorama simulation, but not for the free roam method because it couldn't import a large number of the HDR textures. Unreal imported all the HDR textures successfully, but more work was required after the fact to connect the textures to the model. Godot was settled upon because it both supported the VR system and seamlessly

imported the FBX model with no more setup.

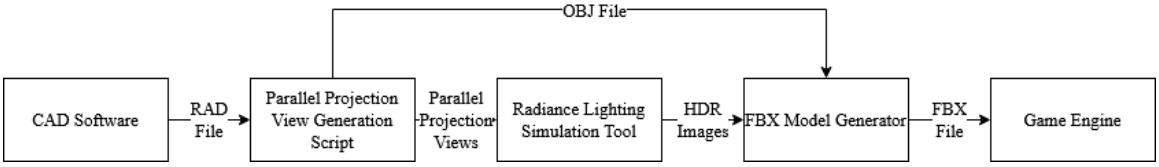

Figure 2.4: The free roam method pipeline starts with a RAD geometry and material file and outputs an FBX file. First an OBJ file is generated where each geometric face is an entry, and parallel projection views are defined for each face. Next Radiance generates HDR textures for each parallel projection view. Finally those HDR textures are applied to the OBJ file and an FBX model is generated, which can be used in the game engine.

2.3 User study

The goal of the user study is to compare the two methods in three categories: reported presence, perceptual impressions, and physical symptoms. The combination of these categories provides a holistic idea of how effective each method is in simulating daylighting in VR. Presence is defined as the sense of being somewhere else than where you physically are [4]. In the case of virtual experiences, it is akin to being fully immersed in the virtual world and feeling like you are there. Perceptual impressions are simply how people perceive the virtual space, such as how pleasant it is. Finally physical symptoms are the physical side effects of being in the virtual reality space, such as how fatigued you feel.

2.3.1 Study design

First, participants go through both the panorama and free roam simulations. Half of the participants start with the panorama simulation while the other half start with the free roam simulation. This minimizes variation between the answers of individual participants. After each simu-

lation participants fill out a questionnaire.

There are three classes of questions asked in the post-simulation questionnaire (Table 2.1). The reported presence questions come from the igroup presence questionnaire (IPQ) [9]. These questions were chosen because they all measure presence in a slightly different way. The idea is that a more objective measure of presence will be arrived at by aggregating the results from all the questions. The next set of questions measure perceptual impressions. The final set of questions measure physical symptoms. The perceptual impressions and physical symptoms questions come from Chamilothori et al. [10].

2.3.2 Study conduct

First, participants are told about the study and its associated risks. After that they are asked to fill out a consent form. Next they are shown how to use the VR controllers and headset. Participants are told they can look around the scene simply by moving their head with the VR headset on like they would in real life. In the free roam simulation participants are instructed they move around the scene by holding down either controller's trigger and releasing it once the spot they want to move to is marked. Conversely, in the panorama simulation participants are informed they move to different predefined positions by looking directly at their marker and then pressing either controller's trigger.

Next, participants put on the headset and grab the controllers with the assistance of research personnel. While wearing the headset participants are standing in an open space and told not to move around with their feet. When the participant is ready the research personnel loads them into the first simulation. To familiarize participants with the simulation they are told to touch a blue box in the virtual scene. After the participant is done with the simulation they are once again assisted in taking off the headset and fill out the paper questionnaire (Table 2.1). This process is completed once more for the second simulation, after which the study is complete.

2.3.3 Hypotheses

H1: Higher presence will be reported from the free roam simulation.

H2: Higher perceptual impressions will be reported from the panorama simulation.

H3: Worse physical symptoms will be reported from the free roam simulation.

First, **H1** follows from the higher interactivity of the free roam simulation. Sense of presence increases the more one is able to interact with their environment. Since users can freely move about the environment in the free roam simulation rather than being confined to predefined positions, I posit this method will have higher presence.

Next, **H2** was chosen because of the enhanced fidelity of the panorama simulation. Recall that this method has specular highlights, which I hold will play a large factor in the positive perception of the virtual space.

Finally, **H3** originates from the enhanced interactivity of the free roam simulation. I believe that users will suffer more physical symptoms as they move around the space more.

2.3.4 Data analysis

A metric for each category being tested is produced based on a participant's answer for that questionnaire. So, for each questionnaire there will be three metrics: reported presence, perceptual impressions, and physical symptoms. The metric is computed by summing up the response for each question in the category. Both "I felt like I was perceiving pictures" and "How fresh does your head feel" are treated as negative for their respective metric calculations. A higher value for reported presence and perceptual impressions is considered better, while a higher value for physical symptoms is considered worse.

The goal sample size, n, for this study is at least 20 adults from the general population. Once the data is collected a sample mean and sample standard deviation will be calculated for each metric. Between the two methods and three metrics there will be six sample means in total.

A one-tailed test with $\alpha = 0.05$ and $df = n - 1$ (degrees of freedom) will be conducted for each hypothesis. Since $n < 30$, a T-Value will be used.

