university of groningen

faculty of science and engineering

Virtual Ray Tracer: Teaching Procedural Textures

Bachelor Thesis

University of Groningen

July 7, 2023

Author:
Tom Couperus

Primary supervisor:
Prof. Dr. Jiří Kosinka

Secondary supervisor:
Dr. Steffen D. Frey

Abstract

Virtual Ray Tracer (VRT) is an open-source application to teach ray tracing to students of Computer Graphics and the general public. Since its initial version, VRT has been changed to teach a wider range of ray tracing-related concepts and improve the teaching process.

This thesis expands VRT with a step-by-step explanation of textures, making it a more general Computer Graphics teaching tool, upon the suggestion of the authors of Virtual Ray Tracer 2.0.

The effectiveness of this expansion is tested with a user study. Despite a small sample size, the expansion was positively received by all users, indicating that our expansion works as we intended.

Contents

1 Introduction

The Virtual Ray Tracer (VRT) is a tool developed by the SVCG Research Group[1] at the University of Groningen (RUG) to aid in teaching ray tracing to students of Computer Graphics and the general public [15]. It does so by visualizing complex ray paths and their interactions with objects and lights. VRT was developed because ray tracing is both complex and commonly taught in Computer Graphics courses, and there were no "openly accessible applications that focus on education" [15] for the subject. The project was met with positive results from Computer Graphics students and the general public [15].

In their latest paper on VRT [14], the authors mention several ways to improve and extend their application. They plan to incorporate ray tracing to render the scene itself, not just the simulated image, and have suggested extensions in the form of support for textures, the ability to step through the ray tracing process, and expanding the ray marching levels.

Of these suggestions, texturing, like ray tracing, is another complex and commonly taught subject in Computer Graphics courses. There are many techniques used in applying textures to objects and making them aesthetically pleasing. However, as we show in Section 2, there are no applications that focus on teaching this.

Therefore, Victor Gaya Walters and I have followed up on one of the suggestions of VRT's authors by extending VRT with a step-by-step teaching programme for texturing, including an introduction on procedural textures. In order to decide the features we needed to implement to accomplish this, we asked the following questions:

- What knowledge is required to understand texturing and procedural textures?
- What is the most educationally effective way to visualize texturing and its prerequisites?
- What is the optimal interface for generating (basic) procedural textures?

Once every feature was implemented, we conducted a user study to test the educational effectiveness of our additions.

V. Gaya Walters and I worked together on determining the general knowledge required for understanding textures, writing the step-by-step tutorial, and conducting the user study. However, we wrote separate theses and divided the research questions. V. Gaya Walters focused his efforts on visualizing texture mapping itself, whereas I worked on implementing the step-by-step tutorial for teaching the prerequisite knowledge and visualizing basic procedural textures.

Section 2 contains an overview of texturing techniques. The current features of VRT and other teaching tools are also discussed in this section. The requirements of this thesis are explained in Section 3. Section 4 covers the implementation of this expansion of VRT, and Section 5 discusses the user study, its results, and any interesting feedback we received. A conclusion is drawn in Section 6 from the results of the user study, and Section 7 describes some interesting ideas for future additions to VRT.

[1] `https://www.cs.rug.nl/svcg/Main/HomePage`

2 Related Work

2.1 Textures

Texturing is used to add detail to 3D models. This was first done by Catmull, who made an animated, textured, smooth-shaded 3D model of a hand [3]. He later calls the technique he used for applying the texture *texture mapping* [4]. Texture mapping is the process of translating each point (x, y, z) of a 3D model to a unique point (u, v) on a 2D image, as shown in Figure 1. This translation produces what is called a UV-map. Colour is then applied to the model by 'wrapping' it in the image.

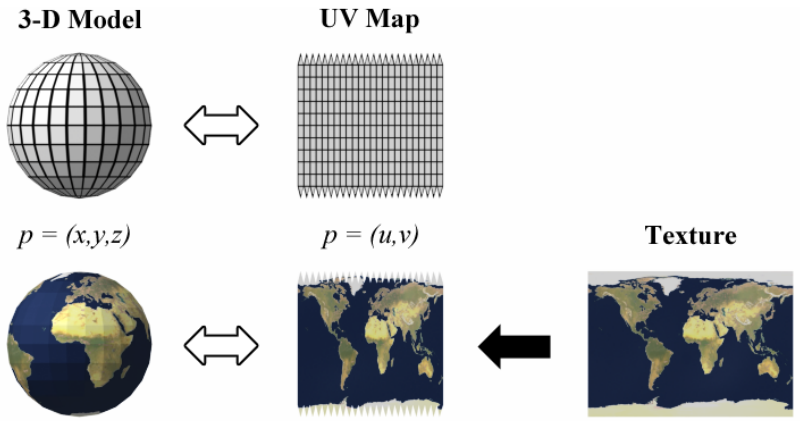


Figure 1: Texture mapping[2]

Despite the ubiquitous adoption of the technique, texture mapping has several limitations: a 2D image is distorted on all but continuous surfaces (such as flat planes, cones, and cylinders), and applying a texture to an object with multiple faces (a cube, for example) distorts the texture as well [6]. These distortions can be mitigated by manually creating the UV-map, a complex process in itself, but this greatly increases the time needed for creating the texture when using highly complex objects (such as a character model) [17].

To completely avoid texture distortions, a texture can be defined in 3D space by a mathematical formula [9, 10]. Instead of wrapping the object in an image, the object is 'carved' directly from the texture, a process called *solid texturing*. This method is well suited for textures that have 3D patterns, such as wood and stone [9], as well as clouds and waves [10]. However, there is no one formula to obtain a desired texture. Synthesizing a solid texture involves experimentally layering various functions and applying pseudo-random noise functions to avoid repetition and keep the texture interesting. Since the textures are fully obtained by a program with adjustable parameters, they are called *procedural textures*. A summary of solid texture synthesis can be found in [6].

