Gregor Liebisch, BSc

Level of Detail Selection for Foveated Rendering in Virtual Reality

Master's Thesis

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Software Development and Business Management

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter Schmalstieg

Institute of Computer Graphics and Vision

Graz, January 2021

Affidavit

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

Date	Signature

Abstract

In computer graphics one approach to save performance is the use of Levels of Detail. The different levels represent the same object with a simplified version of the mesh. The higher the level, the higher the simplification. Those levels are then switched during run-time by certain criteria like the size of the object on the screen. It is quite obvious that smaller objects need less detail.

Every VR Headset consists of a display and a lens through which the renderings on the display can be seen. The lens is used to increase the field of view, which is a very important aspect in Virtual Reality. The lens itself introduces distortions, which are compensated in software by applying a barrel distortion to the rendered output. This kind of distortion squeezes objects at the edge of the screen, which makes them appear smaller. As smaller objects need less detail, the eccentricity, which is the distance to the center of the screen, is another criterion for the selection of the Level of Detail.

The Level of Detail Framework is based on a metric commonly used in visual applications, called Structural Similarity Score. The score of each object gets better with increasing distance to the camera. It was observed that there is a linear relation between the threshold of the accepted SSIM score and the current distance of a certain object. The linear relation was evaluated by conducting a user study. The final step was to measure the performance savings with respect to number of triangles as well as rendering time for the different criteria of the LOD selection.

In conclusion, the SSIM based Level of Detail Framework works well and delivers good results. The performance savings in terms of triangles and render time are a bit less than expected. The foveation function for the properties of a human eye shows more potential for performance savings and is therefore probably the future way to go.

Kurzfassung

Levels of Detail (LOD) ist eine bekannte Methode um in der Computer Grafik Leistung zu sparen. Ein Objekt besteht aus mehreren Versionen die sich im Detailgrad unterscheiden, wobei dieser durch die Anzahl der Polygone des Objektes bestimmt wird. Generell wird angenommen, dass kleinere Objekte weniger Detail benötigen.

Ein Virtual Reality Headset besteht aus einem Bildschirm und einer Linse. Die Ausgabe wird vom Benutzer durch die Linse betrachtet. Diese wird benützt, um das Sichtfeld zu erweitern und jenem des menschlichen Auges näher zu kommen. Das Bild wird durch die Linse verzerrt und per Software wieder entzerrt. Der Benutzer erhält dadurch ein unverzerrtes Bild. Durch die Entzerrung erscheinen Objekte am Rand des Bildschirms kleiner als in der Mitte. Dieser Effekt nimmt mit der Entfernung zur Bildschirmmitte zu. Die Eccentricity, die Distanz vom Zentrum des Bildschirms, wird daher als weiteres Kriterium für die Auswahl des korrekten LODs in Betracht gezogen.

Das LOD Framework basiert auf einer in der Bildverarbeitung häufig verwendeten Kennzahl, der strukturellen Ähnlichkeit. Die Ähnlichkeit zwischen dem Original eines Objektes und einer Version mit weniger Detail, wird gemessen und steigt mit zunehmender Entfernung zur Kamera. Die Auswahl des korrekten LODs erfolgt über einen Schwellwert der Messwerte, wobei ein linearer Zusammenhang zwischen diesem und der Distanz angenommen wird. Diese Annahme wird durch eine Benutzerstudie bestätigt. Als letzten Schritt werden die Einsparungen bezüglich der Leistung durch die Verwendung der LODs ermittelt.

Generell funktioniert die auf dem strukturellen Ähnlichkeit basierenden LOD Framework sehr gut. Die gemessenen Einsparungen liegen etwas unter den Erwartungen. Zieht man statt den Eigenschaften der Linse die des menschlichen Auges für die Auswahl der LODs heran, erhält man mehr Potenzial um Leistung zu sparen.

Preface

This thesis is the final work of my Master Studies at the Graz University of Technology. The researches done in order to compose this thesis were started in 2019.
I would like to use this opportunity to thank my supervisor for his excellent guidance and help during this process. My parents also deserve a particular note of thanks without whom this thesis and my whole studies would have not been possible. You've always kept me motivated.

