university of groningen

faculty of science and engineering

RASTER TEXTURES IN VIRTUAL RAY TRACER

Bachelor's project, Computing Science

Victor Gaya Walters
s3857123
University of Groningen

July 19, 2023

SUPERVISED BY:

Jiří Kosinka
Scientific Visualization and Computer Graphics
University of Groningen

Steffen Frey
Scientific Visualization and Computer Graphics
University of Groningen

RASTER TEXTURES IN VIRTUAL RAY TRACER

Abstract

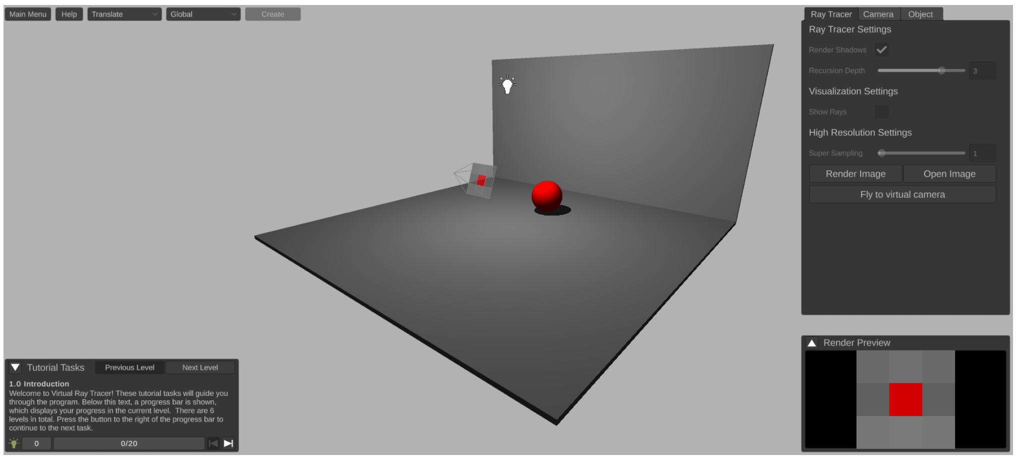
Many concepts and algorithms in Computer Graphics benefit from being visualised when trying to understand them. Virtual Ray Tracer (VRT) was developed to educate students on concepts related to ray tracing. The application explains concepts using textual explanation and visualisation techniques on objects. Different levels are created to elaborate on different topics. Further development in VRT intends to extend the application with new topics. This (bachelor) project aims to extend VRT with concepts related to texturing. This project focuses on raster textures and is done in close collaboration with Tom Couperus, whose concurrent bachelor project focuses on procedural textures. Four additional levels were created to cover 3D objects, texture mapping, texture sampling and procedural textures. A user study with twelve participants was conducted to evaluate the educational usefulness and the overall user experience of the new additions. The results indicate that users gained knowledge of texturing and had good experiences using VRT.

Contents

1. Introduction

To render and display 2D and 3D scenes, a computer executes complicated processes on both the hardware side and in its underlying software. All processes work together to render complicated patterns and colours to a screen. To get realistic-looking 3D objects with visually pleasing colour patterns, images can be mapped onto 3D objects. This process is called texturing. Different rendering techniques, like rasterisation and ray tracing, are used to render scenes with objects into 2D images, which can be displayed on a screen.

Texturing is one of the topics taught in the Computer Graphics course at the University of Groningen. The course uses conventional lecturing elements, like slides with images, to illustrate topics. However, the course desired an external tool for visualisation purposes. This initiated projects to develop Virtual Ray Tracer (VRT), an application that aims to visualise three-dimensional ray tracing. The target demographic for VRT is students following the Computer Graphics course. However, the application is built to be accessible to a broad and general audience. Currently, VRT comprises fifteen levels, teaching the user various concepts related to ray tracing. Figure 1 shows the first introduction level as an example of what the levels look like.



1.0 Introduction
Welcome to Virtual Ray Tracer! These tutorial tasks will guide you through the program. Below this text, a progress bar is shown, which displays your progress in the current level. There are 6 levels in total. Press the button to the right of the progress bar to continue to the next task.

Figure 1: Introduction level of Virtual Ray Tracer.

This project aims to extend VRT with concepts related to texturing. More specifically, this project goes into further detail about raster textures. Another concurrent project, by Tom Couperus [1], focuses on procedural textures. Due to the similarity of the topics, both projects collaborate closely on the implementation of shared components.

This project mostly focuses on the creation of the 3D Objects and the Texture Mapping levels, whereas the concurrent project focuses on the creation of the Texture Sampling and Procedural Textures levels. Since produced components were used throughout multiple levels and different unused visualisation techniques were tested, this thesis will still include information on all concepts. To read more about the implemented concepts, it is recommended to read the thesis on the concurrent project as well.

This thesis starts by going over related work and background theory in Chapter 2. Chapter 3 discusses the requirements set when starting the project. Chapter 4 explains what software is used and how the codebase is designed. It also enumerates the content produced during the project. The quality of the product is evaluated in a user study. The contents of the user study and the results are discussed in Chapter 5. The thesis is finalises with a conclusion in Chapter 6 and suggestions for future work in Chapter 7.

2. Related Work and Background

2.1. **Current state of VRT**

VRT is so far the accumulation of 6 different bachelor projects [2, 3, 4, 5, 6, 7] that have been combined and published in 2 iterations [8, 9]. Over 2 years VRT has taken steps in improving quality and has become a more than adequate teaching tool. VRT uses levels to explain different concepts, mostly related to ray tracing, but some levels cover interaction with the program. An advantage of using separate levels for different concepts is that new environments can be created in each level.

2.1.1. *VRT 1.0*

In the first version of VRT [8], the user is introduced to the application. The first steps in the program aim to inform the user on how to interact with the program and what it offers. This means the first levels of VRT cover controls and functionality before explaining ray tracing concepts.

