RASTER TEXTURES IN VIRTUAL RAY TRACER

Victor Gaya Walters
s3857123
University of Groningen
v.l.g.gaya.walters@student.rug.nl

SUPERVISED BY:

Jiri Kosinka
Scientific Visualization and Computer Graphics
University of Groningen

Stephen Frey
Scientific Visualization and Computer Graphics
University of Groningen

Abstract.

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magnam aliquam quaerat voluptatem. Ut enim aeque doleamus animo, cum corpore dolemus, fieri tamen permagna accessio potest, si aliquod aeternum et infinitum impendere malum nobis opinemur. Quod idem licet transferre in voluptatem, ut.

Contents

1. Introduction

- paragraph introducing texturing

To render and display 2D and 3D scenes, your computer undergoes some complicated processes on both the hardware side as well as in its underlying software. All processes work together to get the complicated patterns and colours on your screen and make texturing possible.

- paragraph to explain what texturing is
- paragraph to tell RUG teaches CG and what option there are currently are

(Sebastian Laque's RT)

- paragraph on goals for the project

1.1. **Introduction texturing**

1.2. **General VRT description**

1.3. **Focus on visualisation**

1.4. **Collaboration project**

- Name the two topics

2. Related Work and Background

2.1. **Current state of VRT**

VRT is so far the accumulation of 6 different bachelor projects [1, 2, 3, 4, 5, 6] that have been combined and published in 2 iterations [7, 8]. Over 2 years VRT has taken steps in improved quality and has become a more than adequate teaching tool.

2.1.1. *VRT 1.0*

In the first version of VRT [7], the user is introduced to the application. Therefore it is important to make the user understand how to interact with the program and what it offers. This means the first levels of VRT cover some controls and functionality before explaining ray tracing concepts.

These concepts are explained in levels and VRT 1.0 does so using 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 on these objects.

To make the application more user-friendly, keyboard shortcuts for some of 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 [8]. 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 adding different components to the program as well, like acceleration structures. Other components added to illustrate the concepts better were area- and spotlights, better ray visuals, and new UI elements and settings. Furthermore, VRT 2.0 gave 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 a pursuit to unlock desirables.

2.2. **Theory**

2.2.1. *3D Objects*

To understand some concepts behind textures, it is important to understand how your computer sees these objects. Your computer, or more specifically your GPU, traverses through a couple of steps where in each step some calculations are performed on some data. Most of this data are different properties of an object's vertices, like location and associated colour. Understanding vertices and moreover edges and faces, are therefore important to understand textures as well.

2.2.2. *Textures*

To get more detail in the colour of an object, we can use images to retrieve specific colours for corresponding locations on the object. Images used on objects in computer graphics are called textures [9]. These images can either be manually produced or generated procedurally. Tom Couperus' project [10], the other half of this collaboration project, focuses on procedurally generated textures. 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. This information can be stored a number of different pixel formats depending on what information it stores [11, p. 71–72]. A renowned format is an information on rgb values and optionally an alpha value to create an image with colour. But greyscaled values in 1-bit, 8-bit or 16-bit formats are also common to map values to other properties than colour.

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 on 2D images. 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 [12], 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). A sphere, for example, could use the following function to map its coordinates [11, p. 261]:

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

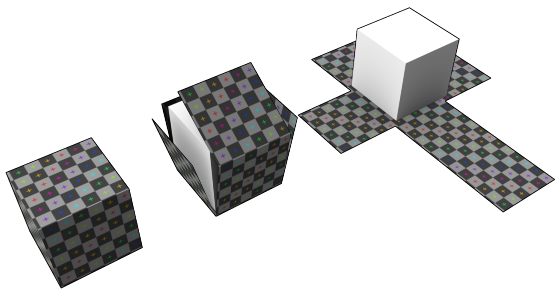


Figure 1: The way faces of an object project on textures is sometimes called "UV-wrapping" [13]

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. 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. In Figure 2 this process is illustrated by colouring the weight of each of the points used in the interpolation step.