Gregor Liebisch
Graz, January 11, 2021

Contents

List of Figures

List of Tables

List of Equations

Acronyms

AR Augmented Reality

VR Virtual Reality

MR Mixed Reality

API Application Programming Interface

GLSL OpenGL Shading Language

SSIM Structural Similarity Index

LOD Level of Detail

HMD Head Mounted Display

GLM OpenGL Mathematics

GLFW Graphics Library Framework

SPSS Statistical Package for the Social Sciences

ANOVA Analysis of Variances

FOV Field of View

IPD Interpupillary Distance

GPU Graphics Processing Unit

Chapter 1

Introduction

Virtual Reality (VR) has gained more and more popularity throughout the last couple of years while still being a quite young and new technology and therefore pushed the development of new hardware in terms of Head Mounted Displays (HMDs) as well as the general exploration of this topic quite a lot. The most important thing in VR is a perfect immersion into the virtual world which comes with a lot of requirements. Replaying the virtual world in real time requiring around sixty frames per second is one key factor to meet in order to be able to get a good virtual experience. To reach this goal good performance is the key.

Geometric Level of Detail (LOD) is a very good way to boost the performance for renderings in general. It takes advantage of using simpler models for objects meeting certain criteria. An example for one of those criteria is the size of the object. This results in the assumption that a smaller object needs less detail. For this work the comparative metric is not the size of the object but a certain score which is normally used to estimate the similarity of two images.

After getting to know the VR hardware a bit better an additional LOD criterion can be found. Lenses, which are part of every HMD, introduce distortions which have to be compensated in software. There is one aspect in the compensation process which can contribute to save performance.

1.1 Motivation

Finding a new criterion for Levels of Detail especially for VR environments by taking lens distortions into account seems to be a good way to further improve performance savings. The evaluation of considering this additional criterion for LOD selection will reveal the importance of it. Furthermore creating a LOD framework based on scores achieved by a metric used to compare two images is a new approach and it will be interesting to see how this implementation performs in the end.

1.2 Structure of the Thesis

The first section deals with related work of all the key aspects required to understand the later presented method. First a brief introduction into VR in general is given followed by some important aspects about the used rendering Application Programming Interface (API). Afterwards renderings, specifically for VR applications, are explained in more detail. Next up is another key component of this work which is Level of Detail. This work uses a specific score reached by each Level of Detail which is part of the next section. Lastly further information about the framework which was used for implementation and evaluation is given.

After getting to know all the background information, the model of the LOD framework is presented and more information on how to acquire the LODs as well as how to rate and choose them is given.

The next chapter deals with the implementation and goes into more detail how to calculate the important parameters to finally select the level of detail sufficient for current settings.

Another really important part is the evaluation of the presented ideas. This chapter is divided into two parts. At first a user study was performed and the results of the study are discussed. Followed by the evaluation of the performance savings based on the presented framework.

In the end a conclusion is made about the components discussed before and how well they performed. Furthermore future improvements and extensions are listed.

Chapter 2

Related work

This chapter is used to introduce the reader to a number of important aspects in order to be able to understand all the parts mentioned in later chapters. The chapter contains information about Virtual Reality in general, discussing basic concepts as well as hardware. Afterwards some steps how to setup renderings for VR are explained. Another part is about Level of Detail and different approaches how to implement LOD systems. The last section gives information about the Structural Similarity score which is later used to run the LOD framework.

2.1 Virtual Reality

Virtual reality is a field of computer graphics which gained significant importance recently. There is a wide field of applications ranging from environment simulation and training, such as surgical training or virtual flight and driving training or even psychotherapies in order to overcome certain phobias, to more entertaining applications like in education, arts or computer games. Because of the increasing interest in VR a lot of different hardware and software has been developed in the past decade and the technology continues to evolve rapidly. Companies compete to create virtual reality devices with the best feeling of immersion which means diving into the virtual environment. There are a lot of different aspects which have to be considered in order to reach this goal. Those requirements are further discussed later in the document.

2.1.1 VR vs. AR vs. MR

Although in general this topic gained a lot of popularity there is a high chance that some terms are mixed or the difference between them is not clear. This is also the case for terms categorizing the field of virtual environments. Milgram et al. [1] have already tried to explain this in 1995 which is what can be seen in figure 2.1.

