communicated by:
Lehr- und Forschungsgebiet Informatik 9

Prof Dr. Ulrik Schroeder

The present work was submitted to Learning Technologies Research Group
Diese Arbeit wurde vorgelegt am Lehr- und Forschungsgebiet Informatik 9

Interactive Visualization of Rasterization as step of the CG Rendering Pipeline in Virtual Reality

Veranschaulichung von Rasterization in Virtual Reality

Bachelor-Thesis
Bachelorarbeit

presented by / von

Koch, Julian
395322

Prof. Dr.-Ing. Ulrik Schroeder

Prof. Dr. Leif Kobbelt

Aachen, May 2, 2022

Contents

List of Figures

Abstract

In this work, a learning application for the topic of rasterization is being designed, implemented and evaluated as one part of the RePiX VR project. The RePiX VR project is a learning application for learning the Rendering Pipeline in Virtual Reality and is split into several substages which explain different parts of the Rendering Pipeline. The Rendering Pipeline is a concept used in Computer Graphics to create an image on a screen from an application. In this work a new implementation for the rasterization stage of that application is designed and implemented. Rasterization covers the step of determining which pixels in the resulting image are covered by which objects in the application. The new stage implementation consists of multiple interactive elements and explains every aspect relevant to the topic of rasterization shortly. The stage itself is build from six substages that each cover one question one could ask about rasterization. Each substage has an own illustration that explains the topic to the user. In one substage the learner is able to create and rasterize their own objects. The stage is evaluated by testers that play the new stage and are surveyed afterwards. The survey includes the questions of the System Usability Scale which can be used to evaluate and compare systems to another. Other questions cover the strengths and weaknesses of the new stage. Future improvements for the stage and RePiX VR are derived from that evaluation.

Part I

Introduction

Chapter 1 Introduction

This chapter starts by giving a motivation and goal for the work of this thesis and introduces to the topic of the *Rendering Pipeline*, *rasterization* and the *RePiX VR* project. In following sections the structure of this thesis is given.

1.1. Motivation

Interactive learning environments in Virtual Reality (VR) are a great concept to teach learners and make them remember what they have learned. The Rendering Pipeline is a well suited concept which can be taught in an interactive VR learning environment.

Give a brief motivation of this work

The Rendering Pipeline is an concept used in Computer Graphics (CG), which creates a 2D image from given 3D geometry and involves multiple steps. One of the steps is the rasterization step in which, given 2D screen-coordinates of all vertices of a polygon, all the fragments should be calculated that are covered on the screen by the polygon.

1.2. Goal

The goal of this work is to rework the current state of the rasterization stage in RePiX VR, which is a project developed in the LTI-Lab of the *Learning Technologies Research Group*. RePiX VR is short for "Rendering Pipeline eXperience in VR" and is a learning application that uses Virtual Reality (VR) to guide the user trough an explanation of the Rendering Pipeline [9]. The main goal of this work is to conceptualize and implement a more detailed rasterization stage for the application as well as to evaluate this new designed stage. The stage should explain the step of rasterization and give the user an understanding of what happens during rasterization by interactive visualizations.

In the following an outline of the stage that will be implemented as part of this work will be given. The stage will be implemented in a style similar to the other stages in the learning application, thus it will contain multiple substages and interactive components for the player. The explanations will use text panels that are located in front of the guide of the tour named *Kyle*, just as in the other stages of the application. You can see a picture of the robot Kyle in Fig. 1.1. The first explanations from Kyle cover the input and the output of the stage. When proceeding to the next panel a 3D scene (with several objects in it) is appear-

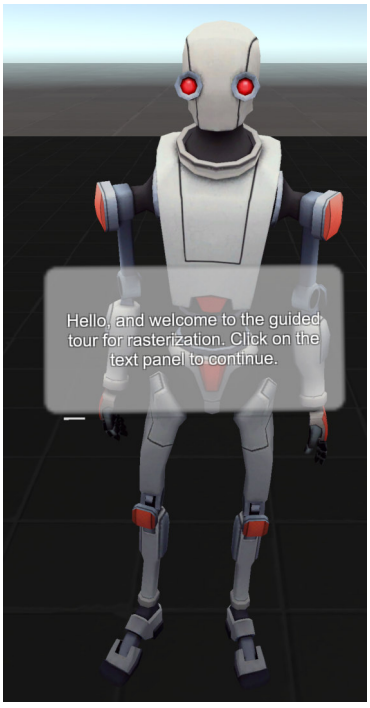

Figure 1.1.: Screenshot from RePiX VR. Kyle and its text panel can be seen in the learning environment.

ing and a screen with an adjustable resolution is shown to the user. In the scene different polygons (all triangles) are highlighted in a fixed order one at a time and on the screen the corresponding fragments that are calculated by the rasterization step are highlighted. The setup can be seen in the storyboard in Figure 1.2. The highlighted triangle from the left scene is shown and rasterized on the right screen. The player can adjust the resolution of the screen to see how this changes the highlighted fragments. It is explained on the text board that mainly two different algorithms can be used for the rasterization, the Pineda and the Bresenham algorithm. Since the application is focused on different learners in the environment, which have a different interest in learning about the details of the stages, the stage ends here without going much more into detail. But at the end of the stage the option is given to go into different substages which go into further details on the used algorithms and different questions the learner could have regarding rasterization. In these substages the algorithms are explained in more detail and how they work is illustrated with animations on a screen. These substages are created for more interested learners, because of the complexity of the algorithms and they have more interest in a deeper understanding of the topic and not only on the surface. In one of the substages the player has the choice to select the highlighted polygon inside the 3D scene themselves and the player even has the possibility to create triangles on their own and observe how these triangles look like in the resulting 2D image when rasterized.

Introduction to RePiX VR and the goal of this work with the content of the new stage

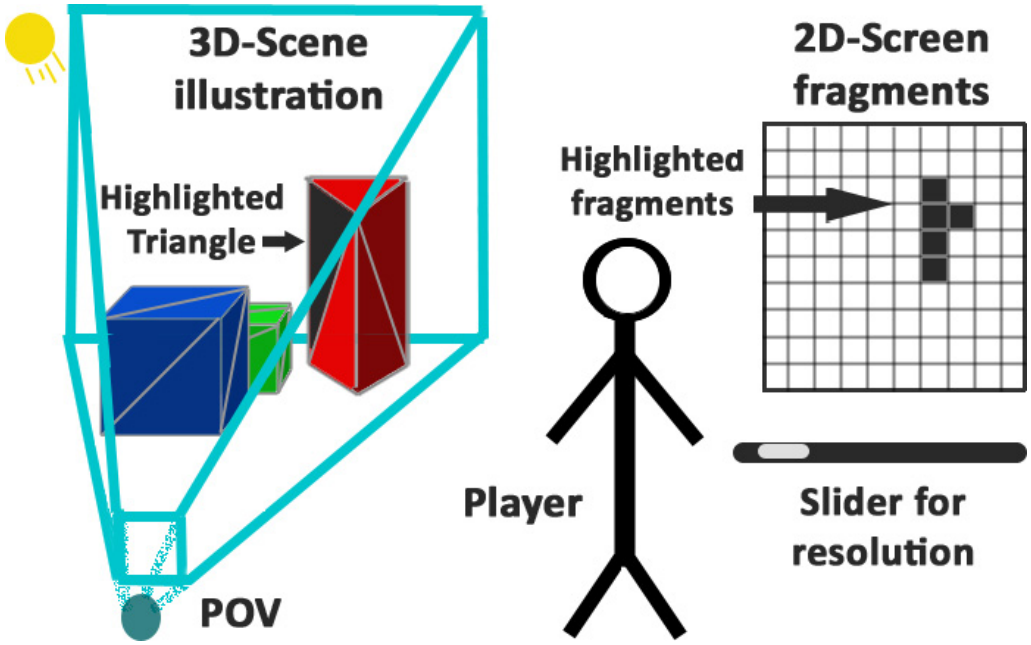

Figure 1.2.: Early sketch for the interactive Rasterization Stage

The conception and implementation of the RePiX VR project is designed using *Design-based Research* [2] and *Iterative Design* [10] which means that the focus is on designing the application iterative that is optimal for all stakeholders of the project. The iterative design means, that the stage is designed, implemented and evaluated multiple times based on the evaluation of the last iterative design by experts' feedback, resulting in a evolving complexity and user friendly design. This work is one of these iterative cycles in which the project gets improved.

1.3. Structure of the Thesis

This section will explain the structure of this thesis and give a brief overview over the different chapters. This thesis is split into three parts: The Introduction part gives a general understanding of the topic with Chapter 2 which describes the Scientific Background of this project and everything related to rasterization as well as giving reasons for the relevance of this work. This is followed by Chapter 3 which introduces all relevant related work that could be interesting for the reader and can be used to look up further information about the respective topics.

Overview over the structure of this thesis

The second part of this thesis is the main part which covers all details about the new implementation of the rasterization stage. It contains Chapter 4 that is split up into three sections with multiple subsections. Section 4.1 has all information about the structure of the new stage and all explanations that are given to the learners in the learning environment. All substages of the stage have their own subsection for explanations. Section 4.2 introduces another stage and its content that is used for the evaluation of the project and thesis. At last Section 4.3 covers all relevant information about the design decisions and implementation details made in the process of this thesis.

The third part of the thesis is the evaluation. It contains Chapter 5 the evaluation itself which is split up into three sections. Section 5.1 deals with the evaluation of the short old stage that was implemented before in the RePiX VR project and comparing it to the new stage. Section 5.2 is the main section of this chapter that deals with the detailed evaluation of the new stage using feedback from multiple testers. The last Section 5.3 lists possible improvements for future development of the stage and RePiX VR. This chapter is followed by the last Chapter 6 of the thesis which gives a summary and conclusion of the whole thesis.

Chapter 2 Scientific Background

Overview what this chapter is about

In this chapter the concepts, algorithms, and the software and hardware used for this project are introduced. The following chapters contain information that are necessary to understand this work about the Rendering Pipeline, the algorithms used for rasterization and the software used for implementation as well as the for Virtual Reality (VR) used hardware.

2.1. Rendering Pipeline

Introduction to the concept of the Rendering Pipeline

The Rendering Pipeline (RP) is a widespread used concept in Computer Graphics (CG) with multiple steps for the creation of a 2D image from a given 3D scene and geometry in it. A 3D scene is made of different *primitives* which are in most cases *polygons* which in turn are often simple triangles. A simplified model of the different steps that are used in the Rendering Pipeline can be seen in Fig. 2.1.

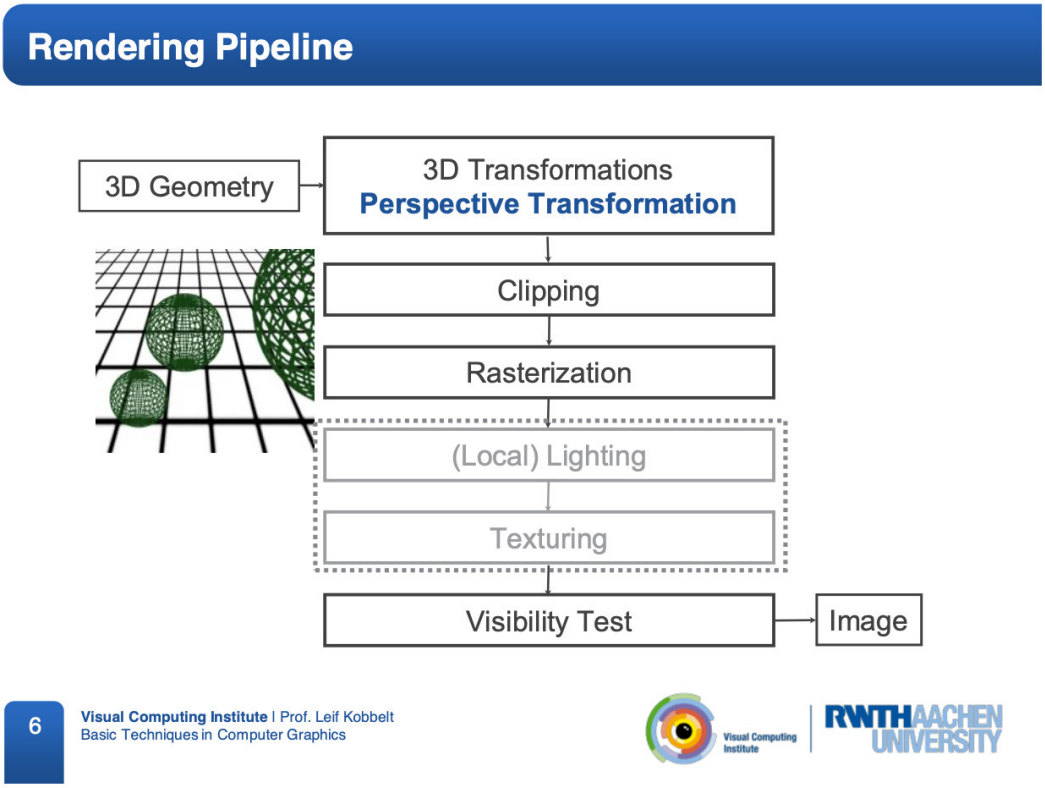

Figure 2.1.: Simplified model of the Rendering Pipeline; split up into several steps; rasterization is one of them. Slide taken from the CG lecture *Basic Techniques in Computer Graphics* by Prof. Dr. Leif Kobbelt.

One of these steps and the focus of this thesis is the rasterization step in which for every primitive in the scene all fragments are calculated that are covered by the primitive (cf. [12], p. 22f.). The input are the coordinates of the vertices of the primitives which are already transformed from the 3D scene space into 2D screen space. A fragment is a screen coordinate which potentially could be a pixel on the resulting 2D image. There can be many fragments per pixel which come from different primitives. The fragments can be discarded during a later step of the RP, because as example another fragment is in front of the fragment in respect to the camera location. Exactly one fragment per pixel is later chosen to be the pixel (see Figure 4.4 for an illustration). There are typically more fragments per pixel when more objects are located in the scene and more primitives overlap each other. Rasterization can be split up into two stages: the *triangle setup* in which for every primitive the necessary information is calculated. The second stage is the *triangle traversal* in which the fragments are calculated which are covered by the polygons. There are two common algorithms that are used for triangle traversal: The Pineda algorithm and the Bresenham algorithm.

2.2. Pineda Algorithm

Explanation of the Pineda algorithm

The Pineda algorithm is used to calculate all fragments that are inside of a triangle, given by the vertices of that triangle in 2D screen space and the edges connecting them. For every edge of the triangles a line equation is used that represents that edge. The line equations are derived in an implicit form.

This form has the property that for a given point that is put into the equation the sign of the result is determined by whether the point is on the left or right side of the line. When the line equations are derived in a correct way, this can be used to check for a point whether it is on the inner side of all line equations of a polygon. You can see an example of this in Fig. 2.2. In this way for every fragments that could be inside the polygon it is tested if it has the right sign on all equations or not. If it has a wrong sign on one of the equations it is discarded, if it has the right sign for all equations it is a fragment inside of the triangle. The number of possible fragments is previously reduced by only checking the fragments that are inside of the minimum and maximum x and y range of the vertices of the triangle. Still the algorithm checks more fragments than necessary and there are faster algorithms for this problem like the Bresenham algorithm. But the Pineda algorithm has a major advantage above other algorithms like Bresenham, because almost every step of the Pineda algorithm can be calculated or tested independent from every other calculation of the algorithm. All fragments can be tested independently from every other fragment. This means that the algorithm is very good for parallel computing which makes the algorithm an excellent choice for this problem [14].