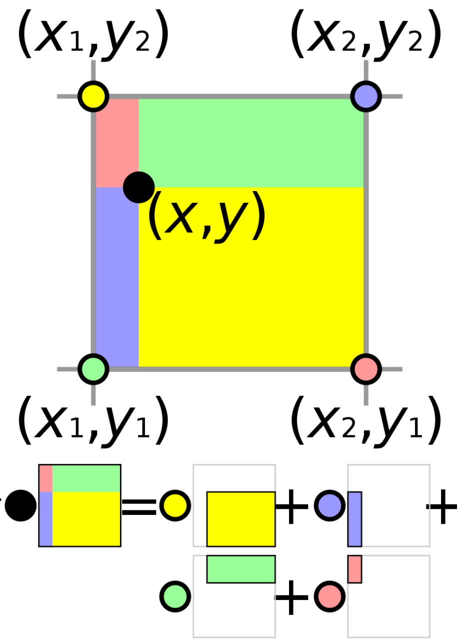


Figure 2: To determine the resulting colour, 4 points are used in the interpolation step, using relative distance as weights.

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 a few different perspectives in which we want to set requirements to meet the standards for VRT. We first want to set goals for 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, there should also be a minimum quantity of teaching material. Introducing a new topic to VRT opens the opportunity to explain a variety of new topics. 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. **Goals**

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. Some requirements in this regard are:

- Textual explanations need to be concise, but complete.
- Visualisation requires a certain 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 determined:

- This project should introduce enough content to give the user a broad understanding of texturing.
- Some of the underlying concepts that should be included besides the main topics are texture mapping and texture sampling. If there is time left, mipmapping could be looked at.
- The new topics should be explained in approximately 4 levels.

3.3. **Workflow**

The main focus of this project is on raster textures and how images can be used as textures. The other project of the collaboration that was mentioned before focuses on implementing procedural textures. To make sure work is completed efficiently, it is necessary to set certain requirements for work distribution and workflow:

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

4. Implementation

4.1. Tools

To create a virtual world where you can visualise the motion of objects and other behaviours, a lot of underlying code is needed to handle things like input, audio, visuals, physics and plenty more tasks. Luckily there are many game engines that can provide these functionalities, some of them being free to use. There are some 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.

paragraph on

paragraph on other tools used (Blender and GIMP) etc.

4.2. Infrastructure

VRT is built on two main components, the Unity application and the ray tracer [8]. All communication with the scene is handled by the `Scene Manager`. This way, the scene manager can act 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` can then use these rays to visualise rays in the scene or create a ray traced image. A diagram of this design is shown below in Figure 3.

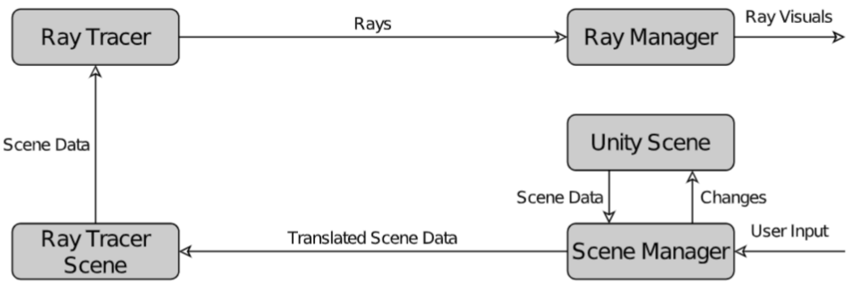


Figure 3: The design of the general application structure [8].

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, it is clear that there does not need to be any modification to 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 what is described is shown in Figure 4.