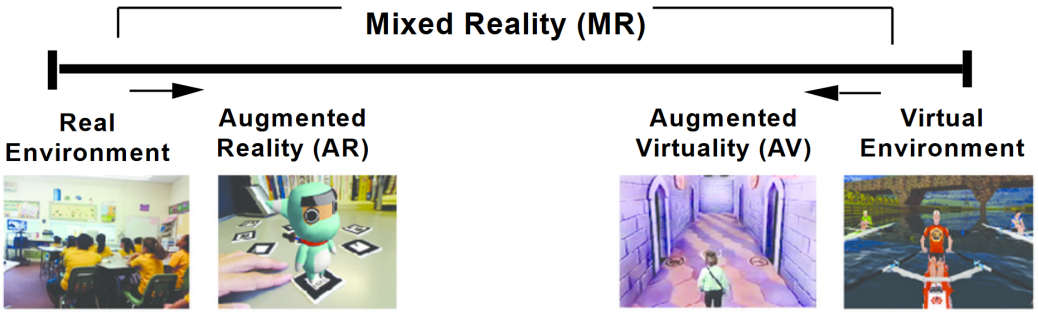


Figure 2.1: Milgram et al explanation - Adapted from Milgram, Takemura, Utsumi, Kishino. Augmented reality: A class of displays on the reality-virtuality continuum [1]

Augmented Reality (AR) focuses on bringing virtual objects into the real world. AR applications use optical or video see through displays. While optical see through displays like special glasses require additional hardware AR applications can be run on common devices like mobile phones. The real world is captured with the back camera, virtual objects are added to this image and the final rendering is displayed on the mobile phone's screen.

Virtual Reality (VR) on the other hand does not include any real world information. Everything is virtual. In order to perfectly immerse into the real world VR has some very important requirements which have to be met. In recent times we got closer to perfectly meet those requirements and therefore technology got better in fooling the human brain to believe the virtual environment is the real one.

Mixed Reality (MR) is just the hypernym for both fields explained above.

In the next chapters some of the components and their specifications are explained as well as information about the human eye and how to render images which are aligned to what the human eye sees is given.

2.1.2 Visual Perception - Human Eye

In order to get a better understanding of how VR and the corresponding renderings work one has to understand how the human eye works. Probably everyone is aware of the fact that the human eye has a great resolution which enables us to see the real world very well but only a few know that this large resolution in only available in the very center of our eye. This part of the human eye is called fovea centralis. This means the resolution decreases with increasing distance to the center. It's not only the resolution which decreases but also the ability of color vision.

No one is able to perceive colors outside of the central part of our eye. Luckily the human brain does a perfect job and once the color of an object is seen, it is remembered and therefore it appears as if color vision is always available.

The whole visual field is called the retina and we have two types of photo receptors namely rods and cones. As seen in figure 2.2 cones are concentrated in the foval area while rods are absent there but dense everywhere else. Although the overall number of rods we have is way larger than the number of cones the latter ones are responsible for color vision. The density of the cones in the fovea is also larger resulting in a better resolution.

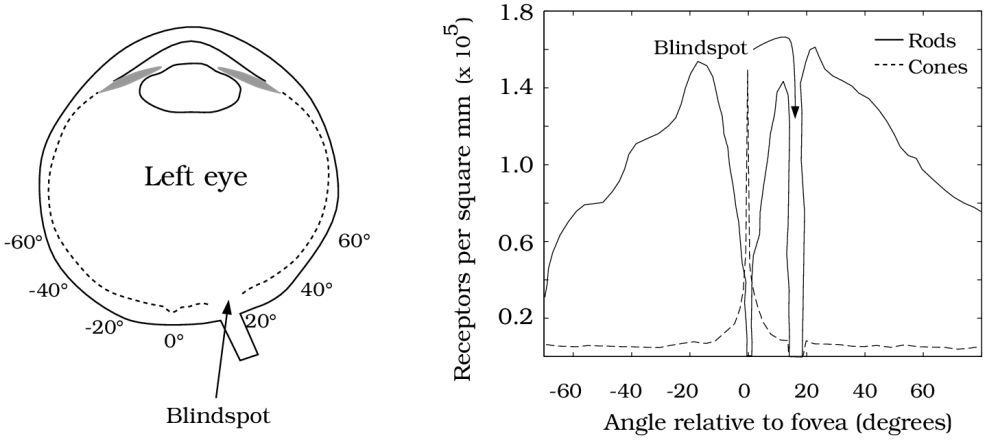


Figure 2.2: Distribution of rods and cones in the human eye - Reprinted from Brian A. Wandell. Foundations of Vision: The Photoreceptor Mosaic [2]