To render the texture onto the screen, sampling is used. The simplest method is point-sampling, also known as nearest-neighbor sampling. Using this sampling procedure on detailed images or images that contain hard edges introduces aliasing artifacts. Sampling the

[2] `https://en.wikipedia.org/wiki/UV_mapping`

texture using bilinear interpolation is an easy and accurate way to reduce these artifacts [7]. Other methods to improve the quality of a rendered texture are mipmaps and anisotropic filtering.

2.2 Existing Tools

Tools to aid in the teaching of computer graphic concepts have been around since 1991, such as TUGS [5], where students build a render pipeline by improving upon basic modules. Some tools provide a more conceptual approach, focusing on changing parameters and algorithms in a scene and visualising the effects of those changes [8, 11]. Others focus on a programming approach, by letting students edit the functionality of the render pipeline [12] or providing an assessment system with immediate feedback for a student's OpenGL assignments [16]. A summary of educational computer graphic tools is found in [13].

The latest version of VRT (see Figure 2) focuses on showing various ray tracing concepts, such as reflection, refraction and super sampling [14]. However, none of these tools focus on teaching texturing or generating procedural textures, instead focusing on mesh generation, transformations, and the render pipeline.

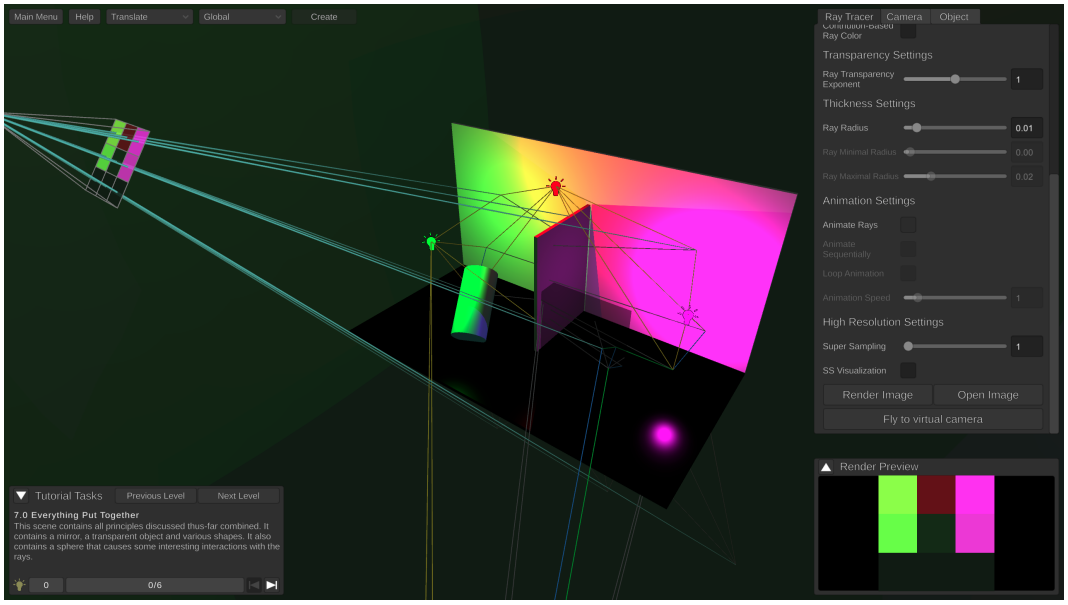


Figure 2: VRT level with all types of materials in a scene, and all options available to the user.

Professional 3D modelling tools such as Blender[3] provide support for texturing. Its latest stable version (3.4) has support for painting UV-mapped textures [2] as well as defining procedural textures using shader nodes [1]. Although it provides extensive support for texturing, we found Blender to be a tool not directly suited for Computer Graphics education as it does not explain the full texturing process in the program itself.

[3] `https://www.blender.org`

3 Requirements

As our goal is to teach the user about texturing and procedural textures, we first had to determine what we need to teach.

As we explained in Section 2.1, texture mapping is the most common method of applying textures to an object, even if it has some limitations. Because of its ubiquity, we chose it as the main method to explain in VRT.

As VRT is aimed not only at people with a moderate understanding of Computer Graphics, but also the general public, we needed to assume that a user has no prior Computer Graphics understanding before using VRT. This means that we have to explain other concepts relevant to the subject, before we explain texture mapping. Since texture mapping is defined as the process of translating each point of a 3D model to a unique point of a 2D image (see Section 2.1), the first additional subject comes to mind: what is a 3D model?

Another important subject when talking about texture mapping is texture sampling. Where texture mapping applies the texture to the object, texture sampling determines the color of each pixel. We chose to explain the two simplest and most common methods for this: nearest-neighbour sampling and bilinear interpolation.

The concepts above can all be demonstrated with simple, static textures. However, procedural textures are about generating textures based on parameters. We chose to demonstrate the power of this type of texture by allowing the user to experiment with some simple procedural textures.

This thought process, as well as taking into account the general philosophy behind VRT, gives us the following concrete requirements:

1. Explain and visualize the definition of 3D objects.
2. Explain and visualize texture mapping.
3. Explain and visualize texture sampling, in particular nearest-neighbor sampling and bilinear interpolation.
4. Explain procedural textures by giving the user simple, interactive procedural textures.
5. Implement the above in interactive, bite-sized chunks, like one would expect from a video game tutorial or a step-by-step furniture assembly guide.
6. Keep VRT easily extensible.