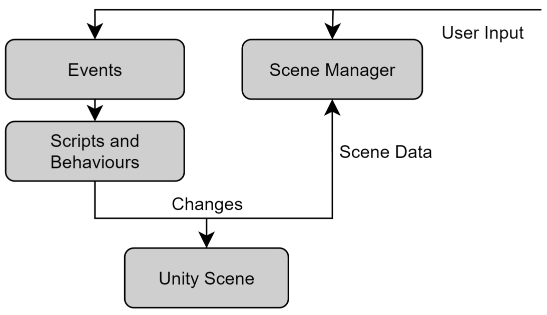


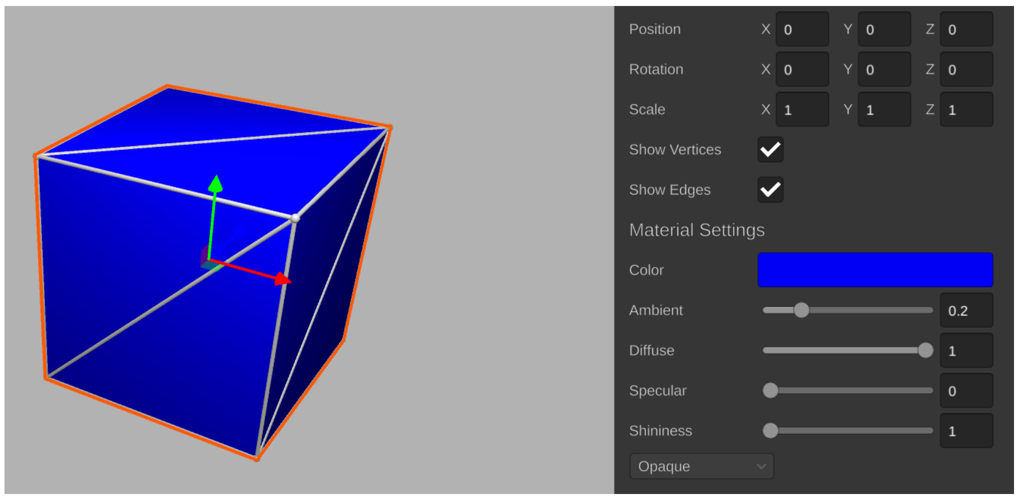
Figure 4: The design of the general application structure [8].

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 texts in the bottom left. In this textbox, explanations are written on 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 textures can be explained, it is important to understand what it is you place textures on. This level shows the user that objects are built up out of vertices and edges, and connecting these creates faces. These faces create polygons, which are easiest for a computer to use in the calculations required in the graphics pipeline. The UI shows 2 buttons to enable and disable the vertices and edges.



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 5: 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 texturally 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 6 shows 3 stages of the transition.

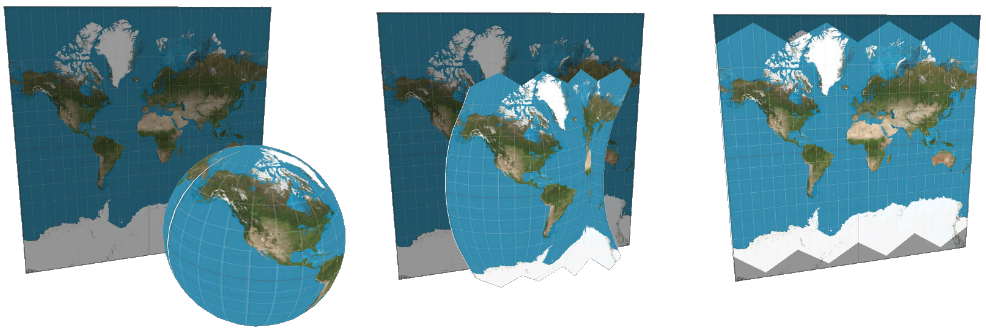


Figure 6: Visualisation of three of the stages of the transition between 3D coordinates and texture coordinates.

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 7 below shows what this looks like in VRT. This way the user can create their own example for the explanation in the tutorial and try this using different textures.