When looking at the distribution of rods and cones there is a certain area without any photo receptors. This are is called the blind spot. That's

where the information of all the rods and cones is bundled and leaves the eye. It could be described as the cable which is needed to transfer visual information. As there are no photo receptors in this small portion of the retina we are completely blind there.

Until now only the spatial resolution was described while the temporal resolution also plays an important role regarding VR renderings. The temporal resolution is around 25-60 Hz or even higher. This depends on aspects like the perceived brightness and frequency modulation as well as the position on the retina. That's also why one of the requirements for VR applications is that everything is rendered with at least sixty frames per second.

Another very important aspect in VR is the field of view of the head mounted displays. Let's first look at the field of view of the human eye. One eye has the following specifications:

- 60° towards the nose
- 90° towards the side
- 50° up
- 70° down

When counting right and left eye together, everyone is able to perceive information for range of around 190° horizontally with about 120° overlapping.

2.1.3 Head Mounted Displays

A head mounted display is the hardware enabling us to dive into the virtual environment. With new and better technology and the development in this sector the displays got better and therefore the quality of the virtual immersion improved a lot recently.

In later chapters where the implemented model was evaluated with the help of a user study different head mounted displays were used and further descriptions of those can be found there.

A virtual reality headset mainly consists of two parts: firstly the screen itself which shows the virtual environment and secondly lenses which are placed directly in front of this screen. Those two main components are obviously held together by a frame which is then mounted directly onto the head via elastic straps.

2.1.3.1 Screen

As for every display one obvious specification is the resolution which states how many pixels there are to show image data. The resolution is getting larger and larger which obviously improves the quality of the final images but also increases the load to render them. For the playback of videos it does not really matter how long it takes to render the output but for VR applications it's important that the rendering runs in real time because the output depends on real time measurements like the head pose and therefore it cannot be rendered in advance.

Two temporal characteristics are important for the display: the refresh rate which is the rate of refresh from memory and the frame rate which is the rate of new image generation. The limiting factor is the refresh rate but if the frame rate is higher the quality of the image for each refresh from memory can be improved, for example with anti aliasing. As previously mentioned the final output should run at approximately sixty frames per second therefore each display should be able achieve this required refresh rate.

Another important aspect in VR is the field of view. The human eye has a pretty large field of view and to be able to fully immerse into the virtual environment the same field of view should be covered. The larger the screen the more of the human view can be covered. The screen size cannot be arbitrarily large because the weight of the screen is quite important as the headset is worn. Obviously moving the screen closer to the eyes covers more of the field of view but unfortunately the human eye is not able to comfortably focus on something very close. This is where the second important part of a HMD kicks in, the lenses.

2.1.3.2 Lenses

The lens is put between the user's eye and the screen and therefore the screen is seen through the lens which has the desired positive effect that the eye is now able to comfortably look at the screen although the distance between them is very close. While lenses are a very good solution to increase the field of view and therefore also increase the feeling of immersion they also come with two downsides:

- Chromatic Aberration
  Lenses are bending the rays that come through but not all wavelengths are bend in the same way. Different colors have different wavelengths

and therefore their ways through the lens differ. This effect is called chromatic aberration. One solution to overcome this is to introduce scaling factors for each color channel to compensate the differences. For this work the effect of chromatic aberration was not taken into account as it does not really influence the final result in this case.

- Distortion
  Another effect which is probably more widely known is that lenses introduce distortions. This means that the image does not look natural anymore but is stretched towards the edge. The distortion, the corresponding compensation, and their effects are the main part of this thesis and discussed in an own section later on.

Normal optical lenses which are the typical lenses for magnifying glasses were also used for virtual reality HMDs but are now often replaced by an alternative, the Fresnel lens. The refractive power of any lens comes from the curvature of it's surface. Compared to optical lenses Fresnel lenses try to reduce as much material of the lens itself while maintaining the desired curvature. Figure 2.3 shows how this type of lens can be constructed.