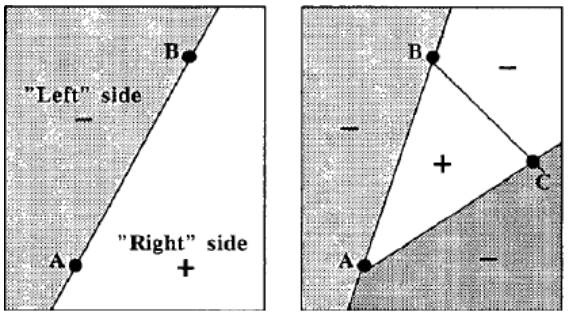

Figure 2.2.: a) "Subdivision of plane by line through points A and B" [14],
b) "Triangle formed by union of right sides of AB, BC and CA" [14]
"A Triangle Can be Formed by Combination of Edges" Source: [14]

2.3. Bresenham Algorithm

Explanation of Bresenham algorithm

The other popular algorithm used to calculate all fragments that are inside of a triangle is the Bresenham algorithm. The algorithm is a very fast algorithm that computes only with addition and without division or multiplication all the fragments that are covered by a line.
In order to make use of some concepts of the algorithm the given line has to be mapped from an arbitrary position to be in the first quadrant of the coordinate system with a slope between 0 and 1. Every line can be easily mapped to this format and after the calculations are done can easily be mapped back to the original form.
Lines in this format have the property that for iterating the integer points that lie on the line there are always only two possible options the next point can be: It is either to the right respectively east or to the upper right respectively north-east of the current position. This is the case because the slope of the line has to be between 0 and 1. This decision can on calculation side be simplified by adding a decision variable d of which the sign is determined by the east or north-east decision. The variable is incremented by specific values dependent on whether the last decision was east or north-east. With this all points that lie on the line can be enumerated by one query for the decision variable and one incrementation of the decision variable per loop [3].
This algorithm is then performed for every edge of the primitive which results in a primitive which borders are known, but the center is not filled with fragments. This filling of the polygons center can easily be done by going line per line trough the primitive and including every fragment that lies in between of two fragments that are on the lines.
The iterated incrementation of the decision variable in the Bresenham algorithm can be seen as a form of a Digital Differential Analysis (DDA) algorithm used with polynomials of degree 1 (lines). DDA algorithms work with polynomials of arbitrary degree and not only lines. Further details can be found in [8] but this work will concentrate on polynomials of degree 1 and only the Bresenham algorithm.

2.4. Virtual Reality

Introduction to the concept Virtual Reality and the used hardware

Virtual Reality (VR) is a concept in which a user interacts with an application using a human-computer-interface with which the user controls and experiences a simulated reality made by the application. The user can move inside of that reality and interact with it. The inputs are controlled in a different way than in a classical application depending on the VR device that is used. The used equipment for evaluation and implementation in this work is the *HTC Vive* [1], which consists of a headset the user puts on their head to see and hear the Virtual Reality, two base stations that track the players movement from outside and two controllers with multiple inputs (see Fig. 2.3). The controllers have laser pointers to interact with objects inside of the Virtual Reality. VR has great potential to be used in learning applications because the learner has more interactivity with the learning material and is more likely to remember what they have learned. This is the reason the RePiX VR application was focused on an implementation using VR [7].

Figure 2.3.: Image of a HTC vive equipment. Source: `https://www.vive.com/media/filer_public/vive/product/comparison/pro-2.png`

[1] HTC Vive: `https://www.vive.com/de/`

2.5. Unity

Basic concepts of Unity

Unity is a free software to develop applications and computer games, which is used for the implementation of the RePiX VR project. This section will introduce the main functionalities Unity has that are necessary to understand for reading this work.

In Unity gameobjects are located inside of a scene. The gameobjects are structured in a hierarchy that looks like in Fig. 2.4. The gameobjects can have components and C# scripts attached to them that determine their behaviour in the scene during runtime of the application.

Another concept that can be used in Unity are timelines and signals. When a gameobject gets a signal, different functionalities that are attached to this gameobject can be triggered. In combination with the timelines, predefined sequences of actions can be performed in the application. The timelines give, after they have started playing, signals to components that were defined in the timeline. This concept is used for the implementation of the stages in RePiX VR and the substages of the rasterization stage.

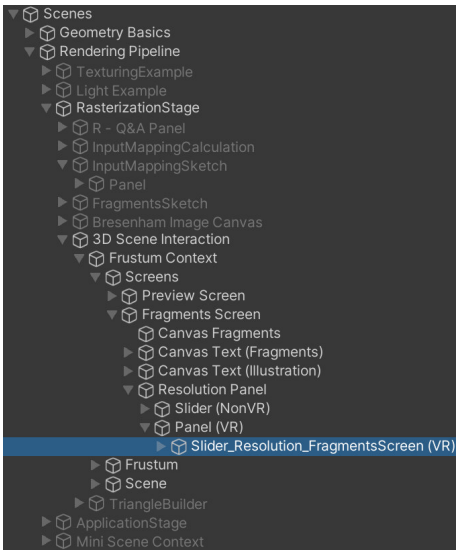

Figure 2.4.: Screenshot from the hierarchy of the gameobjects used in the rasterization stage in Unity.

Chapter 3 Related Work

This work is based on the project "Rendering Pipeline eXperience in VR" (short "RePiX VR") developed in the LTI-Lab of the *Learning Technologies Research Group* [1]. The project was introduced in S. Görzen's work *Technology-Enhanced Learning of Computer Graphics Rendering Pipeline in Virtual Reality* [7]. An application in Virtual Reality was created in which the learner can experience several stages of the Rendering Pipeline in VR, multiple stages offer a high possibility of interaction for the learner. The steps of the Rendering Pipeline are taken of a simplified model from the CG lecture *Basic Techniques in Computer Graphics* by Prof. Dr. Leif Kobbelt (see Fig. 2.1). The concepts of the Rendering Pipeline and rasterization are explained in Chapter 2 and can be looked up in [12]. Also the scientific background of the purpose and possibilities of Virtual Reality in learning environments is explained in S. Görzen's work in detail. In short one can say, that in regular a learner has better understanding and a higher chance of memorizing the given information the more interactive the learning environment is. This gives a high potential for VR as a learning environment. In S. Görzen's work he created different personas to make the development easier and have a better focus on different stakeholders that have interest in the project. The different personas are *pupil*, *student*, *teacher*, *researcher* and *developer* and were created with an online questionnaire. This work will focus on the personas *pupil* and *student*, because they are the targets who will be learning in the environment. The most relevant information about the personas is that both have good knowledge of computer science but less knowledge in CG. They are interested in Virtual Reality, but have only tried it once. For detailed information about the personas see [7].

The RePiX VR project

In the current implementation of the rasterization stage the user gets the main purpose of the rasterization step explained and what fragments are on one text-panel each. On the basis of one image it is shown that 3D geometry is projected to pixels of a 2D image. Furthermore the Digital Differential Analysis (DDA), Bresenham and Pineda algorithms are mentioned but not explained to the user. This implementation gives the user a brief overview of the rasterization step and what its purpose is, but it is not explained how the step is working like in the other steps of the learning application. Also the purpose or the way of working of the three algorithms is not mentioned further and thus gives minor information to the learner. Instead of only one picture it can be very beneficial for the understanding of the rasterization step to give an interactive visualization of the 3D geometry like in the other steps of the learning application. As stated in [9] a constructive and interactive approach has multiple advantages in learning theory explicitly suited for VR. Also for a deeper understanding of the used algorithms for the *student* persona the algorithms should be explained in detail.

Desciption of the old rasterization stage

[1] Learning Technologies Research Groups LTI Lab: `https://learntech.rwth-aachen.de/cms/LearnTech/Forschung/LTI-Lab/~kqtlz/RePiX-VR`

Next some other works with the purpose of giving information to interested learners about rasterization are specified and evaluated. There are only very few works which address teaching of the Rendering Pipeline (even fewer for rasterization explicitly) on a non university level (like lectures) which go way to far into detail and cannot be compared to a short gamified introduction to the topic which also focuses on teaching pupils. One work of the few is "Rasterization: a Practical Implementation" [15] which is a text-based introduction into the concept of rasterization and gives an explanation of the algorithm behind the concept. It focuses on the difference between ray-tracing and rasterization and goes into code-details of the algorithms that can be used for rasterization. Besides the rasterization step of the pipeline it also goes into detail about the other steps of the CG Rendering Pipeline. Since the explanations are mainly text-based and only supported by some images the reader needs a strong or intrinsic motivation to read trough the whole explanation. The target learners for this text are mainly university students or other people with high interest into the details of the concept, since it goes into high detail about the programming part of the algorithms. Thus the text would most likely not be addressed to the *pupil* persona unless they had great interest. Also the lesson gives brief summaries for the reader at important points which could be a good addition for the rasterization stage.

Other works related to teaching rasterization

Another interesting work is the online textbook "Graphics Programming Compendium" which concentrates on teaching the programming part of CG and the fundamentals that are necessary to understand this. In the first part it is concentrated on teaching the fundamentals for an implementation of a software rasterizer and thus the graphics pipeline and rasterization is explained shortly (cf. [6], ch. 1 f.). The focus of the work is the implementation part; the concept of rasterization is only explained rough. But the whole concept is explained in simple language and with examples so that even pupils of higher grades can understand the concept of the rasterizer. The simple language and use of examples is a great way for teaching this topic and may be a good addition to the rasterization level, even though graphical illustrations fit better into the VR environment than textual examples.

Algorithms used for rasterization

The different algorithms that can be used for implementation of rasterization and that are introduced in Chapter 2 are the Pineda algorithm which was firstly developed in [14] and the Bresenham algorithm at first mentioned in [3] which makes use of a concept of Digital Differential Analysis, which can be looked up in [8]. The algorithms are different approaches to efficiently solve the problem of getting all fragments inside a primitive.

Useful papers for design of teaching applications

The concepts of *Design-based Research* and *Iterative Design*, which are used for the development of the RePiX VR project can be looked up in the works "Design-based research: A decade of progress in education research?" [2] and "The VR Book: Human-Centered Design for Virtual Reality" [10] which are great sources for concepts of the design of applications in Virtual Reality.

The concept of student-centred teaching in comparison to teacher-centred teaching and the advantages, which is an interesting concept used in the new implementation of the rasterization stage is described in [13].

Another paper which concentrates on the design of learning applications and has many useful hints and concepts in it derived from interviews with experts on that field is [5].

The *System Usability Scale* (SUS) which is also used for the evaluation of this work was introduced in [4] and is a very general test introduced for software systems. It uses alternating positive and negative formulations of questions and calculates a score from this. How the SUS works and its history as well as an evaluation of its usefulness can be found in [11].

Part II

The new Stage

Chapter 4 Stage Implementation

Overview of the chapter

This is the main chapter of this work that contains all information about the newly implemented stage, the design and implementation details. The chapter is split into three sections. The first two sections deal with the content of the new stage, the structure and learning material contained in the new implementation. While the first section deals with the regular stage, the second section introduces an implementation that was only used for the evaluation of the stage.
The third section deals with the different design decisions and the implementation details regarding this new stage described in the previous sections.

4.1. Content of the Stage

Basic structure of the new stage

In this section the structure of the stage will be explained and it is expanded on, what the different explanations cover. The stage is divided into several parts and substages. The main stage which the user is entering when clicking on the stage's name on Kyle's navigation panel is a short introduction to the topic of rasterization.

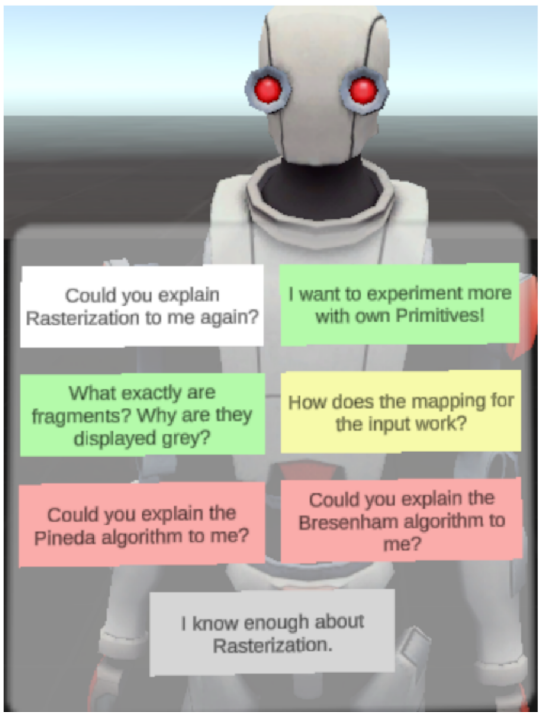

Figure 4.1.: Screenshot of the Q&A-Panel with the seven buttons to enter the different substages.

This is followed by a panel that appears which offers multiple buttons to press. This can be seen in Figure 4.1.
Each button provides a question the user can use to ask Kyle to a specific element of rasterization which was mentioned before in the main stage. The buttons have different colors, indicating the complexity of the topic the substage covers. Where green means that the explanations should be easy to understand even for novices of rasterization; yellow means that the stage covers unnecessary information that the user doesn't have to understand; red means that the information of this stage are complex. It is recommended for learners, who already know the other substages and the content is probably too complex for novices. The not-colorized buttons (First one and last one) have no colors because they cover no new information.

The seven buttons cover the following questions:

- "Could you explain rasterization to me again?"
- "I want to experiment more with own Primitives!"
- "What exactly are fragments? Why are they displayed grey?"
- "How does the mapping for the input work?"
- "Could you explain the Pineda algorithm to me?"
- "Could you explain the Bresenham algorithm to me?"
- "I know enough about rasterization."

When one of these seven buttons is clicked the user enters a substage which explains what the user has asked. If the user clicks on the first button, the user will see the same explanations of the main stage again.
The buttons and explanations are all available in English and German. Which of them is chosen depends on the language the user has picked at the beginning of the application.
The content of the seven stages will be explained in the following seven sections.

4.1.1. Main Stage

Content of the main stage and explanations

The main stage consists of five text panels with explanations from Kyle, one simple image for illustration and one complex more practical illustration. During the whole stage the simple image is shown to the left of the dialog panel to give the user something to visualize what they get explained. The image itself does not get explained or mentioned in this stage, but in the "Fragments explanation" stage (see Section 4.1.3). The image can be seen in Figure 4.4.
The first of the five texts explains the purpose of rasterization: "The **rasterization** detects which pixels on the screen are potentially covered by each primitive. Those potential pixels are called **fragments**. The primitives are **here** mostly simple triangles."
When the user has read the text and clicks on the text panel the next text is shown which describes the input of the stage as: "**Input** of rasterization are the **vertices** of a primitive. These are already mapped to the 2D coordinates on the screen."
The third text panel has explanations for the output of the rasterizer. Also during the explanation the more complex illustration gets shown to the left of the user and the illustration gets also explained on the text panel: "**Output** are all fragments that are covered by the primitive. You can see an **illustration on the left**: A frustum with several objects in it; When an primitive gets highlighted it is **rasterized**."
The following text panel goes into detail on the illustration: "The output can be seen on the **Fragments** screen. You can change the resolution of the Fragments screen. The actual 2D image can be seen on the left screen."
An image of the illustration can be seen in the following Figure 4.2

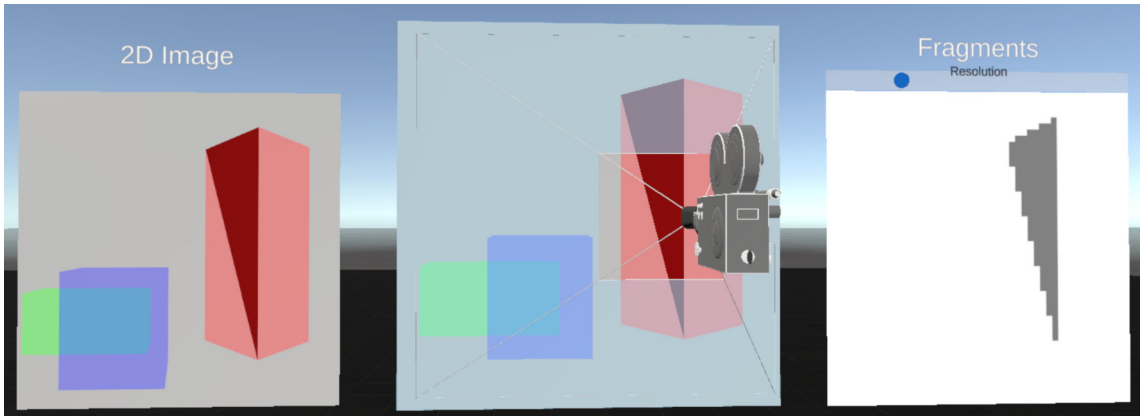

Figure 4.2.: Screenshot from the main stage. Middle: Frustum with objects in it made up from multiple triangles; One triangle is highlighted. Left: 2D image from camera perspective. Right: Screen shows the rasterized, highlighted triangle with current resolution of 50.