These concepts are explained in levels, where VRT 1.0 uses a standard layout for each of the levels. The levels have a plain grey background with some interactable User Interface (UI). In the centre of the screen, there is room to place objects. With this centre focus, different visualisations can be shown using these objects.

To make the application more user-friendly, keyboard shortcuts for the basic interactable components are introduced. This allows for faster interaction once the user is more familiar with these shortcuts.

2.1.2. *VRT 2.0*

The second version of VRT brings new additions in several different areas of the program [9]. VRT 2.0 includes more levels explaining some more computer graphics concepts. Distributed ray tracing, super-sampling, soft shadows, axis-aligned bounding boxes and octrees are among some of the new topics. Explaining these topics required the addition of several components to the program, like acceleration structures. Other components that were added to illustrate the concepts better were: area- and spotlights, better ray visuals, and new UI elements and settings. Furthermore, VRT 2.0 had a big focus on the gamification of the application. The goal of gamification is to make the user feel rewarded after completing tasks. Tutorials were introduced to make explanations more understandable and to show visuals while the user can read the text. A system to earn points and unlock levels and objects gives the user an incentive to unlock desirables.

2.2. **Theory**

2.2.1. *3D Objects*

To understand some concepts behind textures, it is important to understand how a computer interprets objects. The flat nature of polygons makes it easier for a computer to perform calculations on the polygons in the rendering process. To make a shape out of polygons, 3D objects are constructed out of a (large) number of 3D coordinates called vertices. This information, together with information on which vertices are connected to form polygons, can be stored in various 3D file formats, each having its pros and cons [10]. Polygons as elements of a 3D object are called faces and their outlines are called edges.

2.2.2. *Textures*

To get more detail in the colour of an object, images can be used to retrieve specific colours for corresponding locations on the object. Images used on objects in computer graphics are called textures [11]. These images can either be manually produced or generated procedurally. 3D vertex coordinates can be linked to 2D texture coordinates to retrieve certain information stored in the textures. This process is called texture mapping, which is further explained in Section 2.2.3. Commonly, information on rgb-values and optionally an alpha value is stored for each pixel. This requires larger pixel formats, but 1-bit formats can also be used to colour a pixel either black or white, or an 8-bit format for grey-scaled images. These grey-scaled images are more commonly used to map their values to a different property than colour [12, p. 71–72].

2.2.3. *Texture Mapping*

Texture mapping is a technique used in computer graphics to bring more detail to 3D objects by using information stored in 2D images, also called texture maps. 3D coordinates of an object are mapped to 2D coordinates, sometimes referred to as uv-coordinates, on images. Because it visually looks like this mapping wraps an image onto an object as illustrated in Figure 2, this is sometimes referred to as "uv-wrapping". There are two ways of mapping: manually and mathematically. When creating 3D objects, some file formats also optionally store texture information. An example of this is the .obj file format published by Wavefront Technologies [13], which includes texture coordinates for every vertex coordinate stored in the file. Alternatively, for more conventional shapes, xyz-coordinates can be converted to texture coordinates (often called uv-coordinates) mathematically. A sphere, for example, could use the following function to map its coordinates [12, p. 261]:

$$\varphi(x, y, z) = ([\pi + \text{atan2(x,y)}] \,/\, 2\pi, [\pi - \text{acos}(z \,/\, \|x\|)] \,/\, \pi).$$

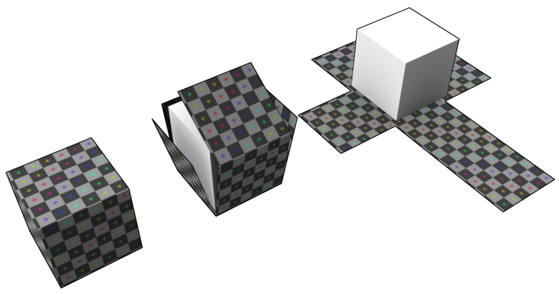


Figure 2: The way faces of an object project on textures is sometimes called "uv-wrapping". Image adopted from [14]

2.2.4. *Texture Sampling*

When a final image needs to be drawn to the screen, the colour found for this pixel might not land exactly in the centre of a texture pixel, also called texels, after a texture lookup. The next step is to determine a method to find the colour that results in the most realistic image. The method to determine the colour is called texture sampling. The easiest way to find the colour is to take the colour from the nearest texel centre, which is called Nearest Neighbor Sampling [15]. Another way to determine the colour is to interpolate between the closest values on the x-axis and after, this step is repeated for the y-axis. This interpolation technique is called bilinear interpolation [15]. In Figure 3 this process is illustrated by colouring the weight of each of the points used in the interpolation step.

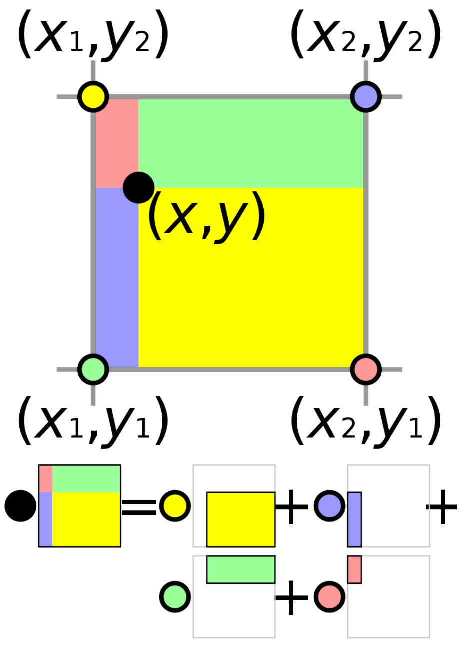


Figure 3: To determine the resulting colour, 4 points are used in the interpolation step, using relative distance as weights. Image adopted from [16]

3. Requirements

Since this project builds on the work of other projects in the form of an application, there should be a notable emphasis on setting requirements. There are different perspectives in which requirements are set to meet the standards for VRT. The first set of requirements focuses on the quality of the product. VRT is used as a learning tool, which means a certain standard of quality has to be met so that the target audience can be satisfied with their experience. Besides quality, the second set of requirements focuses also on the quantity of teaching material and other implemented content. Introducing a new topic to VRT opens the opportunity to explain a variety of new concepts related to this topic. Since this project is done in close collaboration, it is important to set workflow requirements to speed up development and avoid working on the same implementations.