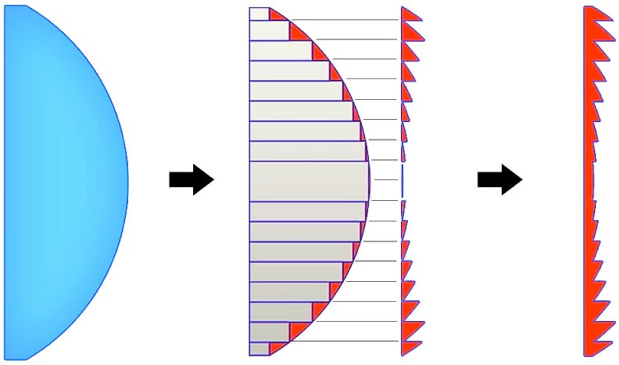


Figure 2.3: (Left) Optical Lens (Middle) Construction of Fresnel Lens by removing Material and aligning remaining segments (Right) Fresnel Lens - Adapted from Parabolix Lightning LLC. Fresnel Lens [3]

The advantages of Fresnel lenses are that by removing a lot of material they get lighter and are thinner than optical lenses which is just perfect for virtual reality applications as they contribute to reduce the weight of the HMD. Another advantage is that they are cheaper than the original ones.

2.1.3.3 Metrics within the HMD

The two above presented main components of an HMD result in the following metrics for a VR headset.

- Lens Separation
  This is the distance between the center of the lenses. It should be the exact same as the distance between the eyes of the user. Some headsets have the possibility to modify this metric.

- Screen to Lens Distance
  This is the distance between the screen and the lens and it is fixed for each HMD. As the HMD is configured to work perfectly at this distance the setting is not configurable.

- Eye to Lens Distance
  Describes the distance from the eye of the user to the lens. It is commonly stated as Interpupillary Distance (IPD). As the eye should be places as closely as possible to the lens this metric contributes to the quality of the VR experience.

2.1.3.4 Tracking

Another extremely important aspect in VR is the tracking of the device. As the head moves around the user expects that those movements are perfectly mirrored in the virtual environment. Therefore tracking of the headset is needed. In general there are two different types of tracking. Inside-out and outside-in tracking. Both approaches are used for VR HMDs.

Inside-out tracking means that cameras are placed on the HMD itself which then track the environment. Outside-in tracking needs to have cameras placed somewhere in the environment which then track the headset. For the user study the Oculus Rift S, which is using Inside-out tracking, as well as the HTC Vive, using the other approach, were used. In figure 2.4 the two techniques are illustrated.

2.1.4 Problems with VR

Now information about the human eye and the key components and aspects of a head mounted display were given but there are still some things to consider

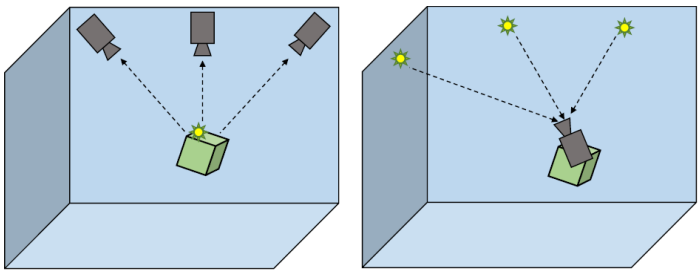


Figure 2.4: (Left) Outside-in Tracking (Right) Inside-out Tracking

to avoid motion sickness as well as vergence accommodation mismatch which both destroy the VR experience.

2.1.4.1 Motion Sickness

Dizziness, feeling of sickness or simply feeling uncomfortable are quite often associated with virtual reality, that's because of the so called motion sickness [4]. In general this feeling can occur if different sensory inputs of the human body do not match each other. In the case of VR this can happen if for example the eye observes motion while looking through the HMD but this motion is not confirmed by the sense of balance of the body. Because of those mismatches the human body raises an alarm. The brain thinks the human is hallucinating and therefore concludes the body is poisoned. The recover from this nausea is activated and that's the reason for feeling of sickness for bad immersions into the virtual environment.