The illustration works as in the previous texts explained. The objects are made of multiple triangles which represent one primitive. When the player has switched to this fourth panel an animation starts playing where every 1.5 seconds a different primitive gets highlighted in a preset order. The highlighted primitive is rasterized and shown on the fragments-screen. The resolution can be changed by the user as said with the slider above the fragments-screen. The minimum resolution of the screen is 20x20 pixels and the maximum resolution is 500x500 pixels. The aspect ratio is always 1:1. The screen gets updated every 5ms if the resolution or primitive is changed.
During development another text panel got inserted at this point. It further points out a thing that had lead to confusion by some learners in the old rasterization stage implementation and that is already visible in the illustration: "As you can see **all primitives** are rasterized independent from their positions. In this step is **not checked** yet, whether a **primitive is visible or covert** by another primitive." With this the misunderstanding that only visible primitives are rasterized should be cleared out.
On the last text the algorithms used for the rasterization are mentioned, but not explained further: "Rasterization can use different algorithms like the **Pineda** algorithm or the **Bresenham** algorithm using **Digital Differential Analysis (DDA)**." The further explanations can be found in the substages available with the Q&A-panel.
When clicking on the last text the user is guided to the Q&A-panel on which the user can ask the predefined questions. If the user picks the first option to get rasterization explained again, the user will restart the main stage and get the same explanations again.

4.1.2. Input Mapping

Content of the input mapping substage

In this substage the mapping of the 3D vertices of the objects in the frustum to the 2D vertices in screen coordinates is explained shortly. Since in past stages the normalized device coordinates were already explained, the texts given focus on mapping the coordinates in the right range.
On the first text panel is described what the input mapping does, using the normalized device coordinates: "As we have seen in the clipping stage, we have **normalized Device Coordinates**. Input mapping calculates from these coordinates the coordinates on the screen."
The following text clarifies the step from 3D to 2D coordinates:

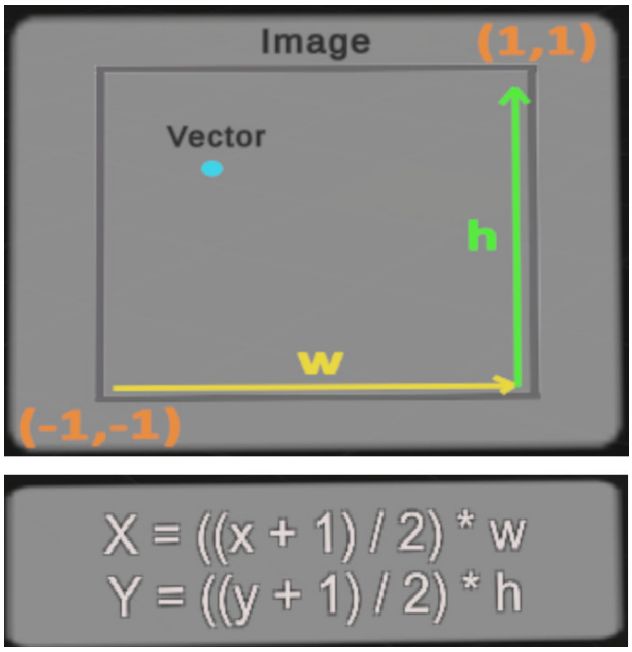

Figure 4.3.: Two screenshots from the input mapping substage. Top illustration shows screens width w, height h, the vector range from (-1,-1) to (1,1) and the vector that should be mapped to screen coordinates. Bottom image shows the calculations for the vector.

"The only thing we have to do is **dropping the z-coordinate** of the 3-dimensional vector to get the 2-dimensional vector. This vector has the position in the scene, that is **projected onto the image**."
The last two text panels concentrate on the explanation of the conversion between the normalized coordinates and the screen coordinates. "Now we have a vector in the **range of (-1,-1) to (1,1)**. But we need a vector in the range of the image resolution with **width w** and **height h**. The range should be from **(0,0) to (w,h)**." While this is brought to the learner on the left side an image is shown which illustrates the coordinates, width and height of the image. The image can be seen in Figure 4.3. At last the calculation is mentioned: "We can achieve this by a **simple calculation**. (See right.) Now we have the **coordinates on the screen** as we wanted." The calculation can be seen in Figure 4.3.

These calculations can be verified: We start at range $(-1,-1)$ to $(1,1)$, which means for $x : (-1)$ to (1) and for $y : (-1)$ to (1).

$$x : (((-1 \text{ to } 1) + 1)/2) * w = ((0 \text{ to } 2)/2) * w = (0 \text{ to } 1) * w = (0 \text{ to } w)$$
$$y : (((-1 \text{ to } 1) + 1)/2) * h = ((0 \text{ to } 2)/2) * h = (0 \text{ to } 1) * h = (0 \text{ to } h)$$

This results in a conversion of the range $(-1,-1)$ to $(1,1)$ to the range $(0,0)$ to (w,h)
When the user clicks on the last text user is lead back to the Q&A-panel.

4.1.3. Fragments Explanation

Content of the fragments substage

In this substage the user gets explained what exactly the difference between pixels and fragments is. The stage consists of six texts by Kyle and the simple illustration that was already visible during the main stage and will now get explained further in this substage. The first text is a simple introduction to fragments: "A fragment is a **potential pixel**. A fragment can be discarded or become a pixel itself." followed by the next text which explains that there are multiple fragments per pixel: "There can be **multiple fragments per pixel**. Only one of them will be chosen to be the pixel in later steps of the **Rendering Pipeline**." The succeeding two texts from Kyle go into detail about the simple illustration, which can also be seen in following Figure 4.4.
The explanations on the text panels are: "You can see an **example** of this in the picture on the left. Given is a scene in top-down view in which every object has one color and consists of just **one primitive**."

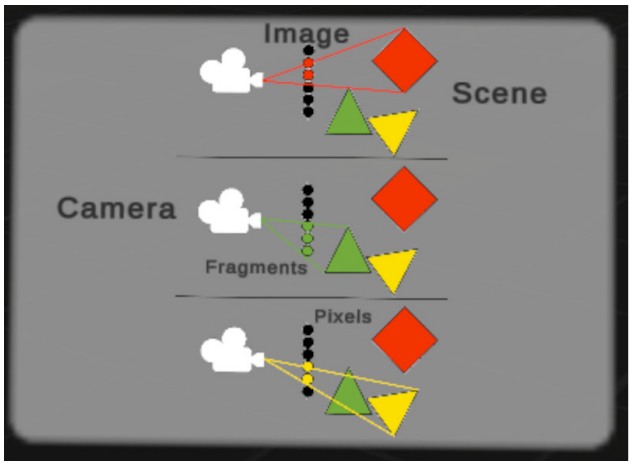

Figure 4.4.: Screenshot from the fragments explanation substage. Illustration shows rasterization of different primitives in a simplified form.

followed by "The scene is displayed three times and in each scene a different primitive is **rasterized**. It is shown for each primitive, which pixels correspond to it, and therefor which fragments are calculated."
Next on the fifth text panel the purpose of fragments is explained using the example: "You can see, that the **fragments** of the green and yellow primitive are **overlapping**. For each pixel only **one** of them will later be chosen to be the pixel, but each of them is one fragment."
The last text explains why on the fragments-screen (used in other substages) the fragments are always displayed grey:
"In the other illustrations of the rasterization stage, **all fragments are displayed in grey**, because the **colors are not determined** in this step. I have displayed them without color so that you don't get confused with this."
When the user clicks on the last text the simple illustration disappears and the user is lead back to the Q&A-panel.

4.1.4. Bresenham Algorithm

Content of the bresenham substage

In this substage is explained to the user how the Bresenham algorithm works and what the advantages of this algorithm are. These explanations are probably the most complex of the substages and some information are left out that are not really necessary for the learner to know, because it would make the substage even more complex. The explanations use as illustration an animation that works similar to the illustration of the Pineda algorithm substage; A screen gets shown left behind of Kyle where the described things get illustrated.
Kyle's first text describes the input and purpose of the Bresenham algorithm: "The algorithm gets as **input** the **three vertices** of the triangle and calculates all pixels that represent the **lines between the vertices**." On the screen, that is located at the same spot as in the pineda substage (see Fig. 4.6), three green vertices forming a triangle in a grid are shown. You can see an image of this in a) of Figure 4.5 .
On the following text panel an important concept is explained shortly that is needed for all of the calculations: "Every line can be mapped to a line which has a **slope between 0 and 1**. These mapped lines are used for the calculations and later **mapped back** to their normal form." On the screen two lines are shown: One in black that has a slope of $m = 1$ and one in green that has a slope between 0 and 1. You can see this in b) of Figure 4.5 .
The succeeding two explanations offer a simple solution for the problem and it is explained that this slow solution wouldn't be sufficient and that a faster solution in the form of the Bresenham algorithm is necessary: "You could now **enumerate all points** on this line by simple calculations using multiplication and rounding of values. This would work, but would be **very inefficient** and can be done faster. " and "Since there can be millions of primitives it is **very important** to do these calculations **as fast as possible**.

Bresenham algorithm achieves the enumeration of the points with **only one loop** and **one check plus one addition** per loop, which is very efficient."

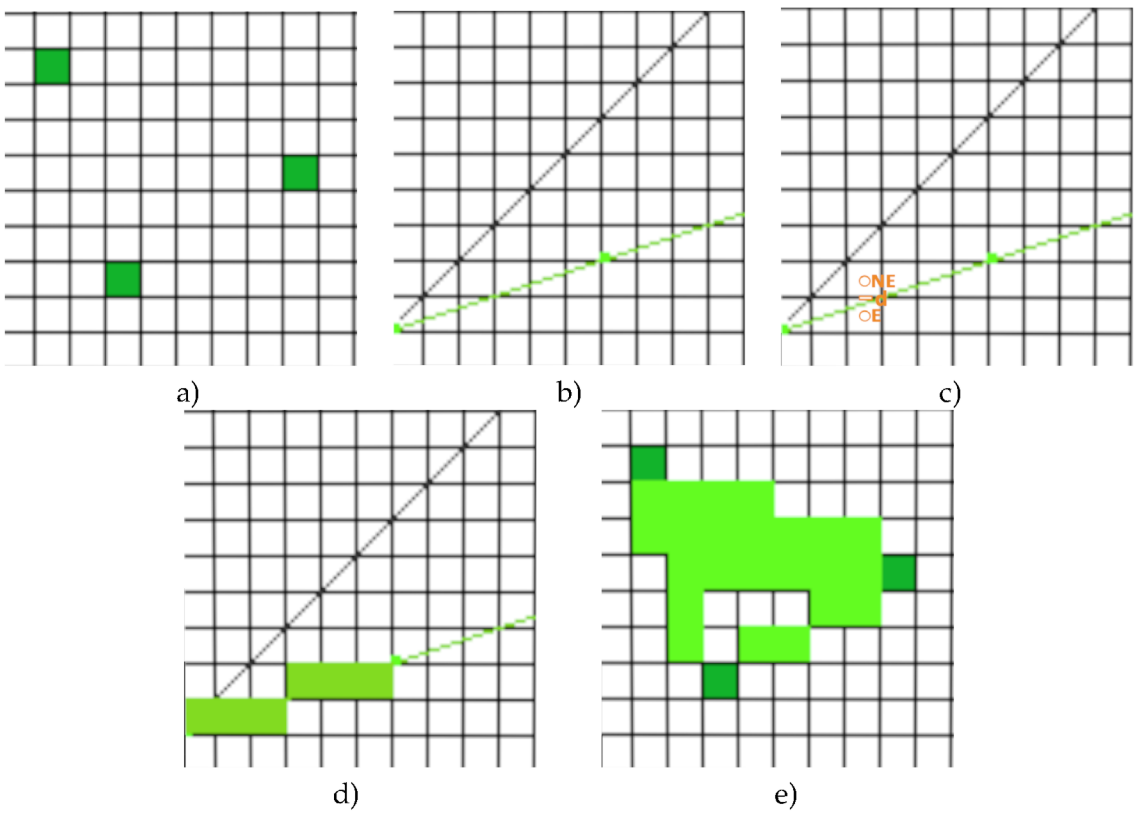

Figure 4.5.: Screenshots from the bresenham substage. a) The three vertices inside a grid. b) Black: Line with slope of 1; Green: mapped line between two green vertices. c) Pixels get marked as E/NE; decision variable d is visualized. d) Final state of the animation with correct pixels. e) Triangle gets filled line per line.

No new elements are added to the illustrations at this point.
The succeeding text panel describes how the calculations of the algorithm work without going into to much detail: "The main idea is, that with these mapped lines in each step it only has to be decided whether the **next point is to the right or upper right** of the current position. The decision is: "**Is my line nearer at east (E) or north-east (NE)?**"" On the illustration three things get displayed: Two points that are labeled with *E* and *NE* and a dash between them that is labeled with *d*. An image of this illustration can be seen in c) of Figure 4.5 .
The decision variable *d* gets introduced next: "This decision can be even **more simplified** by adding specific values to **the decision variable d** in each loop. The remaining decision then is: "**Is d less than 0 or not?**", which results in E or NE. " While this text is shown, an animation starts playing on the screen behind Kyle. The decision variable *d* and the points *E* and *NE* get updated and the corresponding pixel that results from the Bresenham algorithm is colorized. An image of the final state of the animation can be seen in d) of Figure 4.5 .
On the following text panel the connection between the Bresenham algorithm and Digital Differential Analysis are mentioned: "This incrementation of the decision variable is part of **Digital Differential Analysis (DDA)**, which can be used for arbitrary polynomials and not only lines as it is used for the Bresenham algorithm." The animation keeps playing while this text is shown.

The last of Kyle's explanations cover the output of Bresenham algorithm and the last step that is not a part of the algorithm itself: "Now we have the **three lines rasterized** and we only have to **fill in the empty triangle**. This can easily be done by filling all pixels per row, that lay in **between the first pixel** on a line and the **second pixel** on a line." On the screen a triangle is shown that gets filled line per line as you can see in e) of Figure 4.5 .

When the user clicks on the last text the illustrations disappear and the user is lead back to the Q&A-panel.

4.1.5. Pineda Algorithm

Content of the pineda substage

In this substage the Pineda algorithm is explained to the learner using an animation that is shown to the left of the user on a 2D image screen. The image gets updated with every step of the algorithm that is explained to the user.

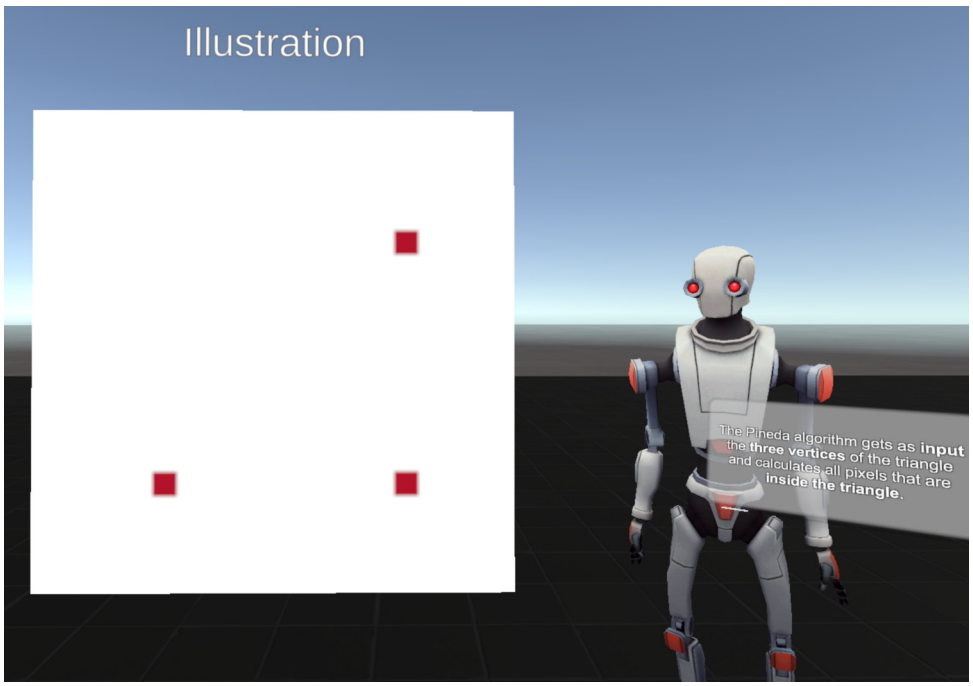

Figure 4.6.: Screenshot from the pineda substage. Shows location of the animation screen in respect to Kyle's position; On the screen: the three vertices of the triangle that are input for the Pineda algorithm.