3.1. Quality

VRT currently uses 14 levels to explain how VRT works and the different elements of raytracing. Each of these levels sufficiently illustrates the respective concepts they are explaining. This also needs to hold true for this project: Everything explained needs to have a level of detail so that the target audience will understand the concepts after completing the tutorial. Requirements for the quality of the resulting product are:

1. Textual explanations need to be concise, but complete.
2. Visualisation requires a sufficient level of clarity and precision.

3.2. Implementation

VRT teaches different concepts of computer graphics using visualisations by modifying objects. This project should follow this style by implementing new components and building new levels. To make sure the new implementations have sufficient quality and quantity, the following requirements are defined:

1. This project should introduce enough content to give the user a sufficient understanding of texturing.
2. New topics should be explained in approximately 4 levels.
3. Additional code written should easily be extensible.

3.3. Workflow

The main focus of this project is on raster textures and how images can be used as textures. The concurrent project focuses on implementing procedural textures. Since both topics require similar components, workflow for the combined effort of producing these components necessitates requirements. To make sure work is completed efficiently, the following requirements for work distribution and workflow are set:

1. Determine beforehand what components are used in both projects and divide implementations.
2. Develop components so that the concurrent project can use them as well.

4. Implementation

4.1. **Tools**

To create a virtual world where motion of objects and other behaviours is visualised, a substantial amount of underlying code is needed to handle input, audio, visuals, physics and plenty more tasks. Many game engines can provide these functionalities, some of them being free to use. There are widely used free and open-source game engines, like Godot, but in the case of VRT, Unity was chosen to develop the application in. Unity uses the C# programming language and has a helpful editor to more easily utilise some of its functionalities.

4.2. **Infrastructure**

VRT is built on two main components, the Unity application and the ray tracer [9]. A diagram depicting the infrastructure used in VRT is shown in Figure 4 below. All communication with the scene is handled by the `Scene Manager`. By using this system, the scene manager acts as a single entity with references to all objects. Whenever an object is modified, an event is passed on to the scene manager. This way the scene manager can ensure objects in the scene are only updated when they actually change. When needed, the scene manager can pass the scene data to the ray tracer scene, where the ray tracer can consequently use this information to create rays. The `Ray Manager` then uses these rays to visualise rays in the scene or create a ray traced image.

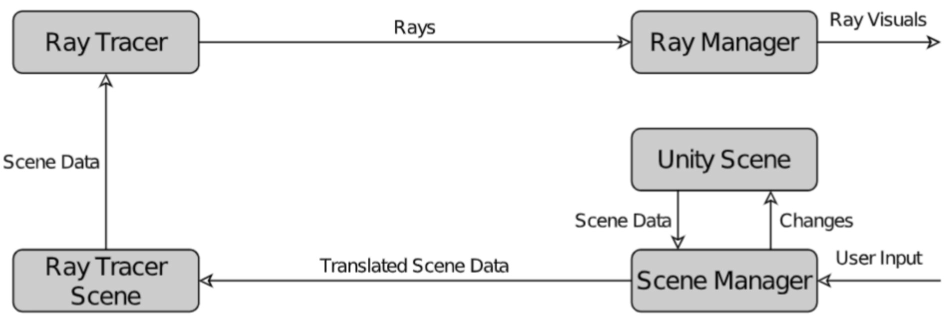


Figure 4: The design of the general application structure. The infrastructure design from this project, displayed in Figure 5, modifies the Unity Scene and Scene Manager components from this design. Image adopted from [9].

Since one of the requirements, as mentioned before in Section 3.2, is to keep the application extensible, this project aims to build upon this design. To implement texturing, there is no need to modify the ray tracer. Textures only modify the scene objects and therefore only changes to the Unity scene need to be made, somewhat similar to the scene manager. Instead of using the scene manager, an event and behaviour system is used to make changes to the scene. Events can trigger scripts and behaviours that can in turn make changes to the object in the scene. A graph of the described design infrastructure is shown in Figure 5.