4 Implementation

4.1 Tools

Since we are building on the existing VRT program architecture, we used the Unity game engine and C# programming language for our project.

Apart from these programming tools, we also used Blender for creating custom UV-maps for the texture mapping explanation. We further used GIMP[4], a free and open-source image editor, for creating small textures used in the texture sampling tutorials.

As VRT's code is open-source[5], we created a fork of the project on GitHub and regularly pushed our progress to the online repository.

4.2 A Step-by-step Teaching Programme

As we were working on our requirements, we also worked on a rough planning for a single VRT level, expecting to be able to explain texture mapping in the required section of the level's tutorial, and procedural textures in the optional section. However, as our requirements became more clear, we discovered that texture mapping and procedural textures are not easy to understand for someone without basic Computer Graphics knowledge. Which is how we soon came to the conclusion to make a level for each of the four concepts we want to explain to our users:

1. 3D Objects,
2. Texture Mapping,
3. Texture Sampling,
4. Procedural Textures.

To present the information in a tutorial (Requirement 5) and decide on the required features, we first wrote a script for each level. This script contained the text of each tutorial step and a short description for the transition between steps. I will explain the script for each level in their respective sections below.

Once we had our required features worked out, we decided to visualize the concepts in an empty level. We wanted to disable as many of the existing options as possible, as they could confuse and distract the user from the purpose of our level. In some cases, such as the Ray Tracer Camera (RT Camera), where disabling was not possible, we hid the feature instead. The time required to investigate which scripts use the RT Camera and to fix other errors was simply too much, especially since we were unfamiliar with those parts of the codebase. We instead solved it by hiding the RT Camera behind the player camera. This basic level layout and the hidden RT Camera is shown as a sketch in Figure 3, where the raster and camera icon represent the RT Camera, and the eye represents the player camera.

As one of our main requirements was to keep the application easily extensible (Requirement 6), all features are implemented as a Unity Behaviour. This allows future extensions to

[4] `https://www.gimp.org`

[5] `https://github.com/wezel/Virtual-Ray-Tracer`

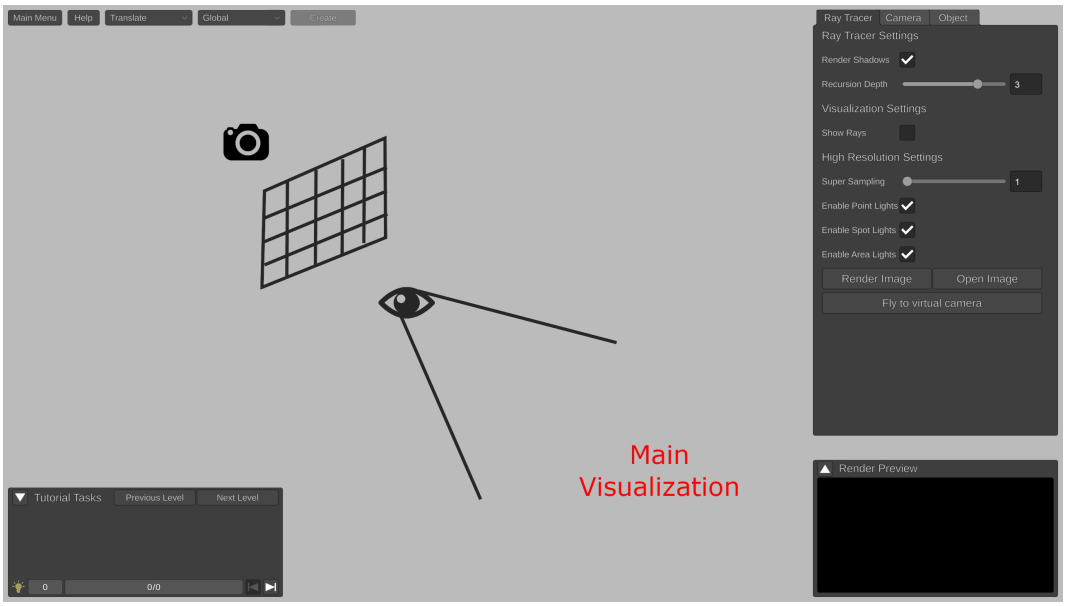


Figure 3: RT Camera hidden behind viewer, aimed at the subject of the level.

simply add or remove a Behaviour from an object, and all related functionality and UI will be enabled if the Behaviour exists on the object, or disabled if it does not.

V. Gaya Walters and I worked together on level 1. We each used different techniques to implement the features of that level, which I will explain in Section 4.3. When we were satisfied with the first level, we continued with the focus of our own theses. He worked on visualizing texture mapping (level 2), and will cover it in his thesis, whereas my focus was on procedural textures. We left the implementation of texture sampling to the one who was done first with their own part, which turned out to be me.

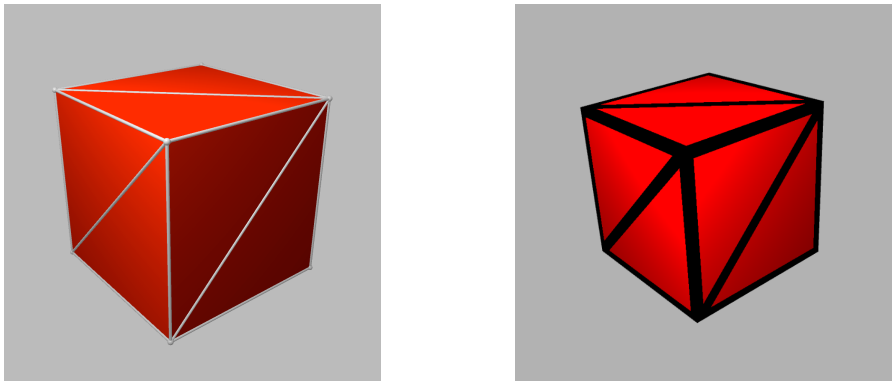
A significant part of our time was spent on familiarizing ourselves with the codebase during our experiments with the various visualization methods. However, once we were familiar with the codebase and our visualization options, development sped up significantly for the other features.