On the first text panel the input and purpose of the algorithm is mentioned: "The Pineda algorithm gets as **input** the **three vertices** of the triangle and calculates all pixels that are **inside the triangle**." On the screen to the left is a white screen with three red dots shown that represent the input triangle. You can see an image of this in Figure 4.6.

The next step of the algorithm is the calculation of the line equations which is described on the succeeding text panel: "At first the **implicit** equations of the **three lines** that connect the vertices are calculated. You can see the lines and points of one example in the illustration on the left." The line equations of the examples are also shown in the illustration as you can see in a) of Figure 4.7.

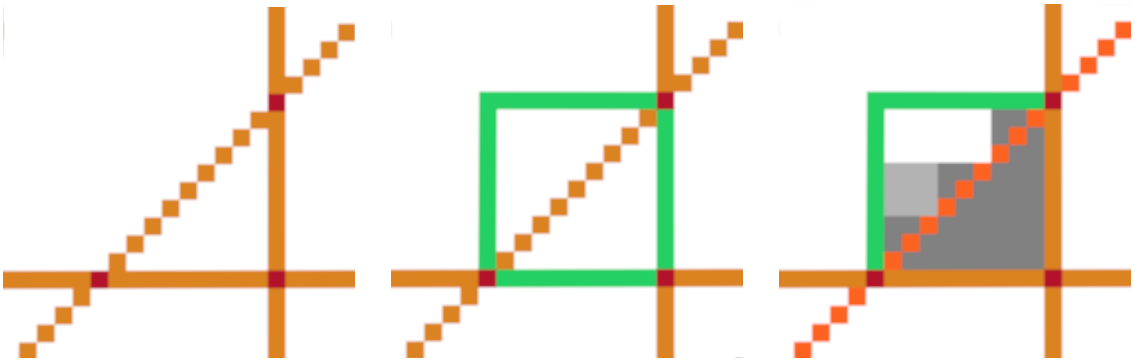

Figure 4.7.: Screenshots from the pineda substage. a) The three line equations that are build from the three vertices. b) The minimum and maximum range the pixels are tested in. c) One state of the animation: 7 pixels were already tested, 8th pixel is tested in respect to the diagonal equation.

The purpose and advantage of these implicit line equations is clarified next: "These equations have the property, that for a **given point** you only need **one simple query** to know on which **side of the line** the point is."
The next step of the algorithm is described as "Now the **minimum** x and y coordinates and the **maximum** x and y coordinates are determined. Inside of these **borders** as you can see on the left, **all pixels will be checked** on which sides of the three lines they are."
The borders are illustrated on the screen in the 2D image as you can see in b) of Figure 4.7.
The last step of the Pineda algorithm is explained on the following text panel: "A pixel will be detected as **inside the triangle** if it is on the **inner side** for all of the **three line equations**. If it is on the **wrong side** of one equation, it is **discarded**." At this point an animation starts playing on the screen. Every 2.5 seconds another pixel gets added to the image. In the animation the pixel gets tested which is visualized by highlighting the three line equations one after another. When the pixel is inside the triangle the color is changed to a darker grey and the next pixel is tested. If the pixel is outside of the triangle is is changed back to white. The animation loops forever. One state of the animation can be seen in c) of Figure 4.7.
Next the animation gets described shortly with: "You can see an animation of **one triangle rasterization** on the left. There are **9 pixels** in a 3 by 3 pixel square. The vertices of the triangle are (0,0), (3,0) and (0,3)."
On the last text panel the advantage of the Pineda algorithm is explained: "The pixels can be tested **independent** from each other. So the order of the tests is not important and the **tests** can also be done **in parallel**. This makes the Pineda algorithm very **efficient for parallel computing**." The animation on the screen is still looping until the end of the substage.
When the user clicks on the last text the screen with the animation disappears and the user is lead back to the Q&A-panel.

4.1.6. Playground

In this substage the user gets the possibility to interact more deeply with the illustration in the frustum, create primitives themself and see the animation shown in the main stage again. The text panels have no new information for the learner, but explain how to interact with the objects and how the mechanics work.

On the first text panel the purpose of the stage gets introduced: "Here you can experiment with **building your own triangles**. You can observe what your **own triangles look like when rasterized** on the fragments screen. Also you can see on the left the animation from before again." As said by Kyle in the text, to the left of the learner the same illustration as in the main stage is shown. The animation in which every 1.5 seconds a different primitive is highlighted is started.
When the user navigates to the following text panel the animation stops playing. The text is: "Now you are also able to **select the highlighted and rasterized triangle yourself!** You just have to **point** with the laser pointer on the **triangle** you want to see rasterized and **select** it." As described the user can hover over one of the existing primitives and select it to rasterize it.

Content of the playground substage

In the next part of the stage the user is able to create the triangles using the same triangle builder that was already used in a previous stage of RePiX VR: "The triangle builder **works just as you have learned** in the geometry step of the pipeline. You can at **first place vertices** wherever you want and **then build triangles** by selecting these vertices."
The explanations go even further on the succeeding text panel: "You need to switch to the **respective mode** when you want to **place vertices** or **build triangles**. Note that **only triangles** that are **inside the frustum** will be visible when rasterized!"
During these text panels the user is already able to create the triangles. Another text helps the learner if they have trouble with the controls of the menu panel: "If you have **trouble switching the modes**, just try to **teleport somewhere** with less objects and try again!"
On the second last text panel the learner is invited to start building the triangles: "**Now you can experiment with it!** When you are done with experimenting, let me know and the animation from before will be shown a last time."

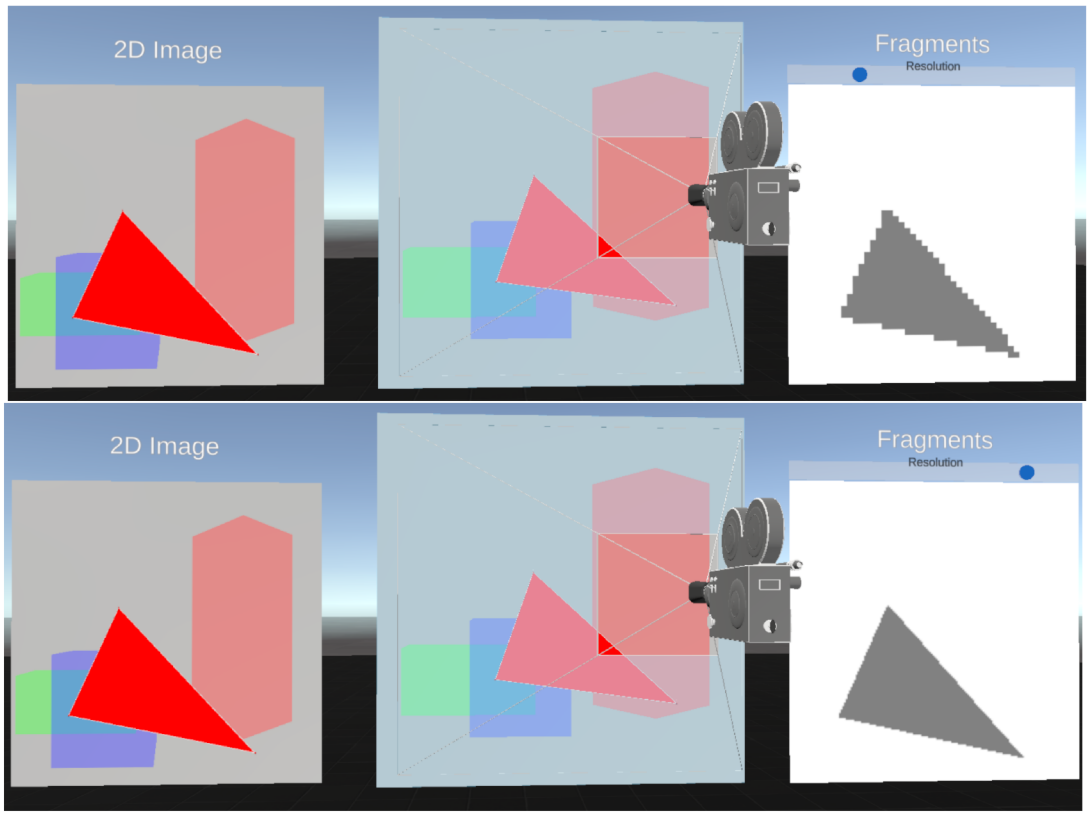

Figure 4.8.: Screenshots from the playground substage. Both show a user-made triangle rasterized on the fragments screen. a) resolution of 50; b) resolution of 180.

When the learner creates triangles they are able to hover over them and select them just as the pre-build primitives in order to see on the fragments screen, what the result would look like. You can see an image of an example triangle build with the triangle builder and rasterized using two different resolutions in Figure 4.8.

While showing the last of Kyle's texts the same animation as on the first text panel and as in the main stage is playing: "Here you can see the **animation** with the pre-build triangles again. I hope you had **fun experimenting** in this stage and that you can understand rasterization better now!"

When the user clicks on the last text the illustrations disappear and the user is lead back to the Q&A-panel.

4.1.7. Next Stage / Summary

Content of the summary substage

When the user clicks on the button with the text "I know enough about rasterization." they enter a substage in which a summary of all the previous stages of the Rendering Pipeline is given. This summary was already part of the RePiX VR project before the implementation of the new rasterization stage and thus was added at the end of this new stage. After the learner has finished the summary they are able to navigate to the next stage of the Rendering Pipeline and the RePiX VR learning application.

4.2. Introduction Stage for Evaluation

Reasons for introducing another stage and content of that stage

This subsection will describe the content of the introduction-stage that was implemented for the evaluation Chapter 5. The rasterization stage can not be tested by new learners without also completing the other stages of RePiX VR or any preknowledge of the Rendering Pipeline, because some aspects and explanations of the new stage refer to previous stages of the Rendering Pipeline. Since the testing of the stage would have been a very long task when the learners also would have to complete all the other stages of RePiX VR and the learners concentration of the evaluation would not focus on the rasterization stage alone, but also the other stages, another small substage was implemented that gives a small introduction to the Rendering Pipeline. This introduction covers all aspects the learner has to know about the Rendering Pipeline in order to understand the rasterization stage. The testers for the evaluation completed the short tutorial sequence that explains the controls of RePiX VR, then they completed the short introduction and after that they were lead to the new rasterization stage. The content of this introduction substage is explained in the following.

The first of Kyle's texts introduce the Rendering Pipeline and its purpose to the tester: "Rasterization is one step of the **Rendering Pipeline**. I will tell you now what you need to know about the Rendering Pipeline in order to understand rasterization."; followed by: "The **purpose** of the Rendering Pipeline is to **create from** an application with a **3D Scene with 3D objects** in it a **2D image** on the screen."

The next panel introduces the term primitive and that 3D objects are made of them, which is an important aspect the learner needs to know before going into the rasterization stage in order to understand the main illustrations: "The 3D objects are made up from many small so called **primitives** which are in most cases **simple triangles**."

Another important part of the illustrations in the main stage is the frustum and the camera which the learner need to at least understand what they are used for: "The scene is **viewed from a perspective** which is represented by a **virtual camera**. In front of the camera a so called **frustum** is created (illustration to your left). Everything **inside of the frustum can be seen** from the camera's perspective."
As written in the text a part of an illustration from a previous stage of the rendering pipeline (namely the clipping stage) is shown to the tester. You can see a complete picture of the whole illustration that gets shown in Figure 4.9.

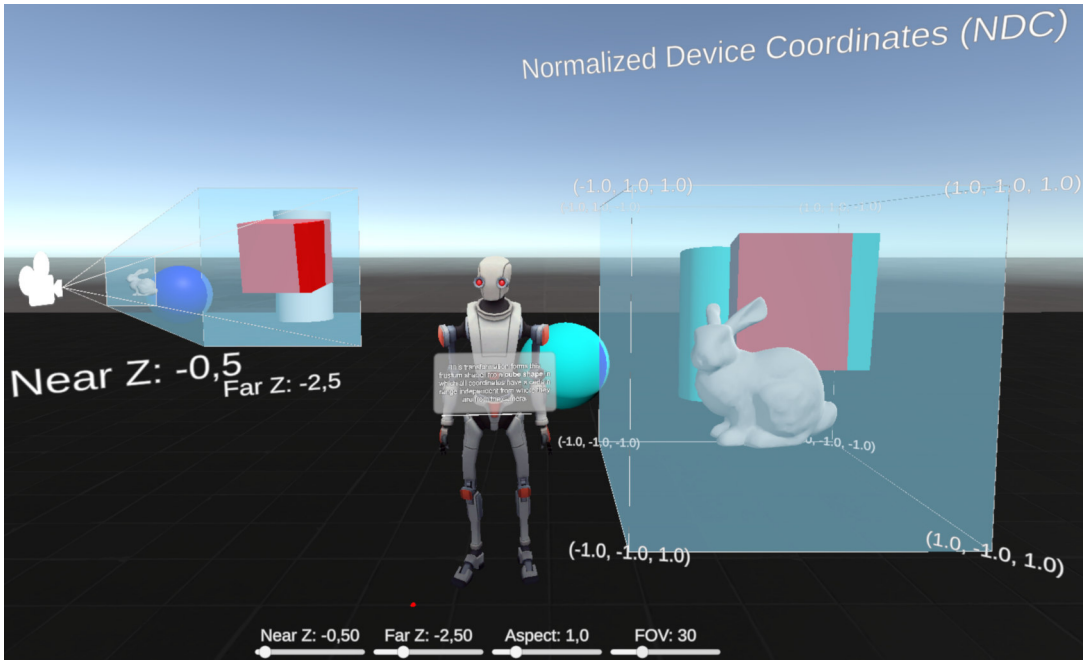

Figure 4.9.: Screenshot from the introduction stage. Kyle with his text is in the middle. Frustum with objects in it on the left. Unit cube with objects in Normalized Device Coordinates in the right. Non-VR controls at the bottom of the screen.

The learner is able to interact with the illustration. This interaction was already implemented in the previous RePiX VR stage and is only reused in this introduction substage. The illustration gets shown completely when the next two panels get shown which cover the explanations of the Normalized Device Coordinates which are an important concept for the Input Mapping substage (see Chapter 4.1.2): "The **shape** of the frustum depends on **different parameters**. In one step of the Rendering Pipeline all coordinates in the frustum are transformed to so called **Normalized Device Coordinates**. "; followed by: "This transformation forms this frustum shape into a **cube shape** in which all coordinates have a certain range independent from where they are from the camera."
On the last of Kyle's text panels in the introduction substage the tester is reminded that this was only an introduction for the real rasterization stage and that the introduction is now over in order to tell the tester that the important part of the implementation that should be evaluated now starts: "Now you know enough about the Rendering Pipeline to start learning about rasterization! I hope you will learn much about rasterization, have fun!"
When the tester clicks on that last text they enter the rasterization stage just as they would in the regular RePiX VR application.