I will now write an example of the analysis to be run on all of the hypotheses. Assume the study has been ran with 20 participants and **H1** is now being tested. Thus, $df = 20 - 1 = 19$ and $t_{\alpha,df} = 1.729$. The reported presence sample mean is 7.5 for the free roam simulation and 5.0 for the panorama simulation. The sample standard deviations are 2.1 and 1.9 respectively. The null hypothesis is the negation of **H1**, so "Higher presence will be reported for the panorama simulation". Now, we use the formula $T = (\bar{X} - \mu)/(s/\sqrt{n})$ to compute the test statistic. Since we are assuming the null hypothesis we use the panorama simulation sample mean as μ, while $\bar{X}$ and s are the free roam simulation's sample mean and sample standard deviation. Hence, $T = (7.5 - 5.0)/(2.1/\sqrt{20}) = 5.32$. Since $T > t_{\alpha,df}$, we can reject the null hypothesis and thus accept **H1**.

Table 2.1: Questionnaire

Question	Anchors*
Reported Presence	
I felt like I was just perceiving pictures.	Fully disagree–Fully agree
I felt present in the virtual space.	Fully disagree–Fully agree
I was not aware of my real environment.	Fully disagree–Fully agree
How real did the virtual world seem to you?	Not real at all–Completely real
Perceptual Impressions	
How pleasant was the space?	Not at all–Very much
How interesting was the space?	Not at all–Very much
How exciting was the space?	Not at all–Very much
Physical Symptoms	
How sore do your eyes feel?	Not at all–Very much
How fresh does your head feel?	Not at all–Very much
How fatigued do you feel?	Not at all–Very much

* A scale from 1 to 5, with the anchor points at both ends, is used for the questions.

2.4 Device and software

2.4.1 Device specifications

The device used for this research has a Intel Core™ I7-4770K CPU @ 4.20 GHz with a GTX 1070 GPU and 16 GB of RAM. The Oculus Rift Consumer Version 1 was used as the VR system.

2.4.2 Implementation specifics

The validated ray tracer Radiance was used to create the daylighting HDR images in this research. These HDR images were then applied to models and imported into the Godot game engine to be viewed in VR.

3. RESULTS

3.1 Render time

Table 3.1 shows the CPU time (user and kernel mode), in seconds, to render each method in Radiance. The time for one equirectangular map to be rendered is reported in the panorama column. Two scenes were used for testing. The first is a simple room in the shape of a rectangle with two windows. This scene has 12 geometric faces. The second scene is a more complex office space with 253 geometric faces. This means that 253 textures must be rendered for the free roam method. Table 3.2 shows the rendering parameters used for the various quality levels.

For the panorama method the low and medium quality times are identical between the two scenes. In contrast, the render time for the high quality rendering of the office space is nearly twice that of the rectangular room. This shows that the high quality render time is greatly affected by the complexity of the scene.

Since the free roam method generates a texture for each geometric face there is a large discrepancy between the render times of the two scenes for it. Unfortunately there is not a clear relation in the rate of growth of either the office space and rectangular room free roam renderings, or between the free roam and panorama renderings.

A clear advantage of the panorama method is that, if only one or a few panoramas are being generated, it takes considerably less time to render the textures for it in complex scenes. However, there is a critical point in the amount of panoramas rendered where this advantage is lost.

3.2 Screen captures

Figure 3.1, Figure 3.2, and Figure 3.3 present the cube map version of the panorama method in the Unity game engine. The floating white sphere is a node that, when interacted with, transports the user to a different panorama rendered in the corresponding location of the scene.

Figure 3.4 and Figure 3.5 show the equirectangular map version of the panorama method

Table 3.1

Scene	Quality	Panorama*	Free Roam
Rectangular room	Low	28.1	1.8
Rectangular room	Medium	47.0	22.5
Rectangular room	High	645.6	793.7
Office space	Low	27.9	46.5
Office space	Medium	46.1	280.1
Office space	High	1,207.4	5,246.2

* Time to render one panorama texture
The CPU time (user and kernel mode), in seconds, to render the textures for both methods in Radiance. The free roam method takes significantly longer for the office space scene due to the number of textures it must generate.

Table 3.2

Quality	-dp	-ar	-ms	-ds	-dt	-dc	-dr	-ss	-st	-ab	-aa	-ad	-as	-lr	-lw	-ps	-pt
Low	128	16	0	0	.2	.25	0	0	.5	0	.3	256	0	6	.003	8	.16
Medium	512	32	0.52	.3	.1	.5	1	1	.1	1	.15	800	128	8	.0001	6	.08
High	2048	64	0.26	.2	.05	.25	.75	16	.01	2	.075	4096	2048	12	.00001	5	.04

Radiance *rpict* parameters used for rendering. See appendix B for an explanation of each parameter.

in the Godot game engine.

Figure 3.6 displays the teleport method of movement in the free roam method. Users point their VR controllers and hold down a trigger, then a path appears showing where they will move to. Once the user releases the trigger they are instantly moved to the marked location. This method for locomotion is used by other VR experiences and can be less jarring than using the joystick or keyboard to move. A more sophisticated movement system for the joystick that better matches human movement could overcome this. Namely, one that has a wind up and wind down phase at the beginning and end of the movement respectively.

Figure 3.7 and Figure 3.8 show the free roam simulation in the Godot game engine.