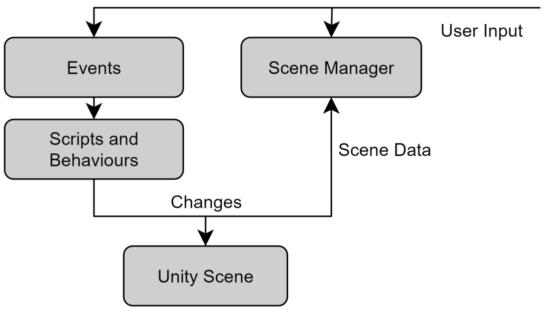


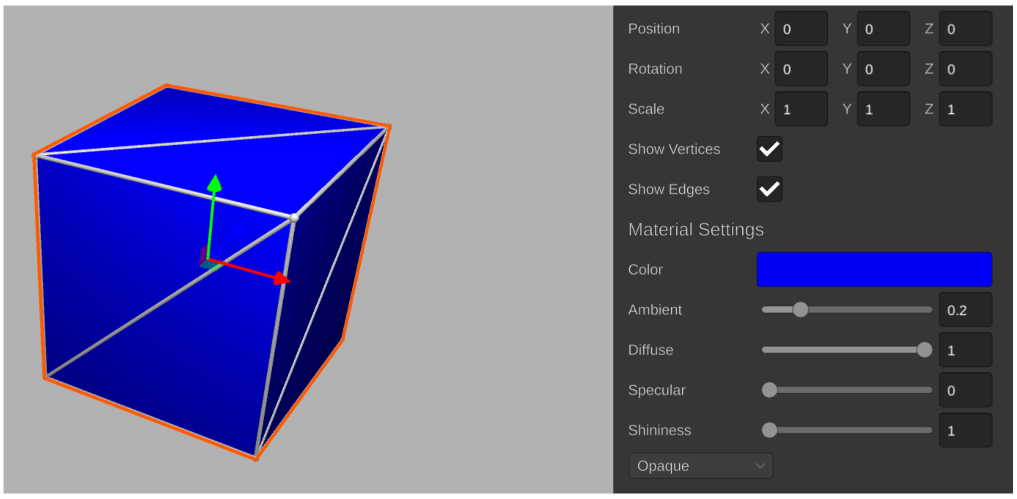
Figure 5: The design of the general application structure.

When creating a new Unity scene, a blank level with a grey background and UI is used, where the developer can start to add components. An important component of the UI is the tutorial text component at the bottom left of the screen. In this textbox, explanations are written about what is shown on screen. Everything else in the UI can be disabled so that the user can focus on the learning objective.

4.3. New levels

4.3.1. *3D Objects*

Before elaborating on textures, an explanation of objects is given, as textures are placed on objects. The user is shown that objects are built up out of vertices and edges, and connecting these creates faces. Faces create polygons, which, because of their flat and elementary structure, are well suited to use in the calculations required in the graphics pipeline. The UI shows 2 buttons to enable and disable the vertices and edges as illustrated in Figure 6.



Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Show Vertices ✓
Show Edges ✓
Material Settings
Color
Ambient 0.2
Diffuse 1
Specular 0
Shininess 1
Opaque

Figure 6: 3D objects level. The user can toggle vertices and edges with checkboxes.

4.3.2. *Texture Mapping*

Now that the user is aware that objects have vertices in 3D space, texture mapping can be explained. This level starts by textually explaining what texture coordinates (also called UV coordinates) are and how 3D coordinates can be mapped to 2D coordinates. The main visualisation of this level shows a transition from these 3D coordinates to the texture coordinates projected next to the object. The user is given a slider to manually transition and a button to set the transition to a loop. Figure 7 shows 3 stages of the transition.

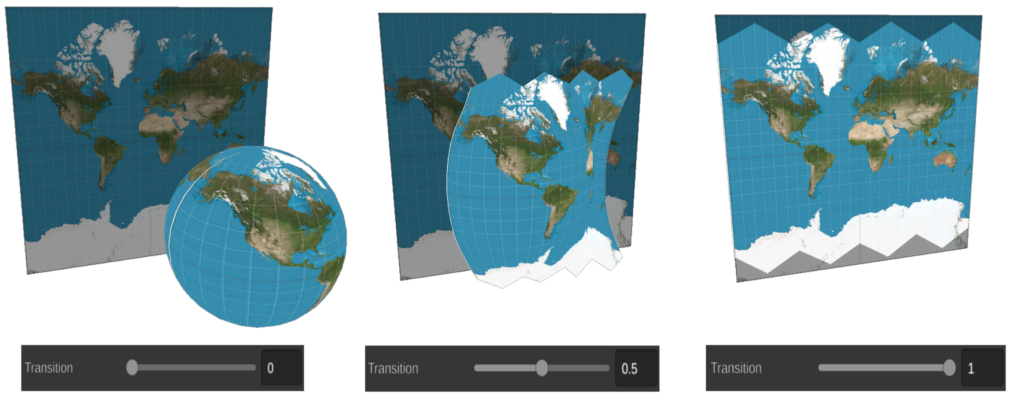


Figure 7: Visualisation of three of the stages of the transition between 3D coordinates and texture coordinates.

4.3.3. *Texture Sampling*

The objective of the Texture Sampling level is to educate the user about interpolation techniques. The level demonstrates Nearest Neighbor Sampling and Bilinear Interpolation. The level makes use of the `Render Preview` component to show the result of interpolating with each of these techniques over a specific point of the texture. Figure 8 below shows the component used to illustrate the different steps taken in bilinear sampling. This way the user can create their own example for the explanation in the tutorial and try this using different textures.

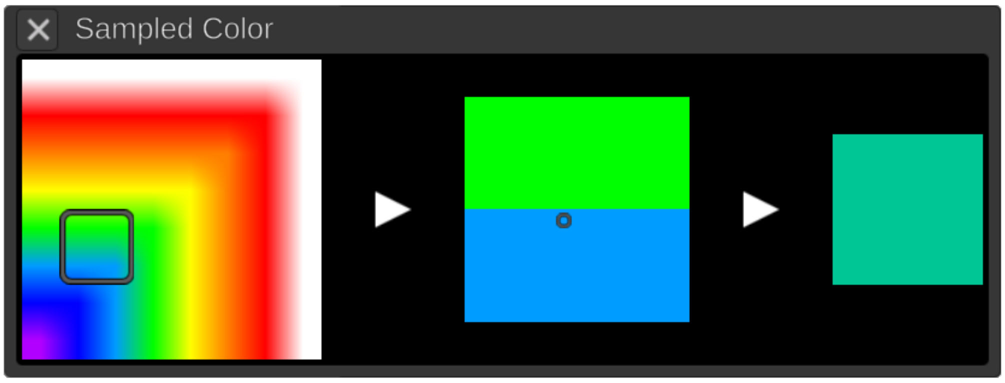
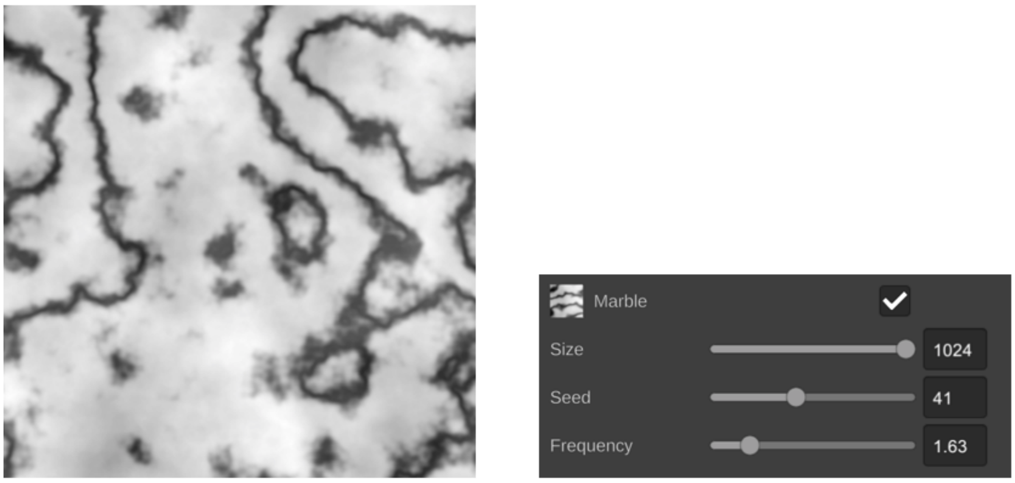