4.3. Implementation and Design

Implementation details and design decisions regarding the stage

This section will go into detail about the implementation and the design of the different components and illustrations that were added to the stage in Unity. Every feature and element that was implemented can be used in VR which is the main focus of the RePiX VR learning application but they can also be used via mouse in the desktop version when there is no VR-headset connected. Overall the design of the stage is chosen to have a continuous design throughout the whole RePiX VR application and thus many components are reused from other already existing and implemented components. Despite the components are focused on reuse, all objects and scripts in Unity that are used for this stage are unique and are not interfering with the implementation of the other stages. In order to achieve this the objects and scripts of the reused components were copied and then changed for the purpose of the rasterization stage and not the originals got changed. This decision was made in order to simplify the merge between the new stage implementation and the code of the old application. No changes to the reused components have impact on the other stages of the Rendering Pipeline. In the next subsections details of the design decisions and implementations of the different components and illustrations used in the Rasterization stage are given.

4.3.1. Multiple Stages

Reasons for implementing the feature of multiple substages

In this section the design decision to have not one linear stage, but multiple substages and its implementation is explained further. The other stages of RePiX VR have (at this state of the application) all a very linear setting:
Kyle explains something, you click on the text panel and the next explanation comes up and you may also be able to have interactive tasks in between. In most stages this design makes sense because in the other steps of the Rendering Pipeline the knowledge needed to understand the step is also more "linear" than for rasterization.
That should mean that you need preknowledge to understand the full concept of the step and this has to be introduced before going into the step itself. The only non linear decision the user can make is the selection of the stages of the Rendering Pipeline which can be chosen on the panel beneath Kyle's texts (see Fig. 4.10), but even this option to choose from the stages is linear because the new stages get unlocked after completing the previous stage. But this linear structure has some disadvantages, mainly inside of the stages of the Rendering Pipeline. One of them is the feature of the progress bar below of Kyle's texts (see Fig. 4.11).

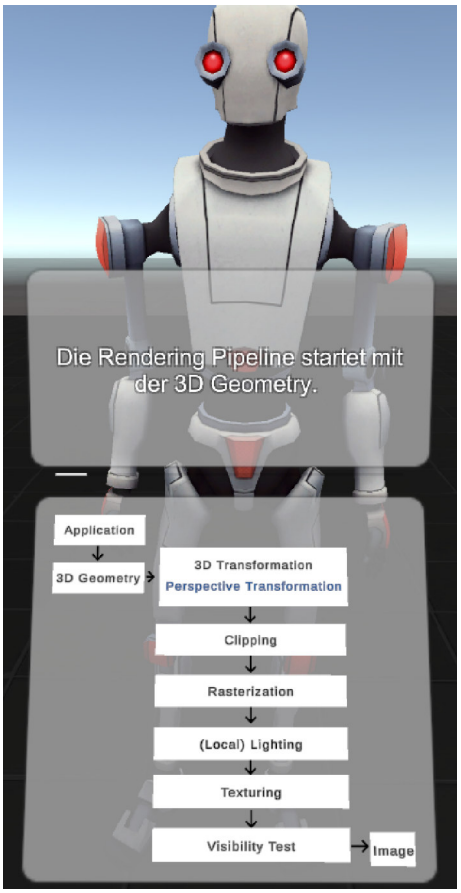

Figure 4.10.: Screenshot from the transformation stage of RePiX VR. Kyle and his text panel are seen in the upper half. The simplified model panel with options to switch between the Rendering Pipeline stages in the bottom half.

The progress bar itself is a very helpful feature added to the RePiX VR project that helps the user to see how much progress they have made throughout the stage and to give an overview on how much content is left in this stage. But this progress bar, which purpose is to motivate the learner, has a major disadvantage when the stage is very long.
In some new implementations of the stages of the Rendering Pipeline which are about to get implemented into the project at this state of writing, the explanations are very long and cover a huge amount of text panels with information. This leads to the problem that the progress bar that is updated on every text panel only increases very slightly for every information learned. This can be very frustrating to see that there is still very much information and explanation left in a stage that the user already has learned a ton of new information in.
For rasterization the linearity is not needed and it is possible to split the information into different parts. With this design there are two major advantages. The first one is that in each stage the progress bar is way shorter than the progress bar over all text panels would be:

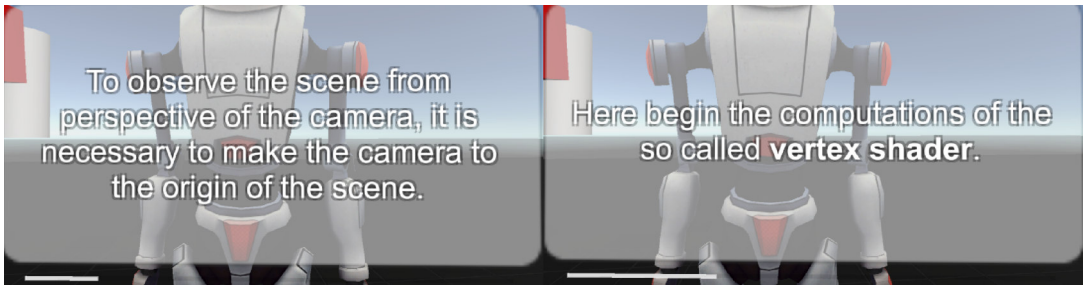

Figure 4.11.: Two screenshots from the transformation stage of RePiX VR at different states of the stage. Right screenshot is further advanced in the stage. The progress bar can be seen below Kyle's text panel. The right progress bar is further than the left.

The explanations (exclusive the introduction made for the evaluation) cover 46 text panels, so if all explanations would be put in linear order the progress bar would move roughly 2% on every text panel. This can be very exhausting for the learner, which is avoided by making multiple substages, in which in this implementation the longest substage is the playground substage with 7 text panels. The second advantage is that the learner can choose which information they want and need to learn and which explanations are to advanced or unnecessary to know for the user at the moment. Also the stage can be revisited at a later time in order to explicitly learn new parts of the rasterization stage they now would like to know about.
Overall one can say that the major advantage of this approach of multiple substages is that the information is split into pieces that are easier to learn part for part and the learner can choose the information they would like to learn in order to skip unnecessary or to deep information for the learner.

Q & A Panel

It was already derived in the previous section why it was chosen to make multiple substages instead of one linear stage. In this subsection the concept and design of the so called Q&A panel that is used for the navigation between the substages is described. The Q&A-Panel can be seen in Figure 4.12.

In [5] multiple requirements for designing a technology enhanced learning application were derived by qualitative interviews with experts on the field. One of these requirements was to provide assistance during learning and one mentioned solution for that is " implementing an “automated e-assistant, which is already programmed to answer probable questions that might come from the participants” " ([5], p. 6).

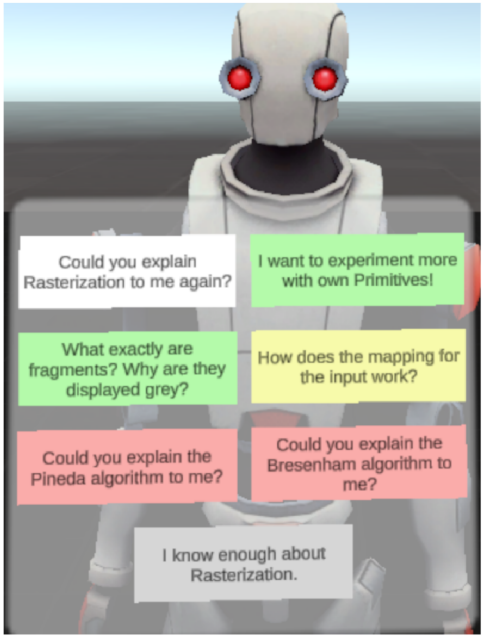

Figure 4.12.: Screenshot of the Q&A-Panel with the seven colorized buttons to enter the different substages.

This concept is used at the end of the main explanation of rasterization, where Kyle offers multiple options of questions the learner could have and which, when selected, guide to the corresponding substage with explanations to answer this question. This has multiple advantages as stated in the previous section: the learner has more interactivity, it is less linear and the learner has to decide themselves on what topic they want to learn more details.
The feeling of an interactive learning environment is also enhanced by this, because the user decides which information Kyle gives them next. The design decision to place the Q&A panel at the same position Kyle’s text panel normally is was made to create the impression that the user is talking to Kyle just as Kyle talks to them. This brings the whole application on another level where the user doesn’t just listen to a monologue from Kyle but is able to ask question and thus move to a dialogue form of communication.
This makes more interaction possible and improves the learning experience. That aspect is roughly comparable to a shift between teacher-centred teaching and student-centred teaching, which gives the student more power in learning and has multiple advantages for the learning experience [13].

Reasons for implementing and designing the Q&A panel

Another design decision regarding the Q&A panel that was made during implementation is that after completing a substage that button is grayed out and disabled so the user can not enter the substage again. It is only possible to look at a substage again if the user decides to restart the whole stage by either clicking on the rasterization button on the simplified model panel where the user can enter all stages of the Rendering Pipeline or by clicking on the white colored button that explains the main stage of rasterization again to the user. This disabling of the buttons was introduced because of some issues in non-VR that was caused when the buttons would be available even after completing the substage. But this decision also has some advantages for the learning experience and thus this design was maintained during development. One advantage is that the learner is able to see which substages and questions they have already completed and which are still left. This gives a better overview to the learner. And the second advantage is that the learner has some incentive to look at all substages because they probably want to see all parts of the stage just as players in video games may want to clear a game to 100%. This gives the same incentive to the player and may motivate them to look at all substages.
Another design decision that was made during development is to give the buttons, that have the different questions for the substages, different colors. You can see the colorized buttons again in Fig. 4.12. The colors should give the learner a hint on how complex

the information are that Kyle will give during the substage the button refers to. The white button indicates that there are no new information because it is just a repetition of what was already explained. The grey button should indicate that it is used to leave the rasterization stage. The green buttons lead to substages that have new information in it, but mainly focus on deepening the understanding of details of already explained concepts of rasterization. The yellow button indicates that the introduced information in this substage are new to the learner. And finally the red buttons should indicate that the given information in these substages are new and complex and thus the learner doesn't have to deal with them if they don't want to or they doesn't have to fully understand the information on the first visit of the substage.
This colorizing of the buttons is not explained to the user by Kyle at all so the user has to understand this relation between the color of the buttons and their difficulty of the substage them self. A possible advancement of the feature could be that Kyle gives a hint via audio to explain this relation, if Kyle gets audio lines as planed in the current state of RePiX VR.
The Q&A panel with the design of the possible questions also has advantages for the *developer* and *researcher* personas. For the developer it is quite easy to add more information to the stage when regularly questions regarding the stage are coming up from the learners. The developer doesn't have to decide where to put the information inside of the stage, they can just put it as a substage with the regarding question on the Q&A panel.
For the researcher it offers interesting data to track, which substage was visited the most or visited less to know what topics the learners have more or less interest in, or which explanations could need a refactoring.
The Q&A panel was implemented by reusing the simplified model panel (see Fig. 4.10), relocating and relabeling the buttons. Also the buttons functionality was adjusted so that they would start the corresponding substages timeline instead of the timeline of a stage of the Rendering Pipeline as they buttons of the simplified model panel do. Since the substages have timelines and work just as the stages of the Rendering Pipeline the concept of the simplified model panel could be reused in order to implement the Q&A panel.

4.3.2. Fragment Calculation

Functionality and implementation details regarding the calculations of the fragments screen

This subsection will describe the implementation details regarding the main interactive component, the fragments screen, which is used in the main stage and the playground substage of the new stage. The functionality of the fragments screen can on design and on implementation side be separated into three separate parts: The inputs, the calculation and the display of the fragments. On implementation side these parts are implemented by multiple simple scripts on the input side, the *Ra_Fragments_Calculation.cs* script for the calculation and the *Ra_Image_Controller.cs* script for the display. In the following subsections the different design decisions and implementation details regarding these parts are given.

Inputs

Implementation of the inputs

There are two inputs the user can give in the rasterization stage in order to change the result represented on the fragments screen (see Fig. 4.13).

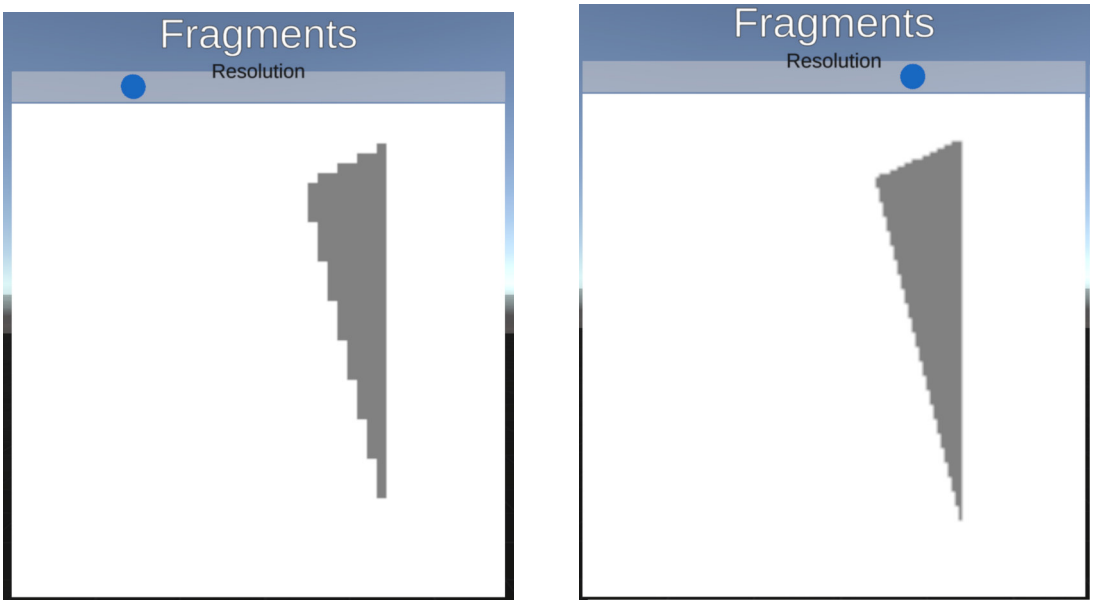

Figure 4.13.: Two screenshots of the fragments screen with different resolutions. Resolution slider can be seen on top of the fragments screen.

The one input which can be used in the playground substage by the user and is automatically used by the timeline of the stage otherwise, is the input of the triangle that should be rasterized. In the playground substage the learner is able to point with the laser pointer on a triangle and select it in order to start the rasterization and look at the result on the fragments screen. With this input option the user can determine theirself, which triangle should be rasterized. This is way more interactive than just looking on the screen showing the predetermined order of triangles that are rasterized and should invite the user to play around with the triangles and understand the rasterization of these triangles more. This input is possible by adding three main components to the triangles, which were created by placing gameobjects as vertices in the scene and creating meshes in form of triangles with these vertices. The three main components are the *Box Collider* which can determine if something is interacting with the hit box of that box collider that is attached to the triangle; in interplay with the reused *Pointable* script which was implemented already in RePiX VR in order to make interaction with the laser pointer of the VR controls possible, it can be checked if the user points with the laser pointer at the triangle. The third component is the *Ra_Triangle* script which is attached to each triangle and is responsible for the calculations regarding the creation of the mesh in order for the triangle to be visible and controlling the material which is attached to the gameobject and determines the look of the triangle.
The second input option the learner can use is the slider for the resolution of the fragments screen. This slider is positioned at the top of the fragments screen and the user can point on the slider and drag it from the left of the panel to the right of the panel in order to change the resolution of the fragments screen below in real time (see Fig. 4.13). The minimum resolution is 10 by 10 pixels and the maximum resolution is 200 by 200 pixels. These values were chosen because the range of the values is so big that a great difference in the picture is visible, with a very smooth picture on the max resolution and a very rough representation of the triangle on the min resolution. Also with a higher max resolution the proportion of the slider that is used for the small resolutions would get less that the high resolutions, but the smaller resolution values are probably more interesting for the learner than the high resolutions because the difference between the images gets

smaller for higher resolutions. The slider is positioned at the top op the screen in order to directly show the relation between the slider and the screen to the user and with this the user is always able to directly see the impact of a change of the slider on the screen. The slider is implemented by reusing and adjusting the *Slider Interaction* and the *(...) Slider* scripts that were already used in another stage of the RePiX VR project. Since these scripts only work for VR an unity slider component was used for the case that the application is used in non-VR mode and another script was implemented, namely *Ra_Resolution_FragmentsScreen* in order to control which of the slider components (VR or nonVR) is used. The value for the resolution that is later used in the calculation process is gathered from this script.
Both inputs, the triangle input and the resolution input, are invoking an update function on the calculation side whenever an input is made by the user or by the timeline.