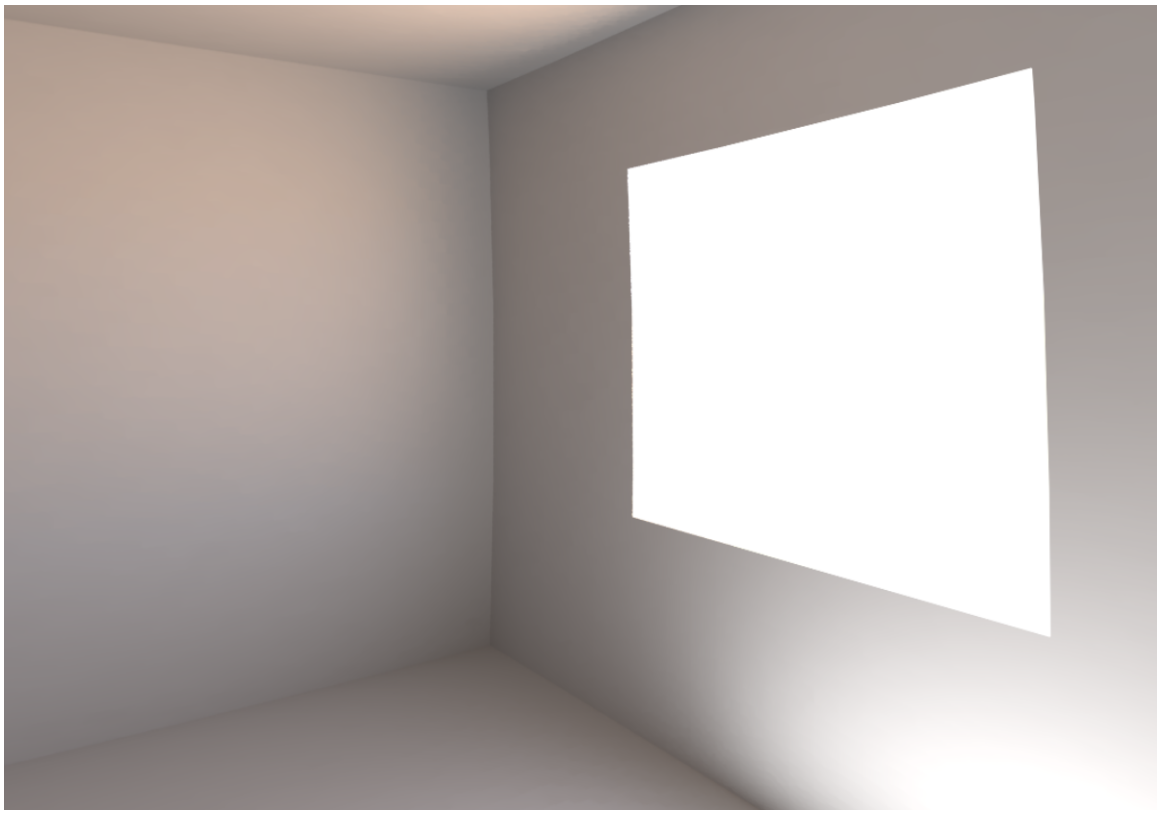

Figure 3.1: Screen capture of the panorama method in the rectangular room.