Figure 8: The render preview component is used to show steps taken in bilinear sampling. The first section shows the area of the texture used in the calculation. The second section shows where the cursor lands in this area. The last section shows the resulting colour used to colour the part of the object under the cursor.

4.3.4. *Procedural Textures*

So far the user has only worked with raster images as textures, but textures can be procedurally generated as well. In this level, the user is presented with information about generating images with mathematical functions. Two procedurally generated textures were added to the assets of the project to be shown in this level.


Marble ✓

Size 1024

Seed 41

Frequency 1.63

Figure 9: Procedurally generated marble texture with UI to change its properties.

4.4. **New components**

This project together with the collaboration project on procedural textures brought a number of new components to VRT. The main changes include:

1. **Scripts:** As was mentioned before in Section 4.2, the main way to make changes to the objects in the Unity scene, is by adding behaviours to objects. These behaviours are components of objects that can be enabled and disabled [17]. This allows for more control over interactions in the levels. Scripts were added to create components at runtime, like the vertices and edges in the 3D Objects level. Scripts are used to interact with the UI after certain events are triggered. Scripts are used to modify the properties of objects, for example when textures are added to objects.
2. **Objects:** Certain visualisations in the levels require custom components to be made. To display textures, a standard cube from Unity's standard object is sufficient to place the textures on. However, the Texture Mapping level requires objects with custom texture coordinates to show contrast when faces do not completely cover an image as shown in Figure 10.

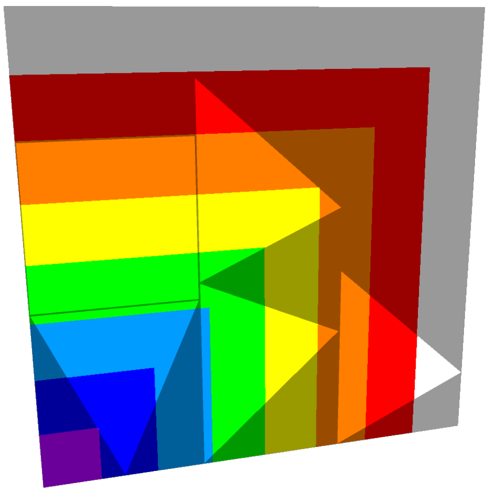


Figure 10: Only a small part of the texture is used on the object.

3. **Materials and Shaders:** VRT uses a custom `Ray Tracer Shader` to use on objects that implement the `RTMesh` component. To make textures work on the objects in the scene, this custom shader, therefore, is modified by passing texture assets to the shader script. To make custom illustrations, new shaders and Unity's shader graphs are written for objects that do not implement `RTMesh`.

Materials apply shaders to specific objects. New materials were created for an easier workflow.

1. **Events:** Scripts in Unity are not exactly like traditional programs that run until they complete their task. Instead, control is passed to scripts by calling functions that are declared within it [18]. In VRT this is similar to the task of the `Scene Manager`, except with events, Unity activates the called functions. For texturing, new events were created to make the levels interactive.

5. **UI:** Because texturing can be considered more of a standalone topic outside of ray tracing, it is fitting to have its UI partially separated from the other ray tracing functionality. The `Object` tab in the UI has a button to open a newly added `Texture` tab. This tab gives the user the ability to modify both raster textures and procedural textures when they are available on an object. Figure 11 shows what this tab looks like in the Texture Mapping level.

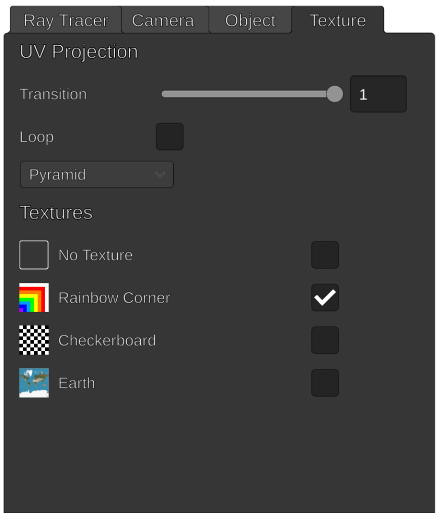


Figure 11: New Texture tab with interactive components for modifying textures.