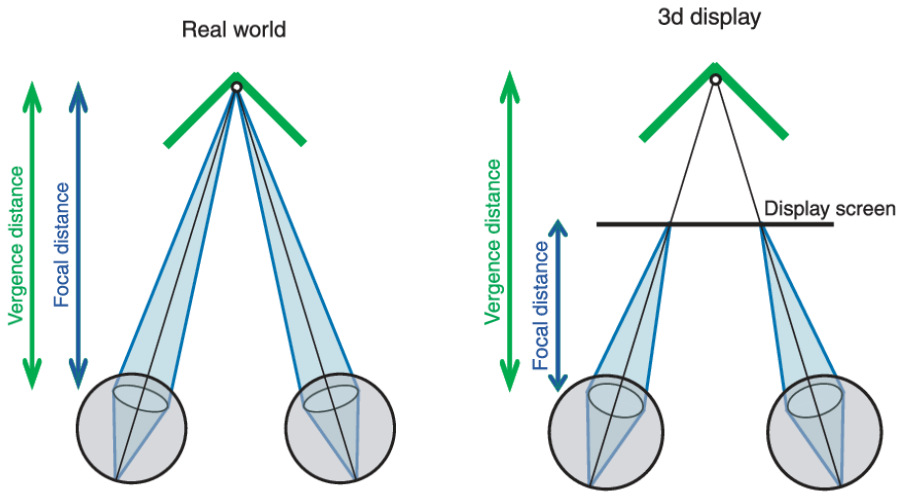
To overcome this problem it's obvious that the measurements taken from the head's pose have to match the real pose but it is as important to deliver this information really fast which means trying to achieve low latency. And as already mentioned earlier the playback of the virtual environment should have a high frame rate.

2.1.4.2 Vergence Accommodation Conflict

Accommodation is the ability of the eye to focus on objects. Humans are able to sharply see object which are closer to their eyes as well as distant ones, but it takes quite some time to focus if they switch between them.

Vergence on the other hand is the ability to move both eyes in opposite direction in order to maintain single binocular vision. This means that if

someone looks at an object placed at a certain distance, the eyes' line of sight would intersect exactly at this distance.



(a) Real World (b) Virtual Reality

Figure 2.5: Vergence and Accommodation Conflict - Adapted from Hoffman et al. Vergence-accommodation conflicts hinder visual performance and cause visual fatigue [5]

Both vergence and accommodation are depth cues and in the real world they are always in sync. This is not the case in VR. The screen is at a fixed distance to the eyes and they are always focusing (accommodating) on it. The virtual objects have different distances tough and the eyes are converging at those distances. Therefore in VR objects which are far away as well as objects which are closer both are in perfect focus which is definitely not possible in real world. This is the reason why sometimes people suffer from headache after taking off their headsets [5]. That's because the eyes have to switch back to their normal way of operating.

A solution to this problem are so called lightfields. They try to solve the problem by multiple focal planes where the eye can accommodate on. This is closer to the real world and behaviour of the human eye.

2.2 Vulkan

In this thesis a framework was used which relies on the Vulkan graphics and compute API [6]. The following sections should give a brief overview of rendering in general to gain a better understanding of the latter implementation and their corresponding benefits in respect to performance.

2.2.1 Rendering Pipeline

In general we want to make use of a powerful component in computer systems namely the Graphics Processing Unit (GPU). It is optimized to work with parallelism which perfectly suits the needs for graphics rendering. Imagine computations for every pixels on a display. In order to get an image as final output we have to run through the so called rendering pipeline and perform certain operations. A simplified overview of this pipeline ca be seen in 2.6.