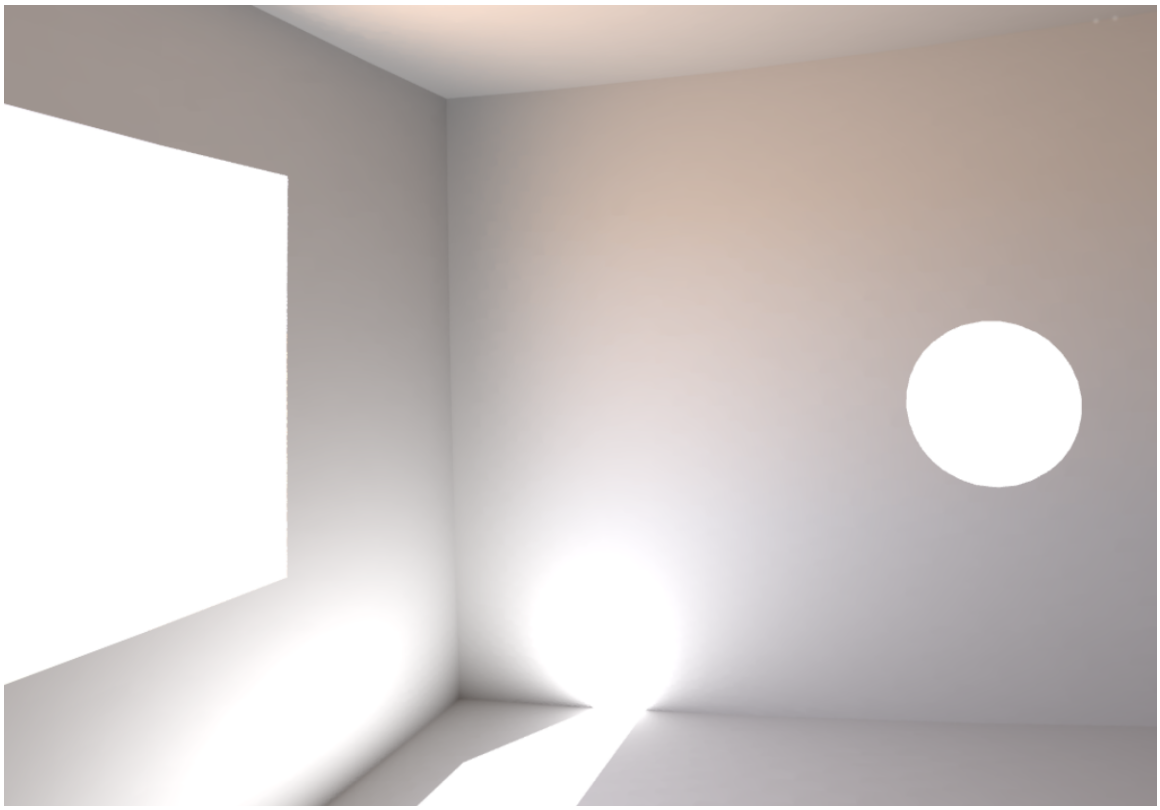

Figure 3.2: The white sphere in the panorama simulation is a node that allows the user to switch to a different position in the room (and thus a different panorama).

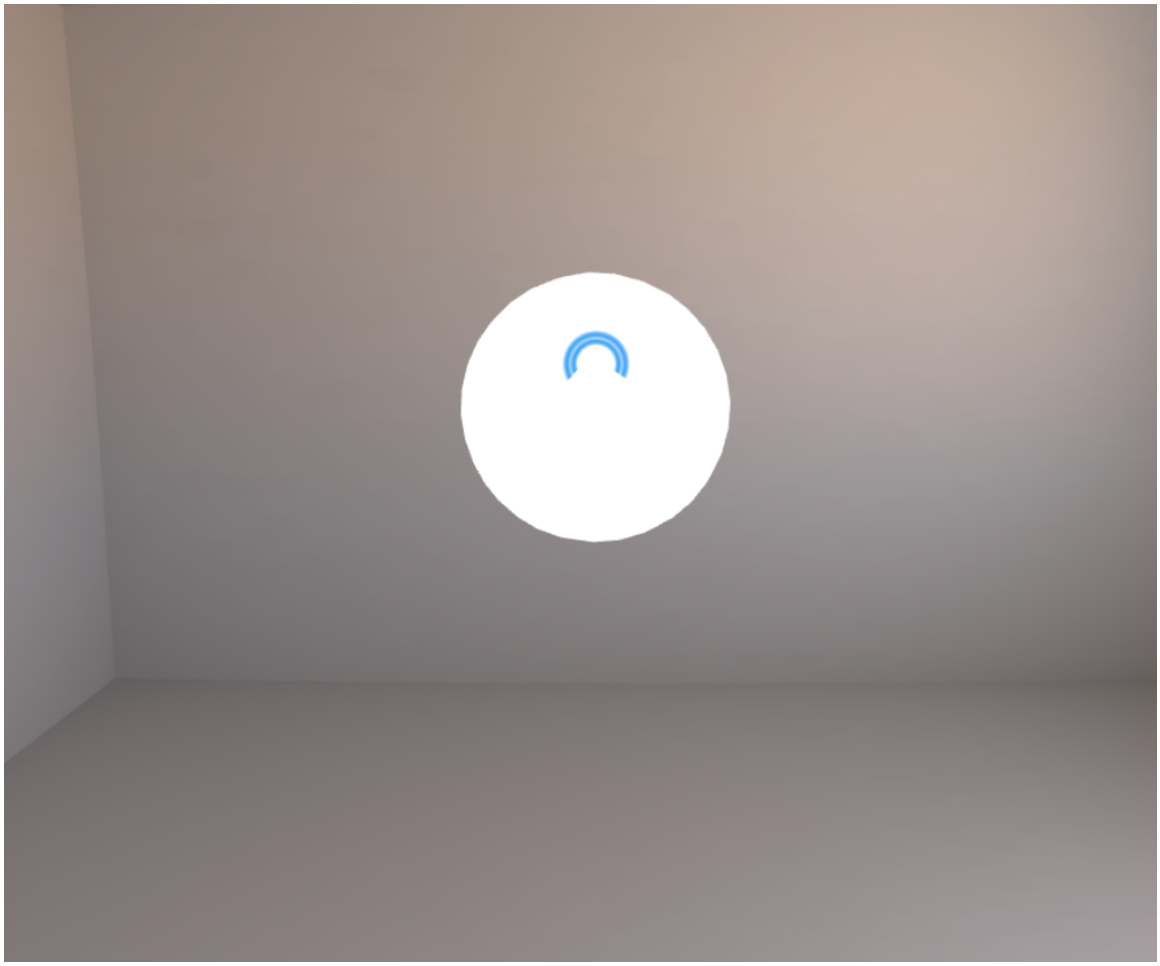

Figure 3.3: In the panorama simulation a UI indicator appears when a user looks at one of the white sphere nodes. After the user presses the trigger they are transported to the new panorama, rendered in the location of the sphere.