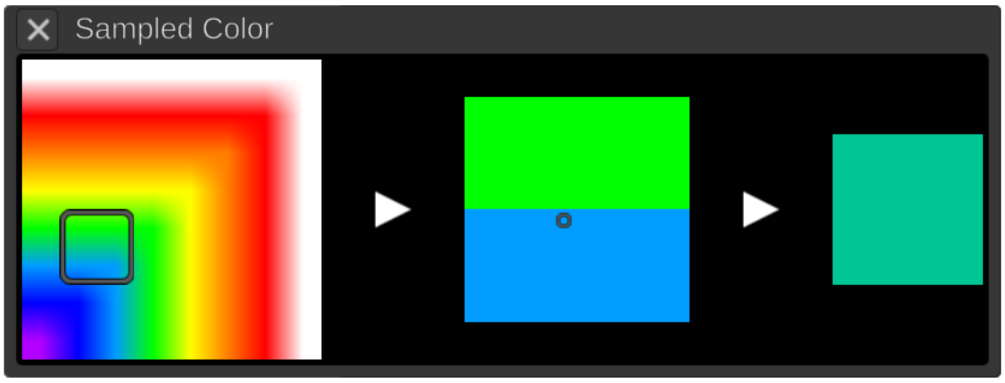


Figure 7: The render preview component is used to show sampling steps.

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.

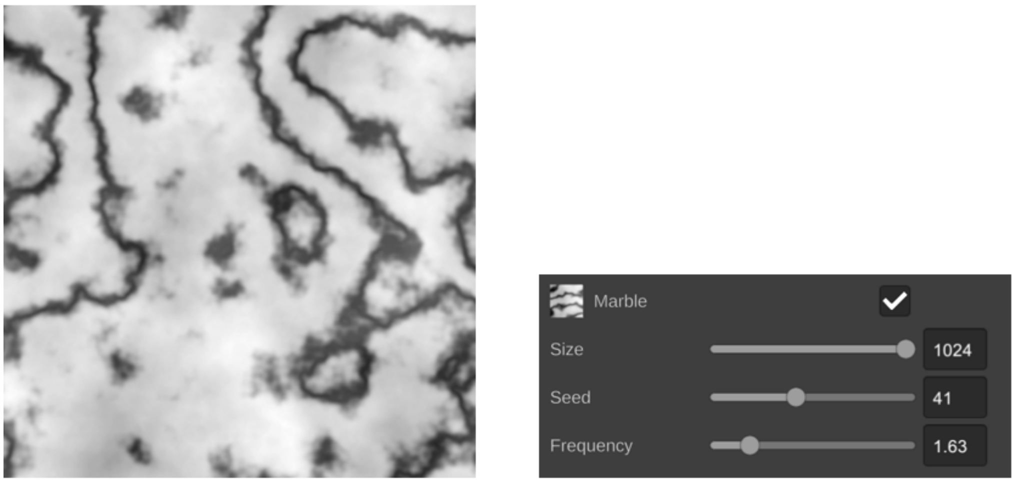

Marble ✓
Size 1024
Seed 41
Frequency 1.63

Figure 8: 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 variety of new components to VRT. The largest 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 [14]. This allows for easier ...
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 9.

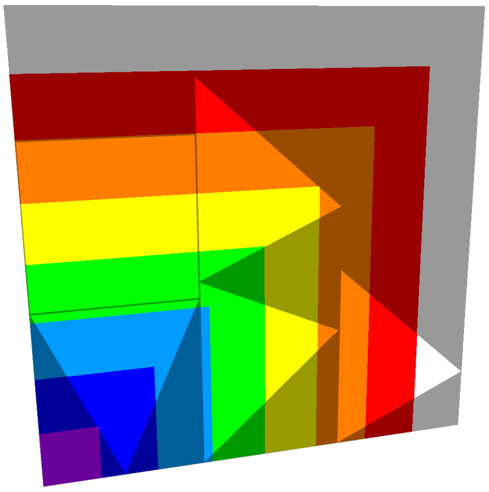
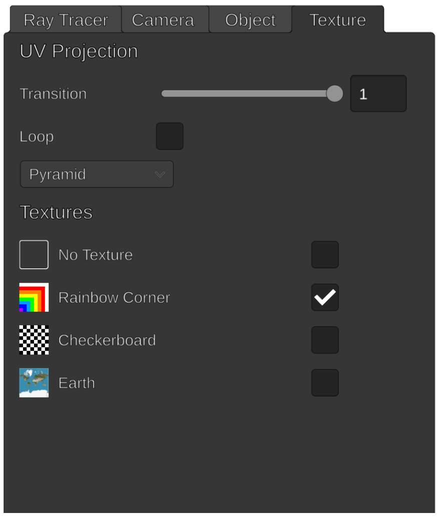