In the subsections below, I will cover the design and implementation of levels 1, 3 and 4.

4.3 3D Objects

The level on 3D objects is meant to ensure that all users know what will be taught in the Texture Mapping level. As a 3D object is defined by its vertices, edges, and triangles, we found it best to introduce the user to these concepts by starting with a shape we expect all our users to know: a cube. Our plan was to have the user check a checkbox, which would highlight the respective part of the cube, before continuing to the next step. As there are three parts, we made 3 tutorial steps.

Because we had a detailed plan before starting implementation, we were able to quickly implement the functionality. I tried out an implementation using objects to visualize the mesh components, as shown in Figure 4a, and V. Gaya Walters experimented with a shader based approach, shown in Figure 4b. We settled for using the object based approach, as we found it to look better and able to display the vertices and edges simultaneously in a clear way. This clarity was lost in the shader approach.



(a) Highlights with objects (b) Highlights with shader

Figure 4: Highlighted cube mesh

4.4 Texture Sampling

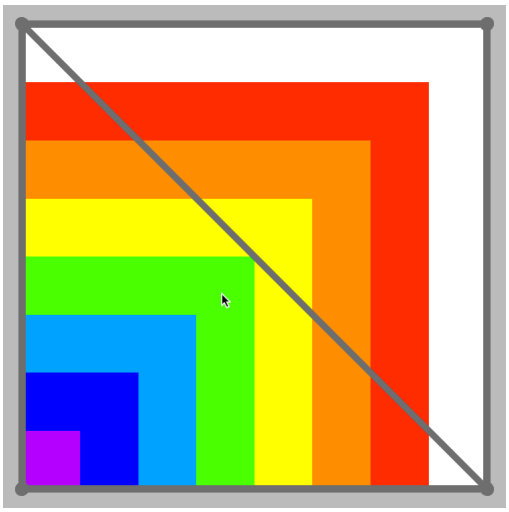
At the start of this level, the user is expected to understand texture mapping. As explained in Section 4.2, texture mapping is only half of the story; the texture is wrapped around the object, but it still needs to be displayed on the screen.

As we were working on the script for this level, we realized we required some way of visualizing this last step. To teach the user about texture sampling, we planned to allow the user to hover over the textured object and inspect the sampling process at their cursor.

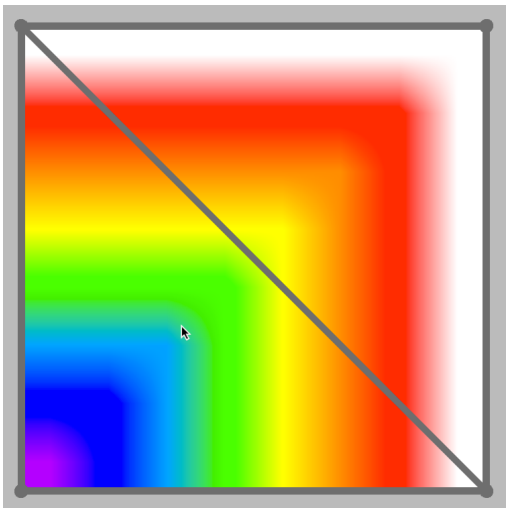
Since nearest-neighbor sampling is relatively simple when demonstrated with a small texture, we thought it best to simply show the texture pixel in a preview window (see Figure 5c).

Bilinear interpolation has multiple steps, and simply showing the end result would not be sufficient. We therefore came up with a visualization of three parts: highlighting which area of the texture is sampled, showing the four texture pixels that are interpolated at the cursor's position, and the final color. This gave us the visual shown in Figure 5d.

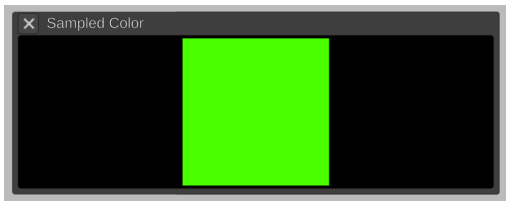
The user can switch between the two modes by changing the mode in the Object menu. The texture that is rendered on the object changes accordingly (see Figure 5a and Figure 5b), as well as the result of the Render Image button. The sampling process for the hovered-over pixel (as shown in Figure 5c and Figure 5d) is placed in the bottom right of the screen. Which of these views is shown is determined by the active sampling mode.



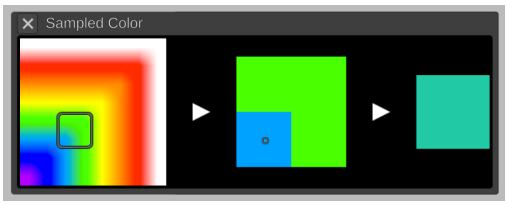
(a) Point sampled plane



(b) Bilinear sampled plane



(c) Point sample view



(d) Bilinear sample view

Figure 5: Sampling visualization: the plane with cursor position in the top row, and the resulting sample view in the bottom row.

4.5 Procedural Textures

As I mentioned in Section 4.2, we explain the concept of procedural textures by having the user experiment with some simple textures. The textures we chose to implement were brick and marble.

To switch between these and other textures, we made the TextureManager. This is a Behaviour that can be put on an object, and manages the current and supported textures of that object.