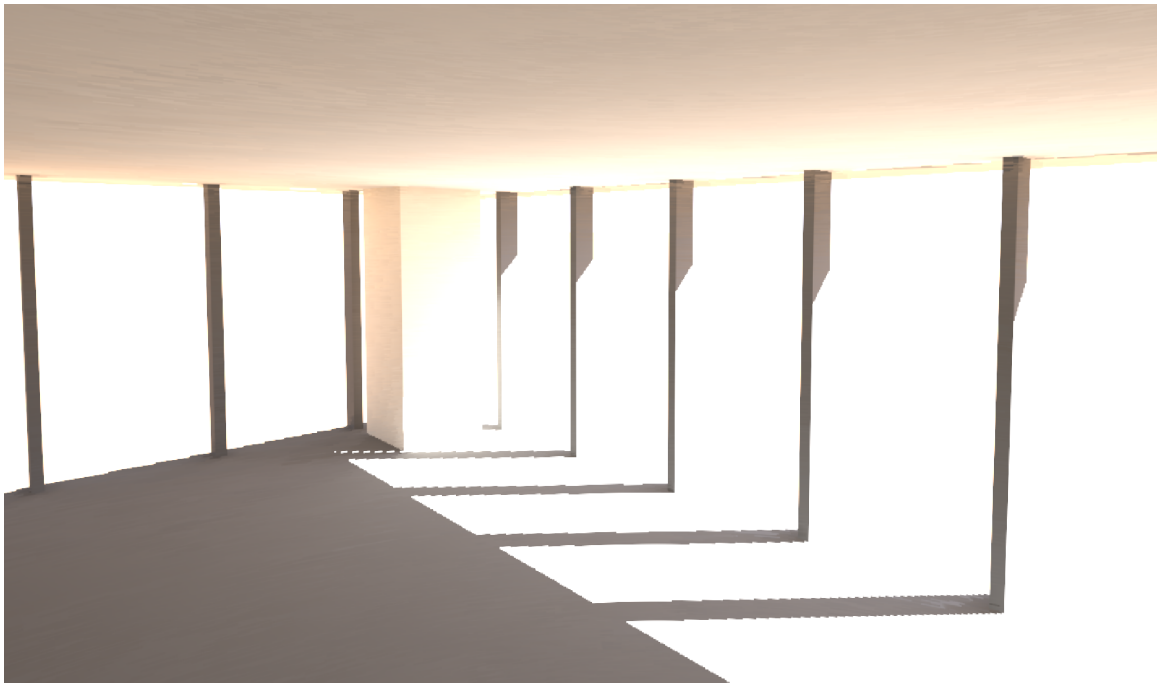

Figure 3.4: Screen capture of the panorama simulation in the office space scene. There is no geometry loaded in the game engine for this method, rather, it is a 360° panorama wrapped around the user.

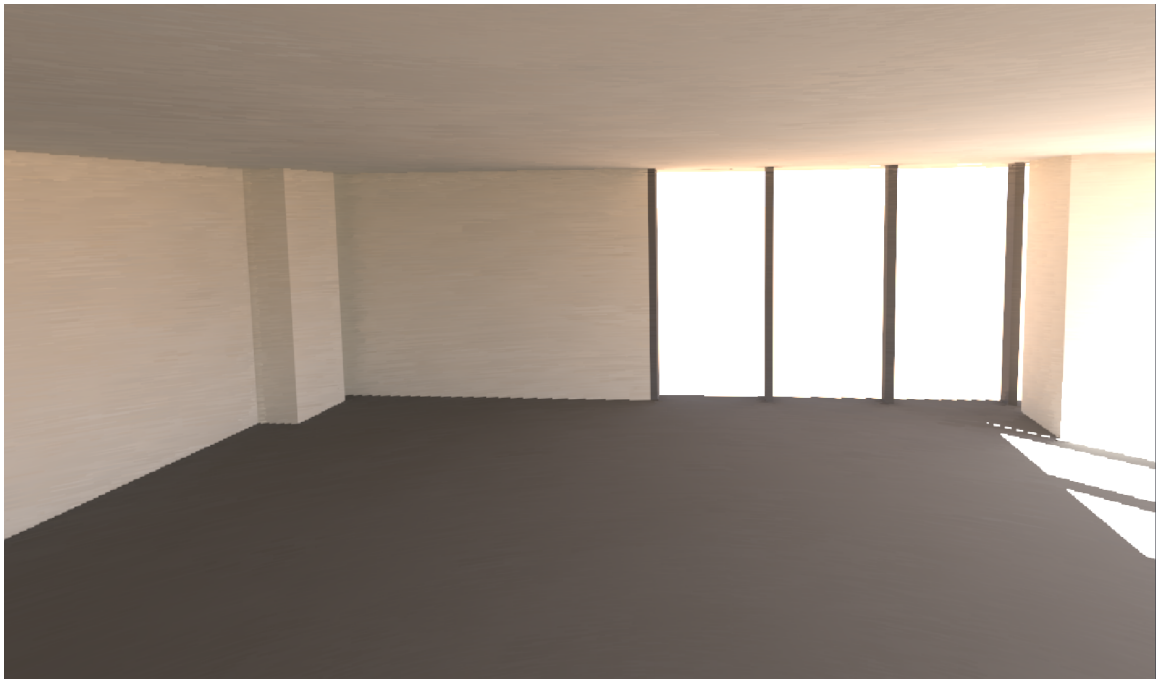

Figure 3.5: A second screen capture of the panorama method in the office space scene. Users change their view simply by moving their head around with the VR headset.