Figure 9: Only a small part of the texture is used on the object.

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, has to be modified. To make custom illustrations, new shaders and Unity's shader graphs can be written for objects that do not implement `RTMesh`. Creating new materials to apply these shaders to specific objects makes for an easier workflow.

4. **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 [15]. 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 10 shows what this tab looks like in the Texture Mapping level.



UV Projection

Transition 1

Loop ☐

Pyramid

Textures

No Texture ☐

Rainbow Corner ✓

Checkerboard ☐

Earth ☐

Figure 10: New Texture tab with interactive components for modifying textures.

5. User study

The aim of VRT is to educate users on a varienty of topics in computer graphics. This requires visualisations and explanaitions that are well understood by a general audience. Since quality of visualisations and user experience are subjective in nature, a user study deems fitting to make a good analysis of 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 afterwards. To get the user familiar with the program, the user is tasked to complete an additional two levels on control and ray tracing basics. A table with all questions and options for answers is listed in the Appendix.

5.1. **Questions in the user study**

To get a good evaluation of the quality of the product that was created, the user study is divided into 3 different sections: 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 what kind of audience was subject in the user study, 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 learns from using the application, a couple of questions are drawn up that evaluate general educational usefulness and some questions to evaluate specific additions. The first 3 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 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, we ask two questions about how the user experienced using the application:

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

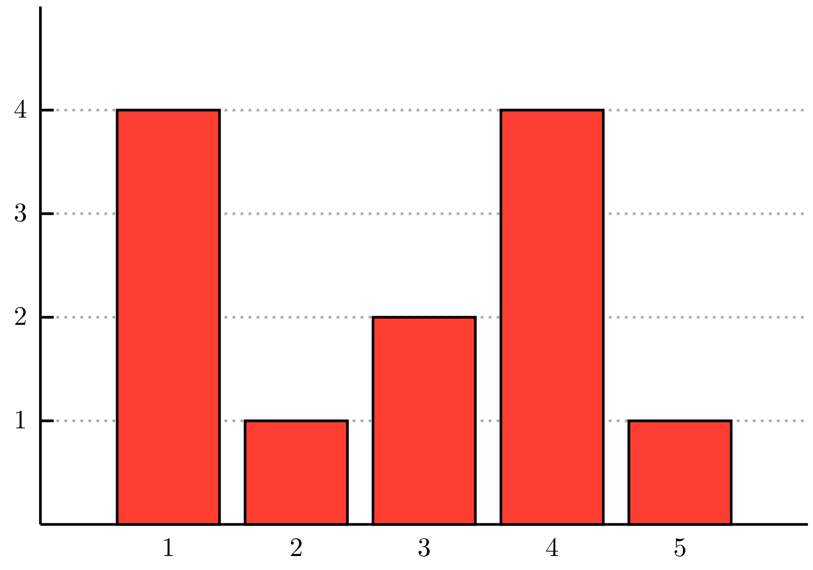
5.2. **Results**

The answers to the questions were submitted via a google forms document. A full table with questions and responses can be found in the Appendix. Most questions were evaluated on a scale from 1-5 with lower numbers indicating an answer in line with a negative response or answering "no" and high numbers indicating the opposite.

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 12 people that submitted the survey, 3 answered to have followed the course Computer Graphics. More interestingly we can see in Graph 1 that familiarity with texturing is more distributed. There were still a substantial amount of people who at least had a decent understanding of texturing.