Calculation

Implementationof the calculations

The calculations are done in the *Ra_Fragment_Calculation.cs* script which is attached to the fragments image and follow the concept of the rasterization step of the Rendering Pipeline. The calculations for the resulting fragments from the input primitive and the resolution of the image work using the pineda algorithm. Whenever one of the two update methods, either *invokeUpdate()* from the resolution panel or *TriangleInput(Ra_Triangle triangle)* for an update of a primitive, the image shown on the fragments screen is recalculated. The first step of the calculation is to get the current resolution from the resolution panel. After that the *InputMapping(...)* method is called which maps the given 3D vertices of the current primitive to the 2D coordinates on the screen. If this mapping was successful the 2D coordinates are given to the implementation of the pineda algorithm (see Listing 4.2), which works just as explained in Section 2.2 of Chapter 2. The function constructs an array with all pixels that are inside the triangle. A struct *Equation* is used in the script to represent the line equations used for the calculations of the pineda algorithm, which simplify the check of the points on which side of the line they are (see Listing 4.1).
At last the *updateImage(Vector2Int[] image, int resolution)* function of the *Ra_ImageController.cs* script is called which is responsible for displaying the resulting image on the fragments screen.

Display

Implementation of the display of the image

The third part of the fragments screen illustration is the canvas which is attached to the fragments screen and is used with the help of the *Ra_ImageController.cs* script to display the given image on the screen. The screen is working in the way that the script gets an array of 2D vectors which represents the pixels to be set as input. Then a new texture is derived from that and that texture is shown on the canvas gameobject. The calculations inside of the script are rather simple and mainly clear the texture when getting an input and then setting all pixels represented by the input array to a grey color. Then the canvas texture is updated which results in the input image shown on the fragments screen. If the resolution of the image is changed, the texture is resized. The script also contains some functions to make working with the canvas easier, for example scaling the image up or flipping the image at the x axis, etc.

Listing 4.1: Code of the line equation struct

```
// This struct is used for the pineda algorithm line equations
// The check method returns true or false if a given point is on the left
    or right side of the line equation
private struct Equation
{
    private Vector2 normalVector;
    private float offset;

    public Equation(Vector2 point1, Vector2 point2)
    {
        Vector2 distanceVector = point2 - point1;
        this.normalVector.x = -distanceVector.y;
        this.normalVector.y = distanceVector.x;
        this.offset = -1 * (normalVector.x * point1.x + normalVector.y *
            point1.y);
    }

    public bool check(int x, int y)
    {
        return (normalVector.x * x + normalVector.y * y + offset) >= 0 ?
            true : false;
    }
}
```

Listing 4.2: Code of the pineda algorithm implementation. Skipped lines are marked with "..."

```
// Implementation of the pineda algorithm
// Uses the Equation struct to calculate all pixels within the 2D triangle
    from the given 2D vectors
private Vector2Int[] Pineda(Vector2 vertex1, Vector2 vertex2, Vector2
    vertex3)
{
    List<Vector2Int> res = new List<Vector2Int>();
    // 1. Create the three equations (No Backface-Culling -> Both sides get
         rendered -> 6 equations)
    Equation eq1 = new Equation(vertex1, vertex3);
    ...
    Equation eq6 = new Equation(vertex1, vertex2);

    // 2. Get the min and max x and y values
    ...

    // Loop:
    // Check -> Add vector to array if one of the 2 equation-triples are
        correct
    for(int x = Mathf.RoundToInt(xMin); x <= Mathf.RoundToInt(xMax); x++)
    {
        for(int y = Mathf.RoundToInt(yMin); y <= Mathf.RoundToInt(yMax); y
            ++)
        {
            if ((eq1.check(x, y) && eq2.check(x, y) && eq3.check(x, y)) ||
                (eq4.check(x, y) && eq5.check(x, y) && eq6.check(x, y)))
                res.Add(new Vector2Int(x,y));
        }
    }
    // Return an array instead of a List
    return res.ToArray();
}
```

4.3.3. Animations

Implementation and design regarding the animations used for Pineda and Bresenham substages

In this subsection the design decisions and implementation details regarding the animations shown on a screen for illustrating the Pineda and Bresenham algorithms are described. The screen is located on the left side behind Kyle, as you can see in Fig. 4.6, during the Pineda and Bresenham algorithm substages. The screen is positioned such that the user can always read Kyle's text panel and also see what is happening on the animation screen on the left. The screen is used to illustrate the steps of the two algorithms to the learner and to give the learner an example on how the algorithms could work. In the illustrations some simplifications are made in order to give the user a better understanding without giving an overwhelming amount of information. This is done because in this application it is enough if the user has a rough idea on how the algorithms work, because the focus of RePiX VR is on the whole Rendering Pipeline and not only rasterization and its algorithms. You can read about the procedure of the animations in Section 4.1.5 for the Pineda algorithm and in Section 4.1.4 for the Bresenham algorithm. You can also have a quick overview over the animations on the screen by looking at Fig. 4.6, 4.7 for Pineda and by looking at Fig. 4.5 for Bresenham.
In the Bresenham animation an illustration with E, NE and d (see *c)* of Fig 4.5) was used to show to the user what the east or north-east decisions could look like in an example. The illustration for the decision variable d in this context was added to give an idea on how this decision variable works, as that the pixel is chosen in which the mapped line is nearer to d. This is not an accurate illustration since for d only a query is made whether it is positive or negative. But this was the easiest to understand illustration for d that would make sense to the user and give a rough idea what d is. The other parts of the animations are straight forward examples on how the pineda or bresenham algorithm respectively would work on an example.
On the implementation side the animation screen is just the reused fragments screen and uses the *Ra_ImageController.cs* script to display the given images of the animations on the screen. You can read more about that script in Section 4.3.2. For controlling the animations another script, called *Ra_FragmentsAnimation* is also attached to the fragments/animation screen. In this script multiple functions are defined, that create the different images you can see on the screen by putting the respective pixels in an array and scaling that image up so that the animation looks smoother. This image is then given to another input method of the Image Controller, the UpdateAnimation method which gets as parameter the image that should be shown on the screen an the color, the pixels of this image should have. With this the animations are done by calling the respective methods of the FragmentsAnimation component in the timeline one after another.

4.3.4. Other Objects

Design decisions for all other objects located in the scene

In this subsection the design decisions regarding the different objects and illustrations and their positioning inside the scene are described in detail.

Fragments illustration

Reasons for using the fragments illustration

In the main stage the first illustration the learner gets to see is the image used for explaining the fragments (see Fig. 4.4). This image is shown to the user in order to give them a rough idea, what the rasterization of a primitive could be. The image is not explained at this point of the stage because some information that can be seen on the image is not explained in detail at this part of the main stage. The image is mainly shown to give the

user something to see and visualize and is explained later in the corresponding fragments substage for which the image got created for.

Input Mapping Illustrations

Design decisions regarding the input mapping illustrations

In the input mapping substage an illustration is shown to the left which illustrates the dimensions of the input vectors and needed output vectors; an equation used for calculation is presented to the right of the user on a panel (see Fig. 4.14). The locations of these panels are chosen because they are perfectly visible from the standard position of the user. Also the panels turn with the user so that they can be read and viewed from any angle.

Figure 4.14.: Screenshot from the input mapping substage. Graphical illustration showing the dimensions of the input to the left. Kyle and his text panel in the middle. Panel with equation on the right.

Screens

Design decisions regarding the screens

The next illustration the user gets shown is the main illustration you can see in Fig. 4.2 to the left of the learner. The frustum is reused from another stage of RePiX VR to give the user the feeling of a continuous design and also make it clear to the user in the playground stage, where self constructed triangles have to be placed in order for them to be visible. The real-time 2D image and the fragments screen are placed on the two sides of the 3D scene and the frustum. This placement was chosen so that the user is able to see both screens perfectly fine when standing inside of the 3D scene which is the case in the playground substage. In an earlier implementation of the stage the two screens were placed side by side, but this had the disadvantage that one screen was seen at a different and smaller angle than the other one which made the comparison between the two images harder. Also a design decision was dropped, to let the screens turn to the user so that the user would always see the screens directly from the front. This design was dropped, because the screens would be less comparable to each other if the angle the user is looking at the screens is not the same. Since both screens show the same image from the same perspective of the camera in different versions it is very helpful for the learner to see both images in a setting where both screens are parallel to each other.

3D Scene

Design decisions regarding the 3D scene with the objects in it

The objects in the 3D scene are made up from triangles. The shapes of the objects were chosen to have a variety of different simple objects that can be made up from triangles. Three objects were chosen to have the situation were two objects overlap, but make the scene still well structured so the user can understand, which primitive is highlighted and rasterized at the moment. Also with more objects the animation would take longer. The time one object gets highlighted and rasterized during the animation is chosen to be long enough to see on all three screens what is happening and also quickly change the triangles to go trough all triangles fast enough to not lose the learners attention. The white triangles in the background were chosen to be placed in order to give the learner a visualized border of where triangles get rasterized (mainly for the building of the self constructed triangles) and also show on the screen what it looks like to rasterize the triangles covering half of the screen. To make the whole scene illustration and the rasterization even more accessible to the user, all triangles beside the background triangles are transparent. With this the user is able to see even triangles that are highlighted and would usually be covered by another triangle from the perspective of the learner. This deepens the understanding of rasterization and the detail, that every triangle gets rasterized independent from their position in the scene and independent from being covered by other primitives.

Triangle Builder

Design decisions regarding the reused triangle builder

The triangle builder is used in the playground substage. It is a reused and adjusted component and was implemented for another stage of RePiX VR. The menu for the triangle builder is placed to the right of Kyle's text (see Fig. 4.15) and follows the player, so that it is always located at this position on the screen. This position for the triangle builder was chosen because it is located behind the player if they look at the 3D scene and illustration with the fragments screen and thus the user can always use the menu wherever they like and the buttons don't interfere with other interactable parts of the scene. Also the triangle builder menu doesn't cover other objects with this placement. The tutorial videos are also part of the reused component and are placed above the menu.

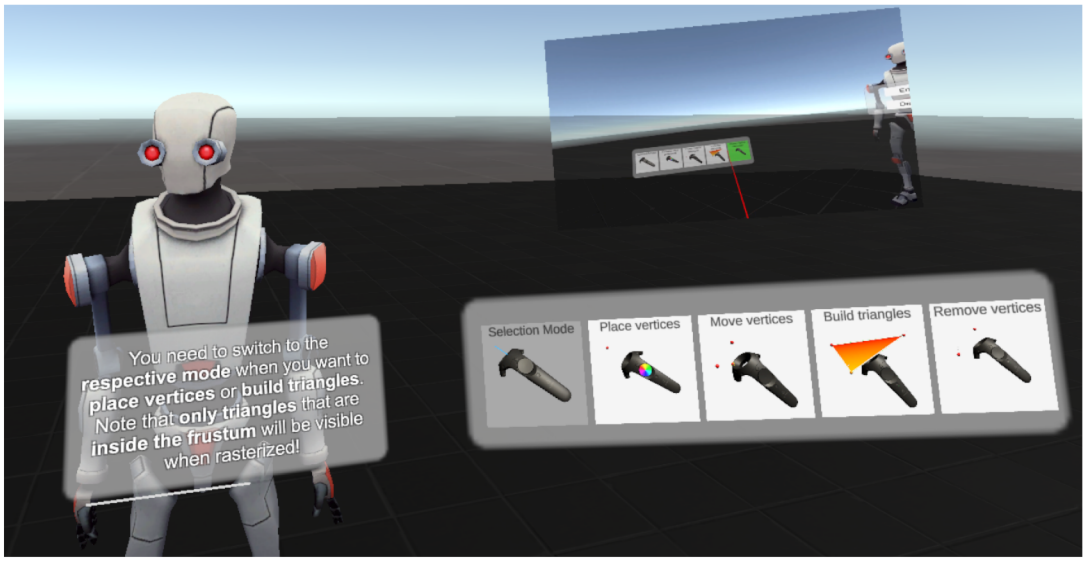

Figure 4.15.: Screenshot from the playground substage. To the left Kyle with his text panel. On the right the triangle builder menu with the buttons. Above the menu panel the tutorial videos for the triangle builder are shown.

The triangle builder is adjusted to the stage accordingly that the build triangles have a *Ra_Triangle.cs* script and also a *Pointable.cs* script attached to them and thus are able to be selected for rasterization and can be shown on the fragments screen.

4.3.5. Tracking

The use of tracking in RePiX VR

The RePiX VR project has an implemented tracker for information on what the user does and how they behave using the application. This tracker is especially useful for the *researcher* persona who can use this data to analyze the learning behaviour of the users inside of the application and with this data improve the experience the learners have in the environment. In the future is is planned to use the Learning Analytics results to give feedback and support reflection for learners and teachers. The trackable data includes all gestures the player can make, even eyetracking and the interaction possibilities with all objects that are explicitly tracked by adding a specific *Trackable.cs* script to the components. With this it can be tracked, which elements in the application the user points on, selects, uses or looks at and much more.
To support this tracking framework all relevant objects of the new rasterization stage were added to the tracker environment by adding the *Trackable.cs* script to all relevant components, such as the resolution slider, the buttons on the Q&A panel, the pointable primitives, etc. All actions that could be relevant for the *researcher* while playing the new rasterization stage can be tracked and analyzed with this to improve the rasterization stage and RePiX VR even further.

Part III

Evaluation

Chapter 5 Evaluation

This part will deal with the evaluation of the implementation and the content of the stage. At first the content of the old stage is summarized and evaluated which was previously used in RePiX VR. In the second section the new stage is evaluated by surveying multiple participants that played the new stage. The third section summarizes all possible future improvements for this stage and RePiX VR that could be derived from the evaluation in this chapter.

Structure of the evaluation chapter

5.1. Old Stage and Comparison

The old state of the rasterization stage that was updated with this thesis contained one illustration for the user and four text panels regarding rasterization for Kyle. The text panels were: "The rasterization detects which primitives can be potentially seen on screen. Those are called **fragments**.", followed by: "A fragment can be discarded or become a pixel itself.", the next panel was: "On the left image you can see a 3D scene from the top. Each objects gets projected one the screen to get potential pixels." and the last panel: "The rasterization is done e.g. with **Digital Differntial Analysis (DDA)**, **Bresenham (Midpoint)** or **Pineda** algorithm.".

Summarizes the conent of the old stage and compares it to the new stage

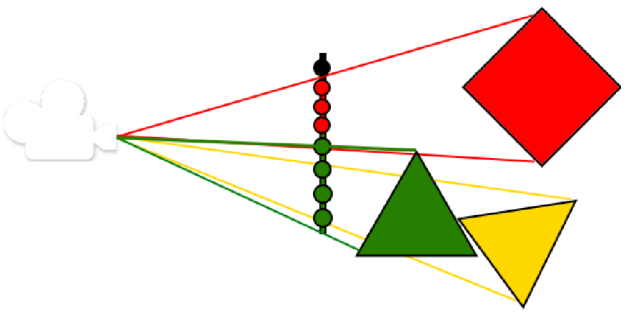

Figure 5.1.: Shows the illustration used in the old rasterization stage.