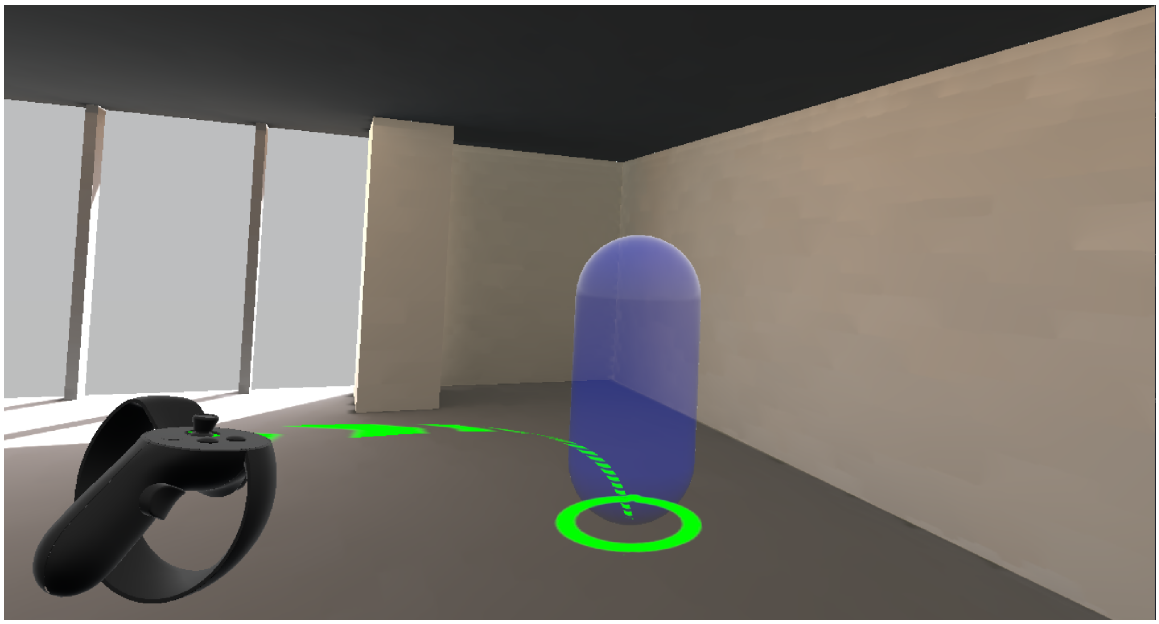

Figure 3.6: In the free roam simulation movement is achieved via teleportation. The user holds down the trigger and aims to the spot they want to move in the scene with the controller, as shown above. Once they release the trigger, they are teleported to the spot marked by the blue cylinder.

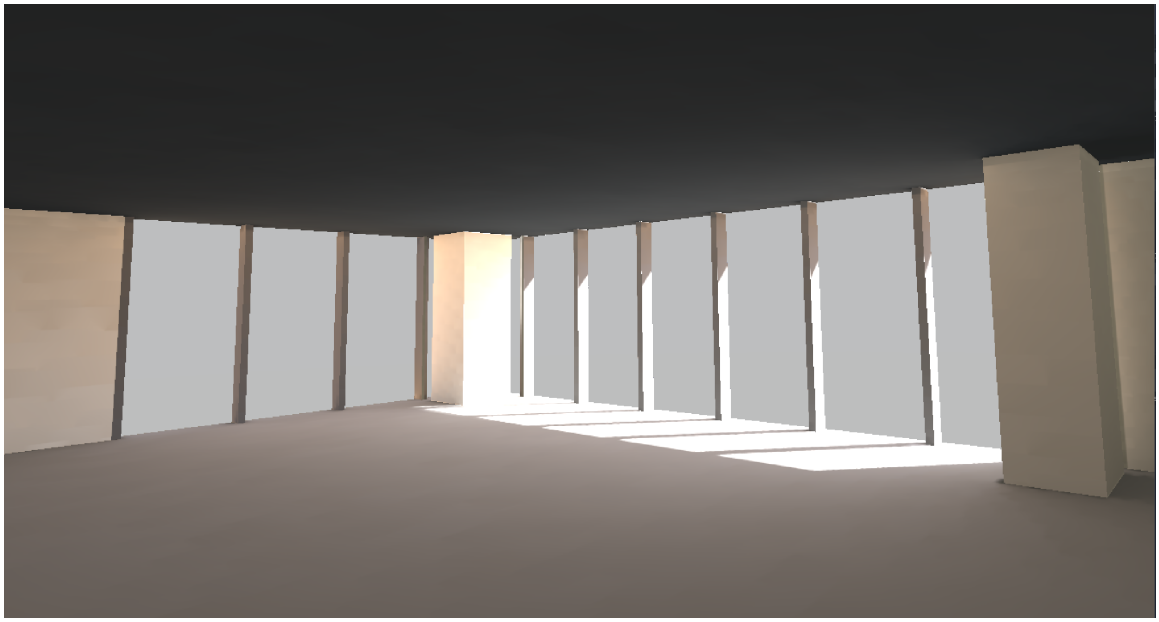

Figure 3.7: Screen capture the free roam method in the office space scene. The geometry of the office space is loaded into the game engine, and each geometric face has its own texture rendered by Radiance.