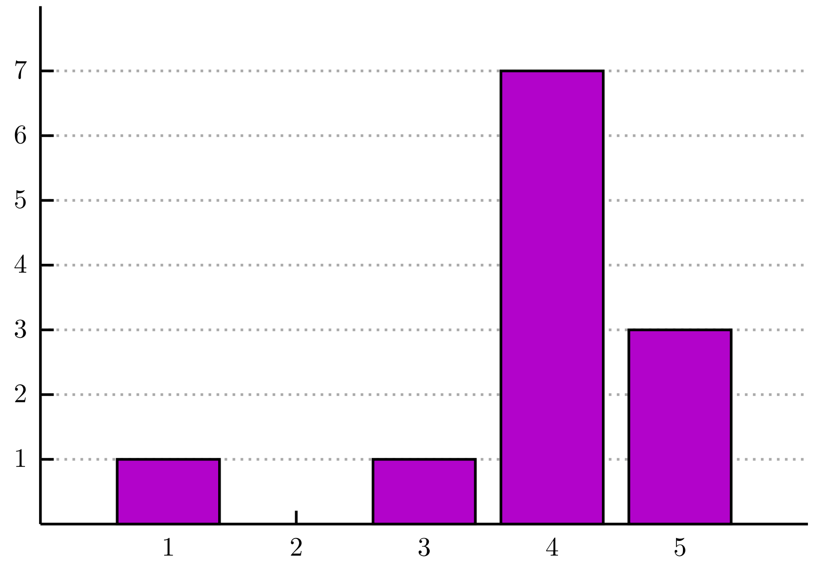
- geen/wel ervaring



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

5.2.2. *Educational usefulness*

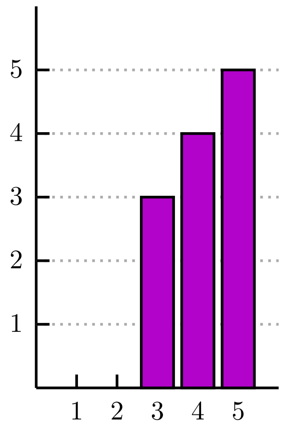
After using VRT users were first asked to answer some questions about the quality of the teaching material. When comparing what was seen before in Graph 1 with the results of Graph 2 below, it is apparent that the evaluated score went up significantly. The only outlier is the single evaluation of 1, 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 can be assumed that the user already sufficiently understood the subject matter.



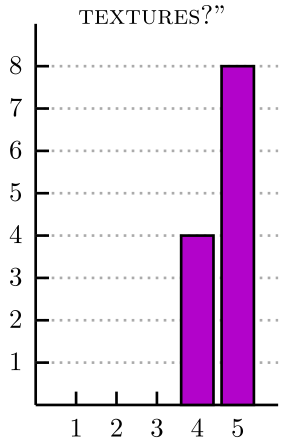
Graph 2: Did the application help you understand textures (better)?

To evaluate specific parts of the program, users were asked how much some of these components helped them understand texturing. This was all evaluated on a scale from 1 to 5. The Graphs below show that most people evaluated the quality of the different components between 3 and 5. The tutorial texts score lowest and visualisation highest as can be seen in Graph 3 and Graph 4, but without large margins.

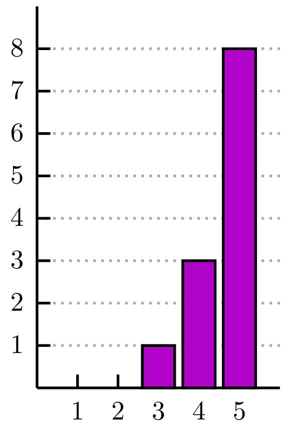
“What aspects were most helpful to you in better understanding textures?”