5. User study

The aim of VRT is to educate users on a variety of topics in computer graphics. This requires visualisations and explanations that are well understood by a general audience. Since the quality of visualisations and user experience are subjective in nature, a user study was chosen to analyze the quality of the additional content. In the user study, the user is asked to complete the new levels and answer two questions beforehand and ten more questions afterwards. After answering all questions, the user can optionally leave general remarks in a textbox for feedback. To get the user familiar with the program, the user is tasked to complete an additional two levels about controls and ray tracing basics before moving onto the new levels. A table with all questions and options for answers is listed in the Appendix.

5.1. **Questions in the user study**

To evaluate the quality of the created product, the user study analyses three different topics: Prior experience, Educational usefulness and User experience. Asking questions about these categories should provide enough insight into the user's prior and gained experience with texturing.

5.1.1. *Prior Experience*

To get a good grasp of the user's prior experience, two questions were asked before the user started using the application. This information provides insight into the knowledge the user already possesses. The two questions the user answers are:

- How well would you say you understand textures?
- Did you follow the RUG Computer Graphics course?

5.1.2. *Educational usefulness*

To evaluate whether the user has learned from using the application, six questions are drawn up that evaluate general educational usefulness and some questions to evaluate specific additions. The first three questions asked in this part are:

- Did the application help you understand textures (better)?
- Do you think the application can help others understand textures?
- Do you think the application can help future students of the RUG Computer Graphics course understand textures?

Then the question "What aspects were most helpful to you in better understanding textures?" is asked about the following three components:

- The tutorial texts
- The visualisation
- The ability to experiment with various settings in the menu on the right

5.1.3. *User experience*

Finally, to get a better understanding of how well the format of the application works for teaching computer graphics topics, two questions about how the user experienced using the application were asked:

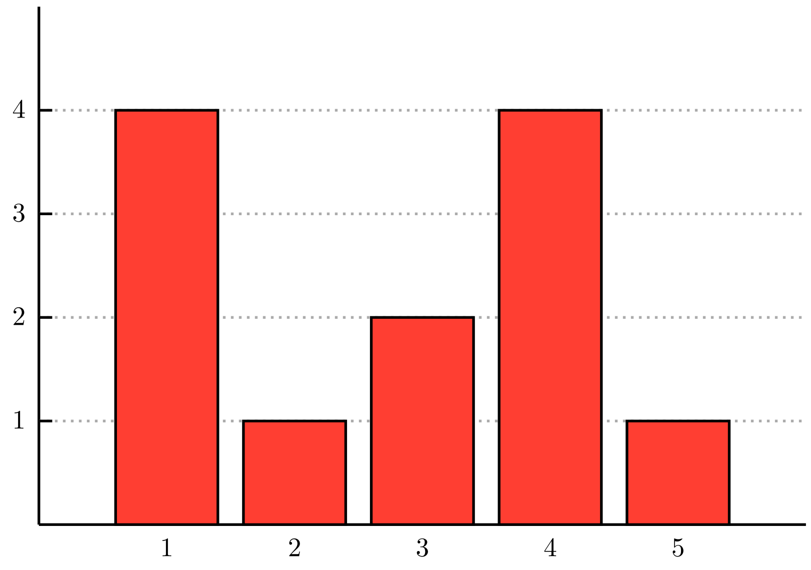
- How would you rate the usability of the application?
- How complex did you find the application?

5.2. **Results**

The answers to the questions were submitted via a Google Forms document. Eight of the questions were evaluated on a scale from 1 to 5 with lower numbers indicating an answer in line with a negative response or answering "no" and high numbers indicating the opposite. One "yes" or "no" question was included and one question with a scale from 1 to 3.

5.2.1. *Prior experience*

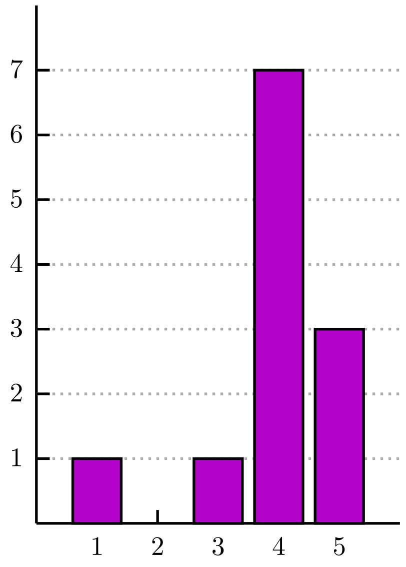
The first part of the questions shows the familiarity of the user with texturing and computer graphics as a whole. Out of the twelve people that submitted the survey, three answered to have followed the course Computer Graphics. Graph 1 shows that familiarity with texturing is very distributed among the users.



Graph 1: How well would you say you understand textures?

5.2.2. *Educational usefulness*

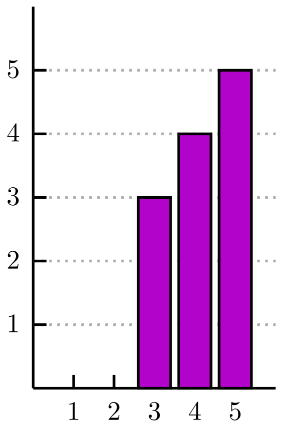
After using VRT users were first asked to answer some questions about the quality of the teaching material. When comparing what was seen before in Graph 1 with the results of Graph 2, it is apparent that the evaluated score went up significantly. The only outlier is the single evaluation of 1, but when taking a closer look at the individual evaluation forms, it became apparent that this was chosen by someone who scored 5 in Graph 1. It seems reasonable to assume that the user already sufficiently understood the subject matter.



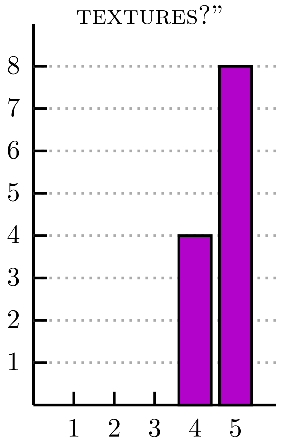
Graph 2: Did the application help you understand textures (better)?