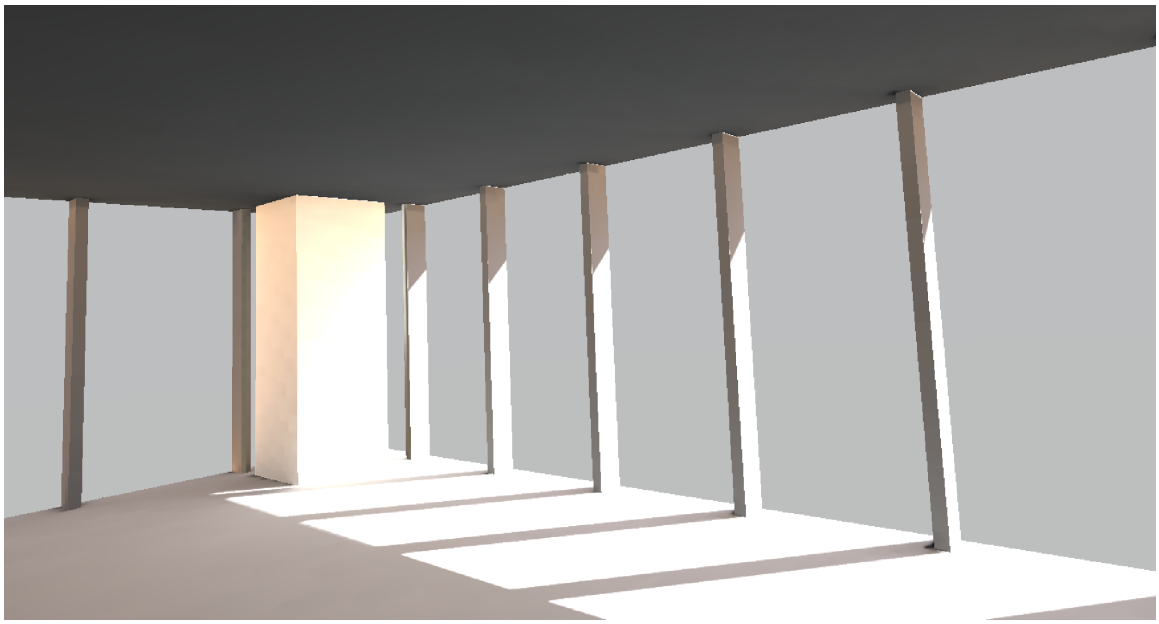

Figure 3.8: A second screen capture the free roam method in the office space scene.

4. CONCLUSION

4.1 Method comparison

Both methods have advantages and drawbacks. The panorama simulation is typically quicker to render in the ray tracer (Radiance) since it only requires one texture per position. This is in contrast to the free roam simulation which must render each geometric face to create textures for each one. However, this advantage is lost once a certain number of panorama renderings are desired. Furthermore the panorama simulation has a specular component, which is necessary to fully understand some daylighting scenarios.

The main drawback to the panorama method is its restriction on locomotion; the user can only look around the scene and move to predefined positions. Moreover, the inability to interact with the environment - such as colliding with walls and furniture - is believed to lead to less presence in a virtual simulation. Thus, the free roam method is appealing for a case where the there is either little specularity in the environment or it is not important to the user.

Conducting the outlined user study would give further understanding on the effectiveness of these two methods. It may be the case that my hypothesis that the free roam simulation grants significantly greater presence is false. This result doesn't entail the method isn't useful at all, but it does imply that in its current form the free roam simulation isn't overwhelmingly preferable. In the final subsection I will outline ways to improve the free roam method.

4.2 Further development

Further work on the free roam method has the potential to yield promising results. I note two obvious avenues for improvement. The first improvement is related to making the software pipeline more accessible to other 3D files. The second improvement is more substantial, but could yield a greater sense of presence for users.

First, the Python script is limited to geometry and material definitions from RAD files.

Specifically, it was built for RAD files generated with the Rhino software. Thus, there is room to make the implementation more compatible with other geometry file types.

Second, the appeal of the free roam approach would be drastically increased if specularity were added. Now, I maintain that the HDR textures for each geometric face should still only have the diffuse component. The specular component would need to be rendered in real time, because it is practically impossible to generate a separate texture with the specular component for each position that the user goes. While the real time rendered specular component may not be as accurate as a Radiance rendering, it could still (1) enhance user presence and (2) give a sense glare in the environment.

Though there does not currently exist a validated ray tracer for daylighting that can run in real time, I find it reasonable to assert that such a thing could be created in the near future given recent advances in real time ray tracing. I hold that the two methods outlined here would likely still have some use for less powerful machines.

REFERENCES

[1] L. Beltran and D. Liu, "Evaluation of dynamic daylight metrics, based on weather, location, orientation and daylight availability," *35th Passive and Low Energy Architecture (PLEA) Conference*, 2020.

[2] G. Ward, "The radiance lighting simulation and rendering system," (Florida, USA), 1994.

[3] D. DiLaura, R. Mistrick, K. Houser, and G. Steffy, *The lighting handbook: reference applications*. New York, USA: IES, 2011.

[4] E. Brown and P. Cairns, "A grounded investigation of game immersion," in *CHI '04 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '04, (New York, NY, USA), p. 1297–1300, Association for Computing Machinery, 2004.

[5] S. Subramaniam, G. Reis, and S. Hoffman, "A tool for assessing visual comfort through an immersive environment," *35th Passive and Low Energy Architecture (PLEA) Conference*, 2020.