As you can see the stage had only few information for the learner and made use of only one illustration, which can be seen in Fig. 5.1.
This illustration was a model for the new illustration for the fragments as you can see in Fig. 4.4, but is not a sufficient image to illustrate the whole process of rasterization.
The stage was very short and didn't cover all information about rasterization that could have fit into the application. The level of detail was below average of the level of detail of the other stages of the application.
This could be a good reason for why the learners and testers of the old RePiX VR project had remembered only minimal about this rasterization stage from all the stages, according to the first DBR cycles [9].
In comparison to that the new stage offers multiple levels of interaction, many illustrations to make the learning material clear to the user and simple explanations of a wide

variety of topics regarding rasterization. The evaluation of the new stage can be found in the next sections.

5.2. Survey of the new Stage

The evaluation of the new stage implementation is done by seven testers who played the new stage with the also implemented introduction stage, which is described in Section 4.2. The evaluation was mainly done via Zoom with screen sharing because of the corona crises and due to limited time at the end of the work. Six out of the seven testers used the application in VR while one participant used the non-VR desktop version. One participant used their own VR setup which was a HTC Vive but with Valve Index [1] controllers. Four of the seven participants tested the application in German while the other three tested in English. After the participants had used the application they were surveyed multiple questions of a previously defined questionnaire. This questionnaire contained three parts: The first part in made of two questions about the preknowledge the participants had about Computer Graphics and the Rendering Pipeline or rasterization which is useful to classify the results of the questionnaires. The second part of the survey is the standardized SUS [4] which was translated to German for the participants that were interviewed in German for this work. Since some questions of the SUS, e.g. the first one, is not really applicable for this setting, for some questions an suitable context was given to the interviewed to make it easier for them to answer the questions. The first two parts were answered with scores from 0 to 4 where for the first part 4 means expert knowledge on the field and 0 means no knowledge of the topic at all while for the second part 0 means strongly disagree and 4 means strongly agree. The third part was answered in text fields and the questions were created to reflect the weaknesses and strengths of the new stage. All results of the seven questionnaires can be found in the Appendix B.

Describes how the evaluation was done

5.2.1. The System Usability Scale

The first thing that will be evaluated here is the standardized SUS score which can be calculated with the following formula [11]:

$$\begin{aligned} SUS_y = 2.5(20 + \sum(&SUS01, SUS03, SUS05, SUS07, SUS09) \\ &- \sum(SUS02, SUS04, SUS06, SUS08, SUS10)) \end{aligned}$$

In this formula SUSXX stands for the value from 0 to 4 that was picked in the questionnaire of the respective question XX that are enumerated from 1 to 10. Since there were more than five participants the results have a stable factor stability for each question according to [11]. This SUS score can be derived for all the participants (with ID y) and the mean of that score can be evaluated as the average SUS score of this system. Using this formula the following seven SUS scores for the seven participants can be derived:

$$SUS_1 = 90, \quad SUS_2 = 90, \quad SUS_3 = 85, \quad SUS_4 = 95, \quad SUS_5 = 87.5, \quad SUS_6 = 95, \quad SUS_7 = 97.5$$

This results in an average SUS score for this system of $SUS \approx 91.43$.

[1] Valve Index Controllers: `https://www.valvesoftware.com/de/index/controllers`

Evaluate the implementation using the SUS

Since the SUS score has no direct meaning and can mainly be useful in comparison to other systems [11] this score can not be interpreted without other systems to compare with. But a really high score like this (in consideration that the SUS score is in a range of 0 to 100) can definitely be interpreted as a good score for a system. According to [11] the research done in [16] have derived a grade system for the SUS that can be used to get an impression of the SUS score independently from other systems. In that grading system a SUS score of 91.43 would be the best grade of A+ and could be interpret as a grade of 1.0 with using the percentile range in the meaning of university grades in Germany.
This means that overall the system is very user friendly and the users are able to understand the system and controls very good, which is an important feedback for the rasterization stage but also for RePiX VR itself since the controls are part of RePiX VR and were just used inside the new stage. But also the elements inside the new stage were well interactable and the testers felt overall comfortable using the new stage.

5.2.2. Questions and Annotations

This section will deal with the answers from the participants to the questions of the third part of the questionnaires as well as with the annotations the testers made during or after the testing. At first the eight questions of the survey will be gone trough one after another:
The first question was whether there were explanations that the tester did not understand. Some testers reported that especially in the German version some sentences were to long for example in the Bresenham substage. One issue is, that the German and English texts are always displayed using the same panel, so that it is not possible to split the German explanations into two text panels while having only one in the English version. This is a problem because German sentences are longer on average than English sentences. However some explanations should be split into two text panels in the German version as well as in the English version, especially in the Bresenham substage. Furthermore one participant didn't quite understand the Pineda substage, but since the algorithm substages are considered to have very much information and the application should only give an inside to the topic of rasterization it is not needed that every learner can understand every aspect of the full application. One other participant didn't understand how exactly in the illustration in Fig. 4.4 is determined whether a pixel is used as a fragment or not when the pixel would only be partially covered by the primitive, which could be explained on one text panels in the future. Most participants didn't have explanations that they overall didn't understand which is a good result.
The second question was whether there were elements or illustrations that the tester did not understand. Every participant negated this question which means that overall all illustrations were very well understandable, at least for the self-report and the lower taxonomy categories [1].
The third question was whether there were objects that were badly placed or visible. Nearly every participant annotated that the main illustration with the 3D scene, the fragments screen and the preview screen was not directly in their field of view. Thus they didn't saw it directly, didn't know what the current explanations refer to or had the problem that during the whole application they always looked around to check whether there would be another illustration that they may have overlooked. The main illustration definitely needs to be relocated inside of the scene. One participant proposed to put it to the right instead of to the left, so that the text clearly tells the learner to turn to the right where no other illustration was before. The confusion is most likely to happen to all participants because the illustration was not in there field of view and because the previous

illustration was already to the left of Kyle's text panel and thus the term "to your left" may also refer to this illustration instead of the main illustration. Some other badly placed elements were the triangle builder menu which was sometimes not interactable because of the frustum and the 3D model of the camera with the frustum that could occult the 2D image preview.

Third part of the survey in detail

The fourth question was whether the preknowledge that was imparted by the introduction to the Rendering Pipeline was sufficient to understand all explanations of the stage. Every participant answered that the preknowledge was enough which means that it can be also concluded that in the default RePiX VR application in which all the preknowledge gets imparted by the other stages is also sufficient to understand the new rasterization stage.

The fifth questions was which substage the participants liked most and the sixth which least. Nearly all participants liked the Playground substage most, which is not surprising since it is by far the most interactable substage. But two participants answered with the Pineda and Bresenham substage which is interesting but also easy to explain because these substages had still with their animations more interesting features than the other substages (excluding the Playground), because moving and evolving animations are more interesting than standing images. Most participants had trouble picking a least favorite substage, some of them didn't pick one at the end. This means that the substages were all very interesting and well made for most of the participants. The participants that chose substages that they liked least chose the fragments substage, the introduction, the bresenham substage and the pineda substage. No substage was taken twice which means that there was no substage that was outstanding with a bad experience for the learners. Also the Pineda substage was chosen by the tester that did not get the explanations of the Pineda algorithm which makes sense. The Bresenham and Pineda substages overall have the most complex content and thus it makes sense that they are maybe not liked the most. That the introduction sequence was picked once is not really important for the whole stage, since it is not part of the rasterization stage that is in RePiX VR.

The seventh question was whether the tester still has any questions regarding rasterization that they would have liked to get answered. Only one participant had a remaining question which was already mentioned before regarding the pixel in the fragments illustration. That no other questions are open to the testers is a very good sign for the overall stage design. In the future it can be researched further whether the participants really understood all the information from the explanations and illustrations using quizzes or other tools. With this it can be further determined whether the illustrations and explanations are sufficient and detailed enough.

The last question for the participants was to grade the stage in a grade system where 1 would be the worst grade and 15 would be the best grade (which is a standard grade system in upper classes of German high school). Four of the seven participants graded the stage with 13 out of 15 points, the other three with 11, 12 and 13,5 points. Even the worst given grade is still a good grade, the average grade is 12,64 and the median is 13. Just as the SUS score these results show that the stage and explanations were overall very well received by the testers.

It can be annotated shortly that the tester that used the desktop version and the tester who used index controllers in their own VR setup had no problems using the application.

The next part will deal with further annotations made by the testers while or after using the application. Every tester was asked if they did understand why the buttons to enter the different substages were colored in green, yellow and red. Three out of the seven participants naturally assumed that the colors indicated difficulty values just as it is meant

to be. Four of the participants didn't mind the colors or thought of it as something like RGB values. Since it was planned regardless that in further implementation of RePiX VR Kyle would give a hint via audio on this it is not fatal, that most participants did not get this indication in the current implementation.
The annotations of the participants can be summarized as: That the introduction was very useful, that the main illustration is not directly visible (as already derived from the questionnaires before), that the algorithms were well explained, that the user marked triangles in the playground substage should stay marked and that they liked that in the preview image of the main illustration everything could be seen including the controllers and laser pointers of the user.

Describe all other annotations made be the testers

Further things that could be noticed while the participants tested the implementation was that every participant except one played the stages in the same order (left to right and top to bottom), that only one participant wanted to see the explanations of the main stage again and that most importantly only one participant heard the full explanations of the triangle builder before using it in the playground substage. Most participants used the triangle builder before knowing how it works, right after it gets shown. This makes sense because the participants have no indication on that the explanations of the triangle builder are not completed. But if the triangle builder menu would get shown after the explanations finish, the users wouldn't know, what the explanations refer to and thus could be confused what the explanations are about. An solution for this problem is proposed in the next section.

5.3. Future Improvements

In this section a list of possible improvements for future development are given regarding this stage and RePiX VR. At first some smaller bugs regarding the implementation of the stage are listed that could be fixed. The reason the bugs were not fixed before finishing this work are that the bugs would have been fixed after the evaluation. But if new bugs occurred while fixing the other bugs the bugs could not be reported in this work because another evaluation would not be possible in the remaining time. Thus a list of known small bugs is arguably better than fixing these and maybe producing new bugs that are not known after reading this work.

Structure of the section

Small Bugs

The bugs that are known to exist in the current implementation are the following: When the learners creates their own triangles and hover over it in order to rasterize it, the triangles are either highlighted in red or are getting invisible. Which of these occur is dependent on the order the vertices are connected to make the triangle because of *backface culling*. In the best case when the user triangles are hovered over the same color of the triangles would appear as usual, maybe a bit darker.
Another known bug is, that the material of the triangles when selected by the user is not updated permanently until the next triangle is selected. Thus the user is not able to see which triangle is highlighted at the moment inside the 3D scene or the 2D preview image. This could be fixed by setting the material of the highlighted primitive to the highlight material when selected and resetting the materials of all triangles when selecting a new one.
One more bug is that the text "Resolution" above the resolution slider of the fragments screen is displayed in English in the German version of the application. Also in the desktop-version the value of the resolution itself is not displayed on the slider that is

Mentions all small bugs currently known for the new stage

used to change the resolution. Thus the slider can be changed but the user doesn't know the exact resolution.
The last known small bug is that the frustum can block some interactions with other elements. This can result in wrong teleportations when inside the frustum or that the mode of the triangle builder can not be changed while inside th frustum in some cases.

Placement of the Main Illustration

Possible solutions regarding the placement of the main illustration

One bigger improvement that has to be made, which has been annotated by all testers, is that the main illustration with the 3D scene and the two screens has to be relocated. This is needed because the illustration appears outside of the users field of view and thus the users can not see it directly. The learners have to look around in order to see it, which also results in situations in which learners look around to look for other illustrations that could also appear outside of their field of view. One solution for this mentioned by one participant is to relocate the whole illustration to the right side of the user. With this it could be clearly stated on Kyle's text panel that there is an illustration on the right. Thus the illustration would not be mistaken for the small image that is also shown to the left of Kyle's text panel. An even simpler solution for that problem could be to just relocate the small image that is shown before to be on the right side of Kyle. Than it would be absolutely clear that on the left side a new illustration must have appeared when mentioned in Kyle's text. Another solution could be to not show the small image in this main stage at all since it is never referred to on Kyle's text panels in the main stage and is just shown to give the learner something to visualize what they read about.

Shorter Explanations

Too long explanations and possible solutions

It was reported multiple times by the participants of the evaluation, that few of the explanations, especially in the Bresenham substage had long sentences and explanations that could be hard to understand. This was mainly in the German version the case since German sentences are longer than English sentences on average. One simple solution could be to split some explanations with long texts into two text panels and to split some of the sentences into two sentences. Also when splitting the text panels the font size on some text panels could be enlarged, because they had to be reduced on some text panels so that the text fits on the panel in the German version. Overall some testers also mentioned that the font size was a little small and it was hard to read the white text on the bright panel. When they teleported nearer to the text panel they could read it easily but having a greater font size or a better background for the text could also be helpful throughout RePiX VR.

Multi-panel explanations

One issue that could be derived in the evaluation is that the timing of displaying the different illustrations that are explained on multiple text panels can be a problem. When a complex illustration needs multiple text panels to be described the timing on when to show the described illustration to the user is an important aspect for understanding. These multi-panel explanations appear often in the rasterization stage, for example in the playground substage in which the triangle builder gets explained, or in the main stage in which the 3D illustration gets explained or the image in the fragments substage. There are two options for showing these illustrations: first one that is used in the new stage

is that they get shown at the start of the explanations, which helps the learner to understand every aspect of the illustration when it gets explained but has the disadvantage that the user doesn't know when this illustration is fully explained and could try to understand an illustration that they doesn't have enough information on yet. This could lead to confusion or frustration. The second option is to show the illustration at the end of the explanation. This would have the benefit that the learner has heard every information they need in order to understand the illustration but would have a major disadvantage, that they have to keep every explained information in mind before even seeing the illustration and having nothing to visualize all these information before seeing it. This would most probably significantly impair the quality of the learning experience.

Multi-panel explanations

One solution for this problem could be to give the user hints on when the explanations of this illustration ends. This could be done for example with an arrow ("->") at the end of each text panel of such an explanation, if the following text panel is still part of the explanation. With this the user always can look at the illustration to just understand the current text panel and at the end of this multi-panel explanation the learner can try to understand the whole illustration. Another way of doing this could be with page numbers like (" 1/5 ") or something like that, which has more information but may be less aesthetic.

The main reason this feature was not implemented in the new stage after the evaluation is, that this feature could also be a great improvement for other implemented or upcoming stages of RePiX VR and should thus be a global and consistent feature throughout the whole application. This could be a major improvement for the learning experience in the RePiX VR project.

Chapter 6 Conclusion

In this work a new rasterization stage for the RePiX VR project was designed and implemented. Some aspects were added to the rasterization stage that are new in the RePiX VR project, like the Q&A-panel that is used to make the user choose a substage on a topic they are interested in. This leads to more possible interaction for the learners in the VR environment which can lead to a better learning experience. A very interactive substage was implemented in which the learner could create their own triangles and see what they look like when rasterized. The different algorithms that can be used for rasterization were explained in respective substages using 2D animations shown to the user. The interactive elements and the illustrations can enhance the quality of the learning and make the users memorize more of what they have learned.
The new stage was evaluated by seven testers using a questionnaire which included the System Usability Scale and other questions regarding the new stage. The results of the survey where overly positive and showed that the learners liked the new stage and understood all of the illustrations and most of the explanations that were given. But also some aspects could be derived which have to be improved in the future.

Summary of the work

The main aspects that need to be improved are the placement of the main illustration, that could not be instantly seen by the learners, because it is outside of their field of view. Also some explanations should be revised that are too long and should be split up on multiple text panels to not confuse the learners by reading long and hard to understand sentences. A last major improvement that can be made to this stage as well as to the whole RePiX VR application are indicators on multi-panel explanations. Which means that when multiple text panels are needed to explain an illustration it should be indicated to the user, that the current explanation is not over yet. Also this should indicate when they should try to understand the illustration they get shown and when they do not need to understand it yet.
For all these problems possible solutions were given in the respective section of the evaluation.
In summary one can say that the new stage is way more interactive and fun to use for the learners in RePiX VR than the old one. Thus this work is a great improvement for the RePiX VR project. Yet there are some improvements that can be made to enhance the learning experience in RePiX VR and explicitly this rasterization stage even more.