To evaluate specific parts of the program, users were asked how much some of these components helped them to understand texturing. The Graphs below show that most users evaluated the quality of the different components between 3 and 5. The tutorial texts score lowest and the visualisation highest as can be seen in Graph 3 and Graph 4.

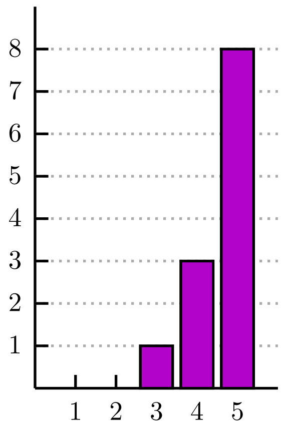
"WHAT ASPECTS WERE MOST HELPFUL TO YOU IN BETTER UNDERSTANDING TEXTURES?"



Graph 3: The tutorial texts



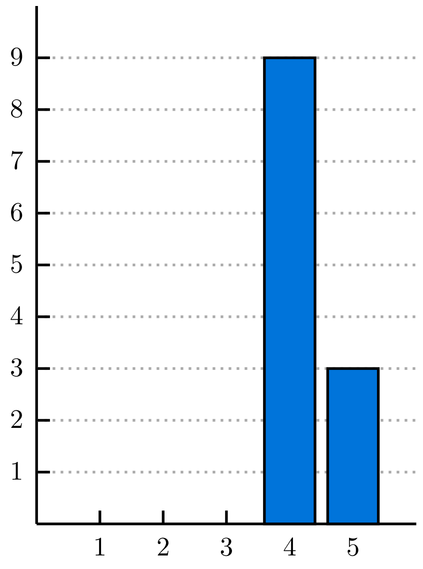
Graph 4: The visualisation



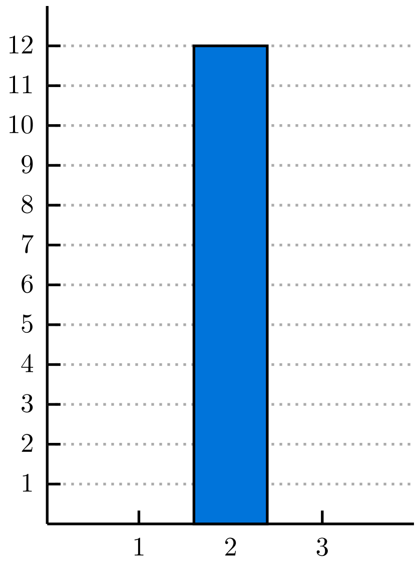
Graph 5: The ability to experiment with various settings in the menu on the right

5.2.3. *User experience*

To conclude the evaluation, the user was asked about the usability and the complexity of the application. The results are shown in Graph 6 and Graph 7. The application is overall well received with a good score for usability. All participants score the complexity a two out of three.



Graph 6: How would you rate the usability of the application?



Graph 7: How complex did you find the application?

5.2.4. *Feedback*

In total, four participants left feedback after answering the questions. The feedback included some remarks with possible improvements. Some useful information concluded from the remarks are:

- Some users who were not experienced with similar programs experienced trouble understanding objectives.
- Some users did not understand all technical terms used.
- One user commented on textual explanations being hard to read.

Some solutions to these problems might be:

- Adding a feature to highlight objective elements.
- Adding a technical terms list.
- Adding a setting to change the font size.

6. Conclusion

The goal of this project was to introduce various concepts behind texturing to this project. VRT aims to explain topics with insightful visuals and understandable textual explanations. Newly added content should teach the user enough about texturing to develop a general understanding of the basics of texturing. This project was made in close collaboration with a second concurrent project, where both projects focussed on explaining texturing, but differed in the type of textures. This project focussed on raster textures, whereas the concurrent project focussed on procedurally generated textures. Since both projects have similar topics, many components were developed in collaboration.

In total four levels were developed. The levels explain 3D objects, texture mapping, texture sampling and procedural textures. To complete all functionality in the levels, new components were added to the project. New scripts were added for new interactions and infrastructure was designed using a behaviour and event system for internal communication. The asset collection in VRT was expanded with new objects, materials, shaders and UI components.

To evaluate the quality of the new additions to VRT, a user study was conducted. The user study assessed the prior experience of the users, the experienced educational usefulness and the user experience. The results were positive, with both practised and novice users benefitting from the program. In the open feedback, users were able to make other remarks they had on VRT. Remarks mentioned included confusing objectives, being stuck on tasks and small tutorial texts.

Overall this project successfully produced useful content to merge into the VRT project. The application now sufficiently educates in texturing, with users acquiring general knowledge about textures with its new learning material.

7. Future Work

VRT is starting to get comprehensive in its lecturing material, but there is still a lot of room for more topics to cover and improvements to make. Some concepts that could be illustrated well in VRT are:

- **Path tracing:** Ever since NVIDIA launched their RTX 20-series lineup of GPUs with dedicated ray tracing cores, many game studios started implementing ray tracing as a feature in their games. Over the last year or so, another form of ray tracing, called path tracing, is being implemented more frequently in games. Path tracing uses the same methods as ray tracing, except path tracing does not follow rays throughout the entire scene, but only traces the most likely path to light sources [19]. Path tracing algorithms use Monte Carlo algorithms to achieve this. Only following rays most likely to hit a light source drastically improves performance, while resulting in very similar images.
- **Curves:** Since VRT was originally designed as a tool to teach the course Computer Graphics, the suggestion of adding curves as a topic makes sense since this is also taught in the course. Different levels could be implemented for polynomial curves, Bézier curves and Continuity just to name some underlying concepts.
- **Colour:** Now that VRT has more support to show different colours and complex colour patterns, tackling colours might be a practical follow-up concept. Colour is also a significant element in the Computer Graphics course. VRT might work very well to illustrate concepts like chromaticity, colour Spaces and colour perception.
- **Dispersion:** In ray tracing light is being portait as rays, where they are mathematically considered linear. But in reality, light is a form of electromagnetic radiation, what mathematically should be considered a wave function. When using wave functions instead of linear rays it is possible to illustrate dispersion of light when for example a ray hits transparent objects. Wave-optical rendering is a path tracing technique that uses generalised rays instead of classical linear rays [20].
- **Extending texturing:** When deciding what topics to in this project, some more topics were originally considered, but ended up being left out. In a following iteration, texturing could be extended to include concepts like mipmapping, aliasing and up- and downsampling as well.
- **Internal additions:** With some of the feedback from the user study, some ideas for internal improvements of VRT came up. Users sometimes had trouble finding out what to do for some of the tutorial tasks. Implementing a component that highlights objectives could be one way to solve this problem. For some presumably older users, some of the text was very small to read. Being able to adjust font size could be considered to add to the settings.

Acknowledgement

I would to make special thanks to my supervisors for assisting in the project by providing feedback and having meetings to discuss progress. I would also like to thank Tom Couperus for the close collaboration at various points in this project. Credit is also due to the 6 bachelor projects that developed VRT thus far [2, 3, 4, 5, 6, 7]. Finally, I would like to thank everyone who participated in the user study.

References

[1] T. Couperus, “Procedural textures in virtual ray tracer,” Bachelor thesis, University of Groningen, 2023.

[2] W. V. de la Houssaije, “A virtual ray tracer,” Bachelor thesis, University of Groningen, 2021.

[3] P. J. Blok, “Gamification of virtual ray tracer,” Bachelor thesis, University of Groningen, 2022.

[4] B. Yilmaz, “Acceleration data structures for virtual ray tracer,” Bachelor thesis, University of Groningen, 2022.

[5] R. Rosema, “Adapting virtual ray tracer to a web and mobile application,” Bachelor thesis, University of Groningen, 2022.

[6] J. van der Zwaag, “Virtual ray tracer: distribution ray tracing,” Bachelor thesis, University of Groningen, 2022.

[7] C. van Wezel, “A virtual ray tracer,” Bachelor thesis, University of Groningen, 2022.

[8] W. A. Verschoore de la Houssaije, C. S. van Wezel, S. Frey, and J. Kosinka, “Virtual ray tracer,” *Eurographics 2022-Education Papers*, pp. 45–52, 2022, doi: 10.2312/eged.20221045.

[9] C. S. van Wezel, W. A. Verschoore de la Houssaije, S. Frey, and J. Kosinka, “Virtual ray tracer 2.0,” *Comput. & Graph.*, vol. 111, pp. 89–102, 2022, doi: 10.1016/j.cag.2023.01.005.

[10] R. Kelly, and D. Chakravorty, “The 10 most popular 3d file formats,” 2022. (https://all3dp.com/2/most-common-3d-file-formats-model/)

[11] R. Tunnel, J. Jaggo, and M. Luik, “Computer graphics,” University of Tartu, 2023. (https://cglearn.codelight.eu/pub/computer-graphics)

[12] S. Marschner, and P. Shirley, *Fundamentals of Computer Graphics 5th Ed.*, CRC Press, 2022.

[13] L. of Congress, “Wavefront obj file format,” 2020. (https://www.loc.gov/preservation/digital/formats/fdd/fdd000507.shtml)

[14] PNGWing, “Uv mapping texture mapping cube mapping 3d modeling, cube, texture, angle, 3d computer graphics png,” 2023. (https://www.pngwing.com/en/free-png-xoahf)

[15] S. Fadnavis, “Image interpolation techniques in digital image processing: an overview,” *Int. J. Eng. Res. Appl.*, vol. 4, no. 10, pp. 70–73, Oct. 2014.

[16] Cmglee, “Bilinear interpolation visualisation svg,” 2023. (https://commons.wikimedia.org/wiki/File:Bilinear_interpolation_visualisation.svg)

[17] U. Technologies, “Scripting api: behaviour,” 2023. (https://docs.unity3d.com/ScriptReference/Behaviour.html)

[18] U. Technologies, “Manual: event functions,” 2023. (https://docs.unity3d.com/Manual/EventFunctions.html)

[19] N. Everson, “Path tracing vs. ray tracing, explained,” 2022. (https://www.techspot.com/article/2485-path-tracing-vs-ray-tracing/)

[20] S. Steinberg, R. Ramamoorthi, et al., “A generalized ray formulation for wave-optics rendering,” 2023. (https://ssteinberg.xyz/2023/03/27/rtplt/)

Appendix

Number	**Question**	**Answer**
1	How well would you say you understand textures?	1-5, “Not at all” - “Perfectly”
2	Did you follow the RUG Computer Graphics course?	“yes” or “no”
3	Did the application help you understand textures (better)?	1-5, “Not at all” - “Perfectly”
4	Do you think the application can help others understand textures?	1-5, “Not at all” - “Perfectly”
5	Do you think the application can help future students of the RUG Computer Graphics course understand textures?	1-5, “Not at all” - “Perfectly”
6	What aspects were most helpful to you in better understanding textures?: The tutorial texts	1-5, “Not helpful at all” - “Very helpful”
7	What aspects were most helpful to you in better understanding textures?: The visualisation	1-5, “Not helpful at all” - “Very helpful”
8	What aspects were most helpful to you in better understanding textures?: The ability to experiment with various settings in the menu on the right	1-5, “Not helpful at all” - “Very helpful”
9	How would you rate the usability of the application?	1-5, “Difficult and confusing” - “User friendly”
10	How complex did you find the application?	1-3, “Too simple” - “Too many options”