Graph 3: The tutorial texts



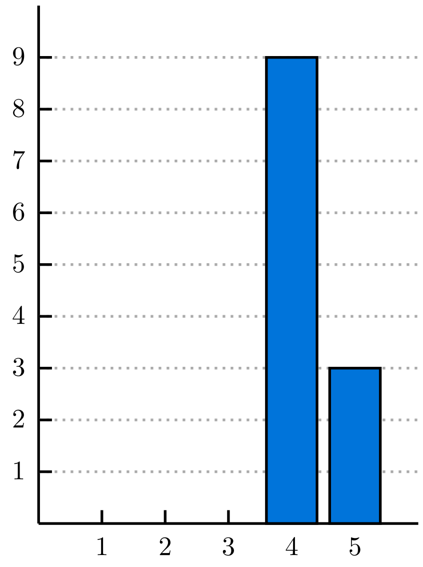
Graph 4: The visualisation



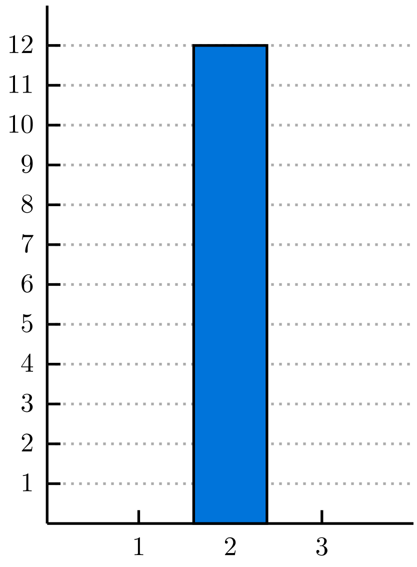
Graph 5: The ability to experiment with various settings in the menu on the right

5.2.3. *User experience*

To conclude the evaluation, the user is asked about the usability and the complexity of the application as seen in Graph 6 and Graph 7 below. It is overall well received with a good score for the usability and a middle ground for the complexity. The evaluated version of VRT only tested for 6 levels, so it might be recommended to also do a user study on the full version of VRT. After this comparisons can be made with former projects and evaluated if too much material might overwhelm the user.



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



Graph 7: How complex did you find the application?

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. This suggested that 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, 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.

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, various components were added to the project. New scripts were added for new interactions and a behaviour and event system was used for internal communication. The asset collection in VRT was expanded with new objects, materials, shaders and UI components.

To evaluate the quality of the product, a user study was conducted. The user study assessed the prior experience of the subject, 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, subjects were able to make other remarks they had on VRT. Some things that were mentioned included, confusing objectives, being stuck on tasks and small tutorial texts.

Overall both projects in the collaboration successfully produced useful content to merge into the VRT project. The application now satisfies general education in texturing with its new learning material.

7. Future Work

VRT is starting to get substantial 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 [16]. 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 [17].
- **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.

8. Acknowledgements

I would to make special thanks to my supervisors for assisting in 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. Some credit is also due to the 6 bachelor projects that developed VRT thus far [1, 2, 3, 4, 5, 6]. Finally I would like to thank everyone who participated in the user study.

9. References

[1] W. V. de la Houssaije, “A virtual ray tracer,” Thesis, Univ. Groningen, 2021.

[2] P. J. Blok, “Gamification of virtual ray tracer,” Thesis, Univ. Groningen, 2022.

[3] B. Yilmaz, “Acceleration data structures for virtual ray tracer,” Thesis, Univ. Groningen, 2022.

[4] R. Rosema, “Adapting virtual ray tracer to a web and mobile application,” Thesis, Univ. Groningen, 2022.

[5] J. van der Zwaag, “Virtual ray tracer: distribution ray tracing,” Thesis, Univ. Groningen, 2022.

[6] C. van Wezel, “A virtual ray tracer,” Thesis, Univ. Groningen, 2022.

[7] 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.

[8] 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.

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

[10] T. Couperus, “Procedural textures in virtual ray tracer,” Thesis, Univ. Groningen, 2023.

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

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

[13] 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)

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

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

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

[17] 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”