[6] D. Moult, "Create 360 vr panoramas with radiance," 2019. `https://thinkmoult.com/create-360-vr-panoramas-with-radiance.html`.

[7] Unity Technologies, *Unity game engine*, 2020. `https://unity.com`.

[8] Godot Engine contributors, *Godot Engine*, 2021. `https://godotengine.org`.

[9] T. Schubert, F. Friedmann, and H. Regenbrecht, "The experience of presence: Factor analytic insights," *Presence*, vol. 10, pp. 266–281, 2001.

[10] K. Chamilothori, J. Wienold, and M. Andersen, "Adequacy of immersive virtual reality for the perception of daylit spaces: Comparison of real and virtual environments.," *LEUKOS*, pp. 1–24, 2018.

APPENDIX: A

The following repository contains Python code developed to generate the parallel projection views from the RAD file - https://github.com/LBess/DaylightingScripts

APPENDIX: B

A portion of the *rpict* manual page from the Radiance [2]. The *rpict* program generates a picture given the scene and view specifications.

-dp D: Set the secondary source presampling density to D. This is the number of samples per steradian that will be used to determine ahead of time whether or not it is worth following shadow rays through all the reflections and/or transmissions associated with a secondary source path. A value of 0 means that the full secondary source path will always be tested for shadows if it is tested at all.

-ar res: Set the ambient resolution to res. This number will determine the maximum density of ambient values used in interpolation. Error will start to increase on surfaces spaced closer than the scene size divided by the ambient resolution. The maximum ambient value density is the scene size times the ambient accuracy (see the -aa option below) divided by the ambient resolution. The scene size can be determined using getinfo(1) with the -d option on the input octree. A value of zero is interpreted as unlimited resolution.

-ms sampdist: Set the medium sampling distance to sampdist, in world coordinate units. During source scattering, this will be the average distance between adjacent samples. A value of 0 means that only one sample will be taken per light source within a given scattering volume.

-ds frac: Set the direct sampling ratio to frac. A light source will be subdivided until the width of each sample area divided by the distance to the illuminated point is below this ratio. This assures accuracy in regions close to large area sources at a slight computational expense. A value of zero turns source subdivision off, sending at most one shadow ray to each light source.

-dt frac: Set the direct threshold to frac. Shadow testing will stop when the potential contribution of at least the next and at most all remaining light source samples is less than this fraction of the accumulated value. (See the -dc option below.) The remaining light source contributions are approximated statistically. A value of zero means that all light source samples will be tested for shadow.

-dc frac: Set the direct certainty to frac. A value of one guarantees that the absolute accuracy of the direct calculation will be equal to or better than that given in the -dt specification. A value of zero only insures that all shadow lines resulting in a contrast change greater than the -dt specification will be calculated.

-dr N: Set the number of relays for secondary sources to N. A value of 0 means that secondary sources will be ignored. A value of 1 means that sources will be made into first generation secondary sources; a value of 2 means that first generation secondary sources will also be made into second generation secondary sources, and so on.

-ss samp: Set the specular sampling to samp.For values less than 1, this is the degree to which the high-lights are sampled for rough specular materials. A value greater than one causes multiple ray samples to be sent to reduce noise at a commensurate cost. A value of zero means that no jittering will take place, and all reflections will appear sharp even when they should be diffuse.This may be desirable when used in combination with image sampling (see ps option above) to obtain faster renderings.

-st frac: Set the specular sampling threshold to frac. This is the minimum fraction of reflection or transmission, under which no specular sampling is performed. A value of zero means that highlights will always be sampled by tracing reflected or transmitted rays. A value of one means

that specular sampling is never used. Highlights from light sources will always be correct, but reflections from other surfaces will be approximated using an ambient value. A sampling threshold between zero and one offers a compromise between image accuracy and rendering time.

-ab N: Set the number of ambient bounces to N. This is the maximum number of diffuse bounces computed by the indirect calculation. A value of zero implies no indirect calculation.

-aa acc: Set the ambient accuracy to acc. This value will approximately equal the error from indirect illuminance interpolation. A value of zero implies no interpolation.

-ad N: Set the number of ambient divisions to N. The error in the Monte Carlo calculation of indirect illuminance will be inversely proportional to the square root of this number. A value of zero implies no indirect calculation.

-as N: Set the number of ambient super-samples to N. Super-samples are applied only to the ambient divisions which show a significant change.

-lr N: Limit reflections to a maximum of N, if N is a positive integer. If N is zero, then Russian roulette is used for ray termination, and the -lw setting (below) must be positive. If N is a negative integer, then this sets the upper limit of reflections past which Russian roulette will be used. In scenes with dielectrics and total internal reflection, a setting of 0 (no limit) may cause a stack overflow.

-lw frac: Limit the weight of each ray to a minimum of frac. During ray-tracing, a record is kept of the estimated contribution (weight) a ray would have in the image. If this weight is less than the specified minimum and the -lr setting (above) is positive, the ray is not traced. Otherwise, Russian roulette is used to continue rays with a probability equal to the ray weight divided by the

given frac.

-ps size: Set the pixel sample spacing to the integer size. This specifies the sample spacing (in pixels) for adaptive subdivision on the image plane.

-pt frac: Set the pixel sample tolerance to frac. If two samples differ by more than this amount, a third sample is taken between them.