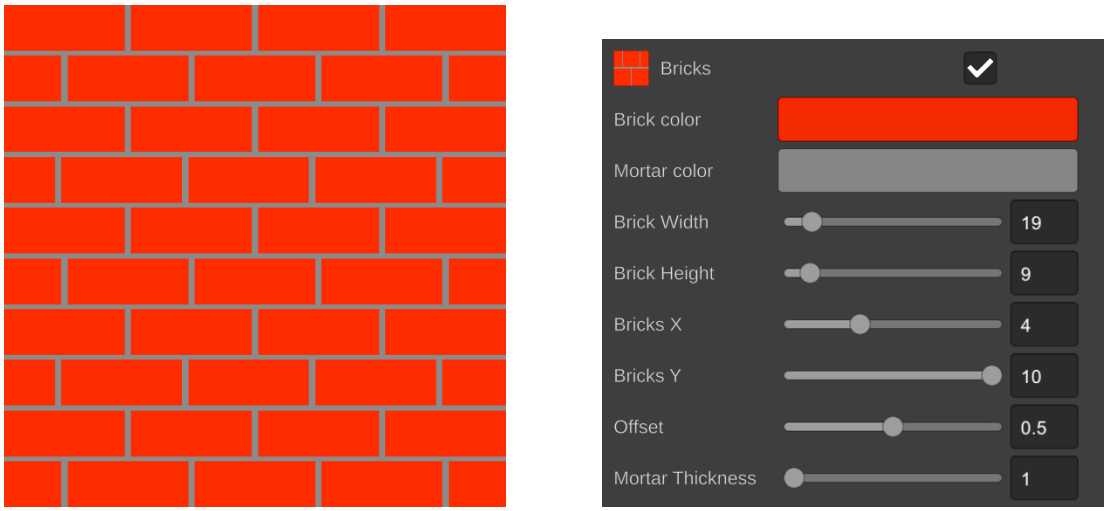
We implemented the procedural textures as flat, non-wrapping 2D images. This means that both procedural textures are only suited for display on a plane.

4.5.1 Procedural Bricks

We chose a brick texture because it has a repeating pattern, and most, if not all, users are familiar with it. We explain that prior to this level all textures were made by hand. However, since bricks have such a repeating pattern, it makes little sense for a brick texture to be hand-drawn. As bricks and their repeating pattern should be familiar to users, this thought process ought to be easy to follow and aid in the user's understanding.

Once the user is familiar with the idea of procedural textures, they can change the following about the brick texture, using the UI shown in Figure 6b:

- Brick and mortar color,
- Brick width and height,
- Number of horizontal and vertical bricks,
- Thickness of the mortar between bricks.



(a) Brick texture (b) Procedural brick UI

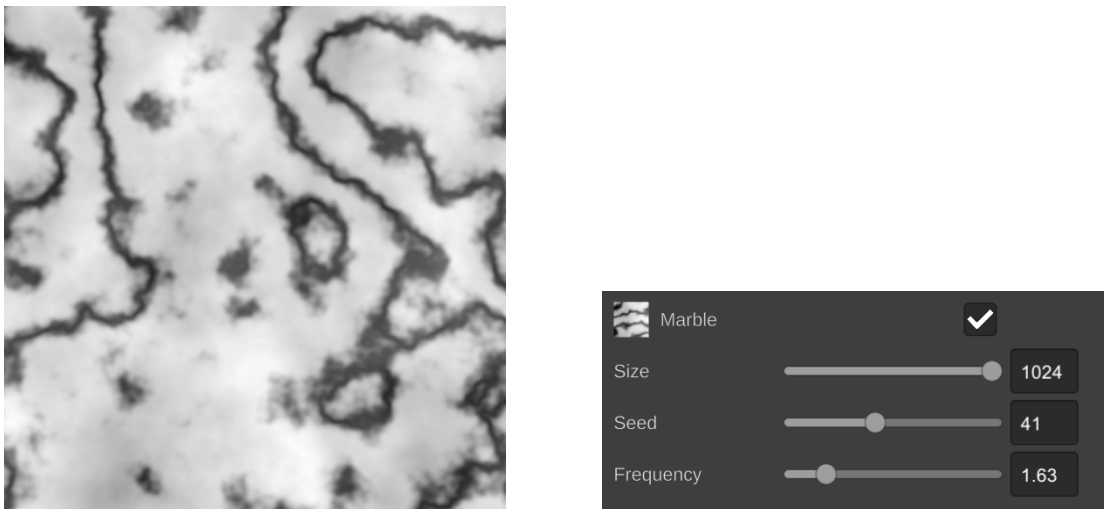
Figure 6: Procedural brick texture

4.5.2 Procedural Marble

The choice to include a procedural marble texture was made as an introduction to a more natural looking texture, compared to our simple brick texture. The texture function was created based on the formula for the marble texture found in [10] and a Blender tutorial video for a procedural marble texture[6].

The user can change the following about the marble texture, using the UI shown in Figure 7b:

- Texture size,
- Marble seed,
- Frequency, which controls the closeness and the number of cracks in the marble.



(a) Marble texture (b) Procedural marble UI

Figure 7: Procedural marble texture

[6] `https://www.youtube.com/watch?v=vV9zZ3Tu7SI`, accessed: June 2023

5 User Study

All previous versions of VRT were tested with a user study [14, 15]. We found this a good idea for our theses as well, and created a user study too. It covers the same general subsections as previous studies: prior experience with the subject, and educational usefulness and user experience of the application.

The target audience of VRT is students of Computer Graphics and the general public. We wanted to know how these two groups experience our expansion to VRT, so we asked for participants among friends and family, as well as among people at the RUG who followed their Computer Graphics course. Our study collected no personal information, apart from asking if a user followed the course.