Appendix A Bibliography

References

[1] Lorin W Anderson and David R Krathwohl. *A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives*. Longman, 2001.

[2] Terry Anderson and Julie Shattuck. "Design-based research: A decade of progress in education research?" In: *Educational researcher* 41.1 (2012), pp. 16–25.

[3] Jack E Bresenham. "Algorithm for computer control of a digital plotter". In: *IBM Systems journal* 4.1 (1965), pp. 25–30.

[4] John Brooke. "SUS - A quick and dirty usability scale". In: *Usability evaluation in industry* 189.3 (1996).

[5] Julian Busse et al. "How to Design Learning Applications that Support Learners in their Moment of Need–Didactic Requirements of Micro Learning". In: *AMCIS 2020 Proceedings*. 15 (2020).

[6] Ian Dunn and Zoë J. Wood. *Graphics Programming Compendium*. `https://graphicscompendium.com/index.html`. Accessed: 2021-12-13.

[7] Sergej Görzen. "Technology-Enhanced Learning of Computer Graphics Rendering Pipeline in Virtual Reality". Master thesis. Aachen: RWTH Aachen University, 2020, 117 S. URL: `https://publications.rwth-aachen.de/record/816378`.

[8] Hans W Gschwind. "Digital differential analyzers". In: *Electronic computers*. Springer, 1962, pp. 139–209.

[9] Birte Heinemann, Sergej Görzen, and Ulrik Schroeder. "RePiX VR - Learning environment for the Rendering Pipeline in Virtual Reality". In: *Eurographics 2022 - Education Papers*. Ed. by Jean-Jacques Bourdin and Eric Paquette. The Eurographics Association, 2022. ISBN: 978-3-03868-170-0. DOI: `10.2312/eged.20221040`.

[10] Jason Jerald. *The VR Book: Human-Centered Design for Virtual Reality*. ACM Books 8. Association for computing machinery Morgan & Claypool publishers, 2016. ISBN: 978-1-97000-113-6.

[11] James R Lewis. "The system usability scale: past, present, and future". In: *International Journal of Human–Computer Interaction* 34.7 (2018), pp. 577–590.

[12] Tomas Möller. *Real-Time Rendering*. Fourth edition. Boca Raton: Taylor & Francis, CRC Press, 2018. ISBN: 978-1-138-62700-0.

[13] Geraldine O'Neill and Tim McMahon. "Student-centred learning: What does it mean for students and lecturers". In: *Emerging issues in the practice of university learning and teaching* (2005).

[14] Juan Pineda. “A parallel algorithm for polygon rasterization”. In: *Proceedings of the 15th annual conference on Computer graphics and interactive techniques*. 1988, pp. 17–20.

[15] *Rasterization: a Practical Implementation*. `https://www.scratchapixel.com/lessons/3d-basic-rendering/rasterization-practical-implementation`. Accessed: 2021-12-13.

[16] Jeff Sauro and James R Lewis. *Quantifying the user experience: Practical statistics for user research*. Morgan Kaufmann, 2016.

Appendix B Evaluation Questionnaires

0.0	ID	01
0.1	Wie viel weißt du über Computer Grafik?	1 2 3 4 5 — Nie gehört … Experten wissen
0.2	Wie viel weißt du über die Rendering Pipeline / Rasterization?	
1.1	Ich denke ich würde das System regelmäßig benutzen	Lehne stark ab … Stimme stark zu
1.2	Ich fand das System unnötig komplex	
1.3	Ich fand das System leicht zu benutzen	
1.4	Ich denke ich bräuchte Unterstützung von einer technischen Person, um das System Nutzen zu können	
1.5	Ich fand die verschiedenen Funktionen im System waren gut integriert	
1.6	Ich denke es gab zu viel Inkonsistenz im System	
1.7	Ich denke die meisten Personen könnten schnell lernen, das System zu benutzen	
1.8	Ich fand das System war sehr merkwürdig zu benutzen	
1.9	Ich war sehr selbstsicher beim Benutzen des Systems	
1.10	Ich musste sehr viele Sachen lernen, bevor ich mit dem System umgehen konnte	
2.1	Gab es Erklärungen, die du nicht verstanden hast?	Manche Sätze sind zu lang / unverständlich
2.2	Gab es Elemente/Illustrationen, die du nicht verstanden hast?	Nein
2.3	Gab es Objekte, die deiner Meinung nach schlecht platziert/einsehbar waren?	Kamera und Frustum haben 2D Preview verdeckt
2.4	Gab es Erklärungen für die das Vorwissen (Einführung) nicht gereicht hat?	Nein
2.5	Welche Substage hat dir am besten Gefallen?	Pineda
2.6	Welche Substage hat dir am wenigsten Gefallen?	Keine
2.7	Hast du noch Fragen zu Rasterization, die du gerne beantwortet gesehen hättest?	Keine
2.8	Von 1-15, welche Note würdest du der Stage geben?	13

Figure B.1.: Questionnaire 01: Played in VR - German version - German interview.

0.0	ID	02
0.1	Wie viel weißt du über Computer Grafik?	1 2 3 4 5 Nie gehört — Experten wissen
0.2	Wie viel weißt du über die Rendering Pipeline / Rasterization?	
1.1	Ich denke ich würde das System regelmäßig benutzen	Lehne stark ab — Stimme stark zu
1.2	Ich fand das System unnötig komplex	
1.3	Ich fand das System leicht zu benutzen	
1.4	Ich denke ich bräuchte Unterstützung von einer technischen Person, um das System Nutzen zu können	
1.5	Ich fand die verschiedenen Funktionen im System waren gut integriert	
1.6	Ich denke es gab zu viel Inkonsistenz im System	
1.7	Ich denke die meisten Personen könnten schnell lernen, das System zu benutzen	
1.8	Ich fand das System war sehr merkwürdig zu benutzen	
1.9	Ich war sehr selbstsicher beim Benutzen des Systems	
1.10	Ich musste sehr viele Sachen lernen, bevor ich mit dem System umgehen konnte	
2.1	Gab es Erklärungen, die du nicht verstanden hast?	Nein
2.2	Gab es Elemente/Illustrationen, die du nicht verstanden hast?	Nein
2.3	Gab es Objekte, die deiner Meinung nach schlecht platziert/einsehbar waren?	Main Illustration nicht direkt gesehen
2.4	Gab es Erklärungen für die das Vorwissen (Einführung) nicht gereicht hat?	Nein
2.5	Welche Substage hat dir am besten Gefallen?	Playground
2.6	Welche Substage hat dir am wenigsten Gefallen?	Fragments
2.7	Hast du noch Fragen zu Rasterization, die du gerne beantwortet gesehen hättest?	Nein
2.8	Von 1-15, welche Note würdest du der Stage geben?	13

Figure B.2.: Questionnaire 02: Played in VR - German version - German interview.

0.0	ID	03
0.1	How much do you know about Computer Graphics?	1 2 3 4 5 (Never heard … Expert)
0.2	How much du you know about the Rendering Pipeline / Rasterization?	
1.1	I think that I would use this system frequently.	(Strongly disagree … Strongly agree)
1.2	I found the system unnecessarily complex.	
1.3	I thought the system was easy to use.	
1.4	I think that I would need the support of a technical person to be able to use this system.	
1.5	I found the various functions in this system were well integrated.	
1.6	I thought there was too much inconsistency in this system.	
1.7	I would imagine that most people would learn to use this system very quickly.	
1.8	I found the system very awkward to use.	
1.9	I felt very confident using the system.	
1.10	I needed to learn a lot of things before I could get going with this system.	
2.1	Were there explanations that you didn't understand?	Fragments - Full pixel or not
2.2	Were there elements/illustrations that you didn't understand?	No
2.3	Were there objects, that in your opinion were badly placed/visible?	Main illustration - not visible
2.4	Were there explanations for which the pre-knowledge (introduction) was not sufficient?	No
2.5	Which substage did you like most?	Playground
2.6	Which substage did you like least?	Introduction
2.7	Do you still have any questions regarding rasterization that you would have liked to get answered?	Fragments - Full pixel or not
2.8	From 1-15, which grade would you give the stage?	11

Figure B.3.: Questionnaire 03: Played in non-VR (desktop) - English version - English interview.

0.0	ID	04
0.1	Wie viel weißt du über Computer Grafik?	1 2 3 4 5 — Nie gehört … Experten wissen
0.2	Wie viel weißt du über die Rendering Pipeline / Rasterization?	
1.1	Ich denke ich würde das System regelmäßig benutzen	Lehne stark ab … Stimme stark zu
1.2	Ich fand das System unnötig komplex	
1.3	Ich fand das System leicht zu benutzen	
1.4	Ich denke ich bräuchte Unterstützung von einer technischen Person, um das System Nutzen zu können	
1.5	Ich fand die verschiedenen Funktionen im System waren gut integriert	
1.6	Ich denke es gab zu viel Inkonsistenz im System	
1.7	Ich denke die meisten Personen könnten schnell lernen, das System zu benutzen	
1.8	Ich fand das System war sehr merkwürdig zu benutzen	
1.9	Ich war sehr selbstsicher beim Benutzen des Systems	
1.10	Ich musste sehr viele Sachen lernen, bevor ich mit dem System umgehen konnte	
2.1	Gab es Erklärungen, die du nicht verstanden hast?	Ja, ein Text Panel zu lang bei Bresenham
2.2	Gab es Elemente/Illustrationen, die du nicht verstanden hast?	Nein
2.3	Gab es Objekte, die deiner Meinung nach schlecht platziert/einsehbar waren?	Main Illustration besser rechts als links
2.4	Gab es Erklärungen für die das Vorwissen (Einführung) nicht gereicht hat?	Nein
2.5	Welche Substage hat dir am besten Gefallen?	Playground
2.6	Welche Substage hat dir am wenigsten Gefallen?	Nein
2.7	Hast du noch Fragen zu Rasterization, die du gerne beantwortet gesehen hättest?	Nein
2.8	Von 1-15, welche Note würdest du der Stage geben?	13,5

Figure B.4.: Questionnaire 04: Played in VR - German version - German interview.

0.0	ID	05
0.1	Wie viel weißt du über Computer Grafik?	1 2 3 4 5 Nie gehört — Experten wissen
0.2	Wie viel weißt du über die Rendering Pipeline / Rasterization?	
1.1	Ich denke ich würde das System regelmäßig benutzen	Lehne stark ab — Stimme stark zu
1.2	Ich fand das System unnötig komplex	
1.3	Ich fand das System leicht zu benutzen	
1.4	Ich denke ich bräuchte Unterstützung von einer technischen Person, um das System Nutzen zu können	
1.5	Ich fand die verschiedenen Funktionen im System waren gut integriert	
1.6	Ich denke es gab zu viel Inkonsistenz im System	
1.7	Ich denke die meisten Personen könnten schnell lernen, das System zu benutzen	
1.8	Ich fand das System war sehr merkwürdig zu benutzen	
1.9	Ich war sehr selbstsicher beim Benutzen des Systems	
1.10	Ich musste sehr viele Sachen lernen, bevor ich mit dem System umgehen konnte	
2.1	Gab es Erklärungen, die du nicht verstanden hast?	Ja, Pineda
2.2	Gab es Elemente/Illustrationen, die du nicht verstanden hast?	Nein
2.3	Gab es Objekte, die deiner Meinung nach schlecht platziert/einsehbar waren?	Triangle Builder Menu
2.4	Gab es Erklärungen für die das Vorwissen (Einführung) nicht gereicht hat?	Nein
2.5	Welche Substage hat dir am besten Gefallen?	Bresenham
2.6	Welche Substage hat dir am wenigsten Gefallen?	Pineda
2.7	Hast du noch Fragen zu Rasterization, die du gerne beantwortet gesehen hättest?	Nein
2.8	Von 1-15, welche Note würdest du der Stage geben?	12

Figure B.5.: Questionnaire 05: Played in VR - German version - German interview.

0.0	ID	06
0.1	How much do you know about Computer Graphics?	Never heard — 1 2 3 4 5 — Expert
0.2	How much du you know about the Rendering Pipeline / Rasterization?	Never heard — 1 2 3 4 5 — Expert
1.1	I think that I would use this system frequently.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.2	I found the system unnecessarily complex.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.3	I thought the system was easy to use.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.4	I think that I would need the support of a technical person to be able to use this system.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.5	I found the various functions in this system were well integrated.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.6	I thought there was too much inconsistency in this system.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.7	I would imagine that most people would learn to use this system very quickly.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.8	I found the system very awkward to use.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.9	I felt very confident using the system.	Strongly disagree — 1 2 3 4 5 — Strongly agree
1.10	I needed to learn a lot of things before I could get going with this system.	Strongly disagree — 1 2 3 4 5 — Strongly agree
2.1	Were there explanations that you didn't understand?	No
2.2	Were there elements/illustrations that you didn't understand?	No
2.3	Were there objects, that in your opinion were badly placed/visible?	Main Illustration badly placed
2.4	Were there explanations for which the pre-knowledge (introduction) was not sufficient?	No
2.5	Which substage did you like most?	Playground
2.6	Which substage did you like least?	Bresenham
2.7	Do you still have any questions regarding rasterization that you would have liked to get answered?	No
2.8	From 1-15, which grade would you give the stage?	13

Figure B.6.: Questionnaire 06: Played in VR - English version - English interview.

0.0	ID	07
0.1	Wie viel weißt du über Computer Grafik?	1 2 3 4 5 — Nie gehört … Experten wissen
0.2	Wie viel weißt du über die Rendering Pipeline / Rasterization?	
1.1	Ich denke ich würde das System regelmäßig benutzen	Lehne stark ab … Stimme stark zu
1.2	Ich fand das System unnötig komplex	
1.3	Ich fand das System leicht zu benutzen	
1.4	Ich denke ich bräuchte Unterstützung von einer technischen Person, um das System Nutzen zu können	
1.5	Ich fand die verschiedenen Funktionen im System waren gut integriert	
1.6	Ich denke es gab zu viel Inkonsistenz im System	
1.7	Ich denke die meisten Personen könnten schnell lernen, das System zu benutzen	
1.8	Ich fand das System war sehr merkwürdig zu benutzen	
1.9	Ich war sehr selbstsicher beim Benutzen des Systems	
1.10	Ich musste sehr viele Sachen lernen, bevor ich mit dem System umgehen konnte	
2.1	Gab es Erklärungen, die du nicht verstanden hast?	Nein
2.2	Gab es Elemente/Illustrationen, die du nicht verstanden hast?	Nein
2.3	Gab es Objekte, die deiner Meinung nach schlecht platziert/einsehbar waren?	Frustum blockiert interaktionen und teleport
2.4	Gab es Erklärungen für die das Vorwissen (Einführung) nicht gereicht hat?	Nein
2.5	Welche Substage hat dir am besten Gefallen?	Playground
2.6	Welche Substage hat dir am wenigsten Gefallen?	Keine
2.7	Hast du noch Fragen zu Rasterization, die du gerne beantwortet gesehen hättest?	Wozu könnte man es noch benutzen?
2.8	Von 1-15, welche Note würdest du der Stage geben?	13

Figure B.7.: Questionnaire 07: Played in VR - English version - German interview.

Eidesstattliche Versicherung

Koch, Julian 395322

Name, Vorname Matrikelnummer (freiwillige Angabe)

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

Visualization of Rasterization in Virtual Reality

selbstständig und ohne unzulässige Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Aachen , May 2, 2022

Ort, Datum Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt
Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt
(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.
(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

Aachen , May 2, 2022

Acknowledgement

Hiermit würde ich gerne meinen Betreuern sowohl für ihre gute Zusammenarbeit und Hilfe, als auch für das nette Feedback danken, was mir sehr bei dieser Arbeit geholfen hat. Außerdem danke ich meinen Prüfern und dem Lehrstuhl für Lerntechnologien für die Möglichkeit, hier meine Bachelorarbeit schreiben zu dürfen!
Des Weiteren möchte ich allen Testern danken, die so freundlich waren, mir bei der Evaluation meiner Arbeit zu helfen!
Insbesondere danke ich Tobias für das ausführliche Testen und Korrekturlesen dieser Arbeit!