5.1 User Study VRT Build

Before evaluating our user study, I will first describe the level selection we chose for our users to test. Since we made an expansion to VRT and all previous versions were already evaluated, it made little sense to include all of their levels in another user study. Therefore, we chose to include only two levels from the previous VRT version in order to tie our levels into VRT's existing levels and present a coherent story to the users. The two levels we chose are the introductory level ('Introduction'), which teaches the controls, and the 'Basic Ray Tracing' level, which teaches the user about ray tracing. The latter is necessary, because our explanations about texturing are presented in a ray tracing context. In the end, we presented the following levels to the users in our study, in order:

1. Introduction
2. 3D Objects
3. Basic Ray Tracing
4. Texture Mapping
5. Texture Sampling
6. Procedural Textures

5.2 Questions

5.2.1 Prior Experience

This set of questions is asked before the user opens VRT. It is intended to give us information to judge further questions with more context.

The first question we ask is if the user followed the Computer Graphics course. This question is important, as all participants are anonymous and a later question is intended only for those who followed the course.

The second question asks how the user would judge their own understanding of textures. We expect a user's answer to this question to heavily influence their answers to the questions asked in the next section. The user can answer within a range of 1 to 5, where the 1 indicates they say they have no understanding of textures, and a 5 indicates a user who understands textures perfectly.

Once the user has answered these two questions, the link to VRT becomes available, and we ask them to play through all of the available levels.

5.2.2 Educational Usefulness

Once the user has played through all the levels of this user study, they are asked to review the educational usefulness of the application. This set of questions should allow us to determine if our extension to VRT was actually helpful, and what parts made it so.

The first question asked in this set is if the user understands textures better compared to before they used VRT. We see this as one of the most important metrics we can get from our study; If the application helps most people understand textures better, we will have achieved our goal. We expect this question to be the most influenced by the second question of the previous set.

As the focus points of our theses are different, we deliberated on splitting this question into two: one aimed at texture mapping and another aimed at procedural textures. However, as these two questions would not have included the two other subjects we teach and both subjects fall under 'textures' in general, we found it best to ask only the single question.

We also ask our users if they think the application can help other people in general to better understand textures. This question is posed to all our participants, but we also pose a similar question to our course member participants: we ask if they think this expansion of VRT can be useful for future RUG Computer Graphics students. We ask this last question because one of VRT's main points is to be used as a teaching aid in that course.

Last in this set of questions we ask our users what parts of the application were most helpful to them. As we use textual explanations, visualizations, and experimentation to teach our users about textures, we would like to know which of these was the most helpful, and which could use improvement. We asked our users to grade these three aspects on a scale of 1 to 5, from 'not helpful' to 'very helpful'. To clarify, with each of these three parts we target the following:

- Textual explanations: The text shown in the bottom left of the screen, in the tutorial panel.
- Visualizations: The visualization is displayed on the main, center part of the screen and the window in the bottom right.
- Experimentation: The various options available to play with in the UI on the right side of the screen.

5.2.3 User Experience

The last set of questions is aimed at usability, complexity, and general feedback.

With the first question, we ask our users to rate the usability on a scale of 'difficult and confusing' to 'user friendly'. We aim to determine the clarity of our additions to the interface as well.

The second question is targeted towards the complexity of our additions. Since we hid most of the ray tracing features by hiding the RT Camera, and introduced options of our own, we would like to know if we made the application too simple or too complex.

Finally, we have an open question for general feedback. We hope to gather additional feedback that was not covered already by the previous questions, by allowing the user to express their opinion more freely.

5.3 Results

We managed to find 12 participants for our study, 3 of whom followed the RUG Computer Graphics course.

In the subsections below I will discuss the results of our questions in the same categories I introduced them. The results are included in tables. If a user followed the course, their answer to a question is included in the **CG** row of the results of that question. If they did not follow the course, their answer is added to the **Non-CG** row.

5.3.1 Prior Experience

Table 1 lists the answers to the only question in this category.

Those who answered 'yes' to having followed the Computer Graphics course at the RUG all answered that they, before using VRT, understood textures moderately (33%) to well (66%), with no one who said they did not know what textures were.

Among those who did not follow the course, the results are more evenly distributed. The majority (44%), however, did not know what textures were before using VRT. Interestingly, one person who did not follow the course said that they understood textures perfectly, before using VRT.

The total distribution of answers to this question should allow us to determine a difference in usefulness for those who understood textures beforehand or not.

Questions		Results					
	Not at all	1	2	3	4	5	Perfectly
How well would you say you understand textures?	*CG*	0	0	1	2	0	
	Non-CG	4	1	1	2	1	

Table 1: Prior experience

5.3.2 Educational Usefulness

This subsection of our user study starts with one of the most important questions: 'did VRT help you understand textures better?' The answer to this question, regardless of a user's participation in the course, is overwhelmingly positive (92% who said they understood textures at least moderately better).

One user did not experience any benefits of using VRT. However, they are the same participant who said they already understood textures perfectly.

For the second question in this subsection, members of both the **CG** and **Non-CG** groups all replied with a 4 or higher. This means that all participants of our study thought the application can help other people in learning about textures.

We also asked the same question, changed to helping future member of the course, but only to our **CG** group. The answers were identical to the answers this group gave to the previous question.

Table 2 contains the complete distribution of answers to these questions.

Questions		Results					
	Not at all	1	2	3	4	5	Very much
Did the application help you understand textures (better)?	*CG*	0	0	1	1	1	
	Non-CG	1	0	0	6	2	
Do you think the application can help others understand textures?	*CG*	0	0	0	1	2	
	Non-CG	0	0	0	5	3	
Do you think the application can help future students of the RUG Computer Graphics course understand textures?*		0	0	0	1	2	

*Only answered by the **CG** group

Table 2: Educational usefulness

The last questions in this set were to determine which of our explanation methods were the most helpful. As can be seen in Table 3, the visualizations were judged as the most helpful, with 100% of all participants rating it a 4 or higher, and 67% rating it with a 5. Interestingly, all course member participants rated the visualizations with a 5, whereas the non-course members were almost evenly split between a 4 and a 5.

Second most helpful was the ability to experiment. 92% rated this feature a 4 or higher, also with 67% giving it the maximum score. The **CG** group is also more divided on this aspect, compared to the visualizations.

Lastly, the tutorial texts were still helpful, but less so than the other two features. 75% of all participants rated this feature better than mediocre (≥ 4), with even more people of both groups tending towards mediocre. Less than half (42%) rated the textual explanations with a 5.

None of the responses were negative, however, which is encouraging.

Questions		Results					
	Not helpful	1	2	3	4	5	Very helpful
The tutorial texts	*CG*	0	0	1	1	1	
	Non-CG	0	0	2	3	4	
The visualization	*CG*	0	0	0	0	3	
	Non-CG	0	0	0	4	5	
The ability to experiment with various settings in the menu on the right	*CG*	0	0	0	1	2	
	Non-CG	0	0	1	2	6	

Table 3: Explanation methods

5.3.3 User Experience

The responses to the last subsection of our study are listed in Table 4 and Table 5. The first of these tables lists the usability scores, which all of our participants rated at a 4 or higher. 25% of our users rated it with a 5, the highest rating we could ask for.

The second table shows our users' opinions regarding complexity. We asked them to rate the complexity among three categories: too simple (represented with the 1), too complex (represented with the 3), and just the right number of options, in the middle. All of our participants said that the complexity of our expansion is just right.

These responses indicate that we have kept VRT user friendly.

Questions		Results					
	Difficult and confusing	1	2	3	4	5	User friendly
How would you rate the usability of the application?	*CG*	0	0	0	2	1	
	Non-CG	0	0	0	7	2	

Table 4: User experience

Questions		Results			
	Too simple	1	2	3	Too complex
How complex did you find the application?	*CG*	0	3	0	
	Non-CG	0	9	0	

Table 5: Complexity

Our last question of this set was an open question for more general feedback, and below I will discuss some of the more interesting points.

- Some of our users found some technical terms difficult to understand, slowing down their overall learning. None of the users who experienced this hinted at specific terms. As the feedback is anonymous, we cannot use it to track down the exact issue.
- One user found the text in the tutorial window too small. As our users are anonymous, we do not know about any of the factors that could influence this, such as screen resolution, the user's sight, or other factors.
- One user reported they got stuck on some tutorial steps, and mention not being able to find a key. The example they give is not being able to find the left control button. Perhaps this can be improved with a more visual tutorial, highlighting the buttons we want a user to press for that particular step.

6 Conclusion

The goal of this thesis was to expand VRT with a teaching programme for textures, in order to make VRT a more general Computer Graphics teaching tool. This programme teaches not only texture mapping and procedural textures, but also fundamental knowledge required to understand those two subjects. Each concept we wanted to explain is explained in its own level. The concepts are also visualized and interactive.

To implement this programme in such a way that kept the application easily extensible for future projects, we first wrote a detailed script that describes each subject in small steps. With this script, we implemented each functionality as its own Behaviour, giving future bachelor students the ability to easily enable and disable the features created for this thesis.

To test the effectiveness of the implementations, we conducted a user study with 12 participants. This is a relatively low number of participants, which means the results of the study cannot be relied upon for a definitive conclusion. Instead, I will treat the results as a decent indication of our success.

One of the most important results of our study is that 92% of our users said that they, regardless of knowledge of textures before testing VRT, understood textures at least moderately better after using VRT. Furthermore, all of our participants felt that the application can be used by other people to understand textures better. Our participants who followed the RUG Computer Graphics course were asked if they felt that VRT could be useful for future members of the course, and all replied positively.

Regarding the various methods we use to teach teach textures (textual explanations, visualizations, and experimentation), the participants of our study said that every method helps, though one helps more than the others. Specifically, the visualizations were found the most useful, with 100% of our participants scoring it with a 4 or better. The ability to experiment received a 4 or higher from 92% of our users, and the textual explanations were rated a 4 or higher by 75% of our users.

We discovered some issues from the general feedback, such as people getting stuck on tutorial steps, not following the more technical explanations, or the size of the tutorial text being too small. I will discuss some solutions to this in Section 7.

Overall, however, this extension to VRT was very positively received.

7 Future Work

VRT has been the subject of 6 bachelor theses prior to this thesis, and we will almost certainly not be the last to work on this application. It started out as a tool to visualize ray tracing, but, with this thesis, VRT has become a more general Computer Graphics teaching tool. It would be interesting and useful to see this trend continue, as there are many more parts of Computer Graphics that can be explained with visualizations or other methods. I will mention some suggestions for future development of VRT in a list below.

Splines Since this program is intended for students who follow the RUG Computer Graphics course, it makes sense to complete the list of subjects taught. The ray tracing and texture explanations cover a significant portion of the course, and a remaining part of that course is smooth curves, also known as splines. Based on the many tools that already use splines for smooth curves, such as various vector graphics applications like InkScape[7], I feel that an introduction to this subject could fit into VRT.

Animations Another major part of Computer Graphics are animations. This is not taught in the RUG course, but it is a very visual application of Computer Graphics, which could make it an ideal subject to explain the basics of in VRT.

Accessibility A major part of the feedback we received in our user study can be traced back to accessibility. Issues such as text size and getting stuck in certain tutorial steps are hampering the learning process provided by VRT for some people. Introducing a suite of accessibility settings to solve these and other issues could prove to be a very valuable addition to VRT.

Solid textures This project explains textures by using texture mapping. However, as I covered in Section 2.1, this is not the only way to add detail to an object. Since our current implementation of procedural textures simply creature a 2D texture image, it would be very interesting if this can also be done as a solid texture. This would enable the procedural textures explanation to also show examples of actual 3D objects, instead of simply a flat plane in 3D space.

[7] `https://inkscape.org`

Acknowledgements

I would like to thank my supervisors Jiří Kosinka and Steffen D. Frey for their guidance and invaluable feedback. Without them this project would not be as good as it is. I also thank Victor Gaya Walters for working with me on this project. His perspective made me consider our explanations from different angles, making for a better tool overall. And finally, I want to thank my family for their amazing support, for proofreading my thesis and suggesting improvements.

References

[1] Blender Documentation. Texture shader nodes, 2023. URL `https://docs.blender.org/manual/en/latest/render/shader_nodes/textures/index.html`.

[2] Blender Documentation. Texture paint, 2023. URL `https://docs.blender.org/manual/en/latest/sculpt_paint/texture_paint/introduction.html`.

[3] Edwin Catmull. A system for computer generated movies. In *Proceedings of the ACM Annual Conference - Volume 1*, ACM '72, page 422–431, New York, NY, USA, 1972. Association for Computing Machinery. ISBN 9781450374910. doi: 10.1145/800193.569952.

[4] Edwin Earl Catmull. *A Subdivision Algorithm for Computer Display of Curved Surfaces.* PhD thesis, The University of Utah, 1974. AAI7504786.

[5] John Clevenger, Rick Chaddock, and Roger Bendig. Tugs—a tool for teaching computer graphics. *SIGGRAPH Comput. Graph.*, 25(3):158–164, jul 1991. ISSN 0097-8930. doi: 10.1145/126640.126648.

[6] JM Dischler and D Ghazanfarpour. A survey of 3d texturing. *COMPUTERS & GRAPHICS-UK*, 25(1):135–151, FEB 2001. ISSN 0097-8493. doi: 10.1016/S0097-8493(00)00113-8.

[7] Steve Marschner and Peter Shirley. *Fundamentals of Computer Graphics, Fourth Edition.* A. K. Peters, Ltd., USA, 4th edition, 2016. ISBN 1482229390.

[8] Avi C. Naiman. Interactive teaching modules for computer graphics. *SIGGRAPH Comput. Graph.*, 30(3):33–35, aug 1996. ISSN 0097-8930. doi: 10.1145/232301.232330.

[9] Darwyn R. Peachey. Solid texturing of complex surfaces. *SIGGRAPH Comput. Graph.*, 19(3):279–286, jul 1985. ISSN 0097-8930. doi: 10.1145/325165.325246.

[10] Ken Perlin. An image synthesizer. *SIGGRAPH Comput. Graph.*, 19(3):287–296, jul 1985. ISSN 0097-8930. doi: 10.1145/325165.325247.

[11] Dino Schweitzer, Jeff Boleng, and Lauren Scharff. Interactive tools in the graphics classroom. In *Proceedings of the 16th Annual Joint Conference on Innovation and Technology in Computer Science Education*, ITiCSE '11, page 113–117, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450306973. doi: 10.1145/1999747.1999781.

[12] Shogo Sueyasu, Tatsuro Nagano, Hiroaki Nishino, Tsuneo Kagawa, and Kouichi Utsumiya. An interactive cg learning system through 3d authoring and programming. In *2010 International Conference on P2P, Parallel, Grid, Cloud and Internet Computing*, pages 115–122, 2010. doi: 10.1109/3PGCIC.2010.23.

[13] Thomas Suselo, Burkhard C. Wünsche, and Andrew Luxton-Reilly. Technologies and tools to support teaching and learning computer graphics: A literature review. In *Proceedings of the Twenty-First Australasian Computing Education Conference*, ACE '19, page 96–105, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450366229. doi: 10.1145/3286960.3286972.

[14] Chris S. van Wezel, Willard A. Verschoore de la Houssaije, Steffen Frey, and Jiří Kosinka. Virtual ray tracer 2.0. *Computers & Graphics*, 111:89–102, 2023. ISSN 0097-8493. doi: 10.1016/j.cag.2023.01.005.

[15] Willard A. Verschoore de la Houssaije, Chris S. van Wezel, Steffen Frey, and Jiri Kosinka. Virtual Ray Tracer. In Jean-Jacques Bourdin and Eric Paquette, editors, *Eurographics 2022 - Education Papers*. The Eurographics Association, 2022. ISBN 978-3-03868-170-0. doi: 10.2312/eged.20221045.

[16] Burkhard C. Wünsche, Zhen Chen, Lindsay Shaw, Thomas Suselo, Kai-Cheung Leung, Davis Dimalen, Wannes van der Mark, Andrew Luxton-Reilly, and Richard Lobb. Automatic assessment of opengl computer graphics assignments. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE 2018, page 81–86, New York, NY, USA, 2018. Association for Computing Machinery. ISBN 9781450357074. doi: 10.1145/3197091.3197112.

[17] Cem Yuksel, Sylvain Lefebvre, and Marco Tarini. Rethinking texture mapping. *Computer Graphics Forum*, 38(2):535–551, MAY 2019. ISSN 0167-7055. doi: 10.1111/cgf.13656.