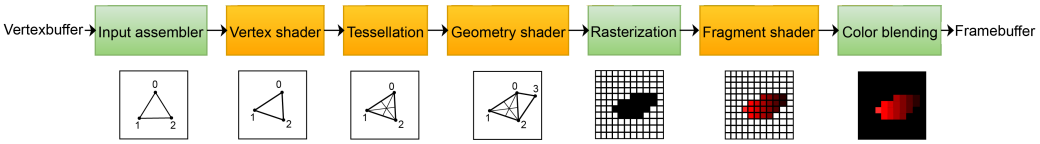


Figure 2.6: Simplified Vulkan Rendering Pipeline - Adapted from Alexander Overvoorde. Vulkan Tutorial [7]

Stages colored in green are fixed-functions stages which means they cannot be changed whereas orange stages are programmable and therefore the interesting parts of the pipeline. The most popular stages are the vertex shader and the fragment shader. Both look like simple functions written in OpenGL Shading Language (GLSL). They have input parameters and output parameters where some of them are fixed. As the name tells the fragment shader is called for every fragment in the output image. Multiple fragments result in one pixel value after color blending. In this work the focus lies on the vertex shader. This shader program processes each incoming vertex. The main purpose is to map an input vertex position which is in world space to a final position in clip space. This final position is the mandatory output for this shader. In most cases the vertex shader takes additional attributes like color or texture coordinates which are then processed in later stages. This work aims to reduce the load of the vertex shader, this means reducing the number of vertices in the scene and the number of vertex shader invocations.

2.2.2 Coordinate Systems

As already mentioned before in rendering there are different coordinate systems. They will appear more often later in the thesis when discussing the implemented model therefore I want to give a brief introduction. In 2.7 we can see different matrices and a multiplication of one of these matrices with a 3D position results in a transformation from one coordinate system to another. In object space we have the object as it is but when multiplying it with its corresponding model matrix we are able to perform certain transformations like translation, rotation or scaling. This means we can move the object around and place it somewhere in world space. The final output will be seen through a camera which is placed somewhere in the scene. Moving the camera around will give us different viewing angles and therefore different output images. The view matrix is the inverse of the camera's transformation matrix and transforms from world space to view space. The projections is responsible for removing things that can't be seen anyway due to the viewing frustum therefore is clips those areas away.

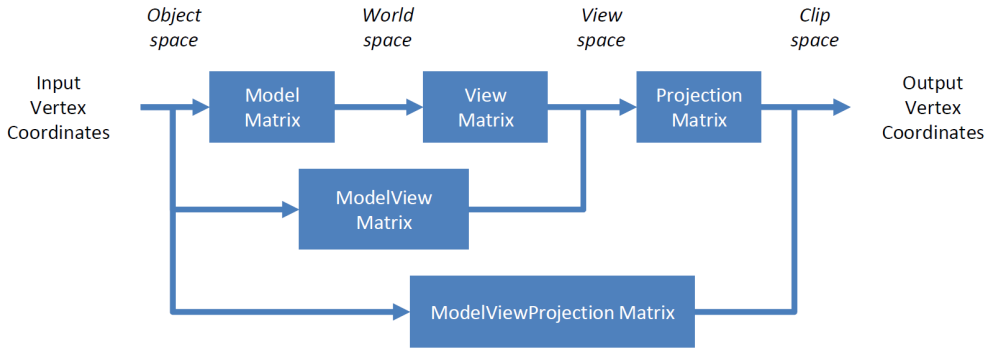


Figure 2.7: Coordinate Systems in Rendering

Therefore the vertex shader gets the 3D position of the current vertex as input in object space and multiplies it with the so called ModelViewProjection matrix which transforms this position directly from object space to the final clip space. Now we know the position of the vertex on the final output image. Another thing to mention here is that the final 2D coordinate in the output image ranges from -1 to 1 for both X and Y coordinate. Specifically for Vulkan this means that (-1, -1) describes the top left corner whereas (1, 1) is in the bottom right.

2.3 Stereo Rendering

Stereo rendering means rendering two slightly different images one for each eye [8]. This is essential in order to simulate the behaviour of the human eyes. Two eyes and respectively two rendered images enable us to gather three dimensional information. To achieve stereoscopic rendering two cameras with a slightly different viewport are used. There are two common methods how to align those cameras.

2.3.1 Toe-In Stereo

Two cameras with identical opening angle and symmetric frustum pointed towards a single focal point to models the vergence of the eyes. The advantage of this method is that it is very easy to set up but it comes with vertical parallax. Vertical parallax means that the projection of points is vertically shifted. It does not help to perceive depth and results in uncomfortable stress for the muscular system of the eyes. Therefore this method is incorrect and should be avoided. It's better to use the approach described in the next paragraph.

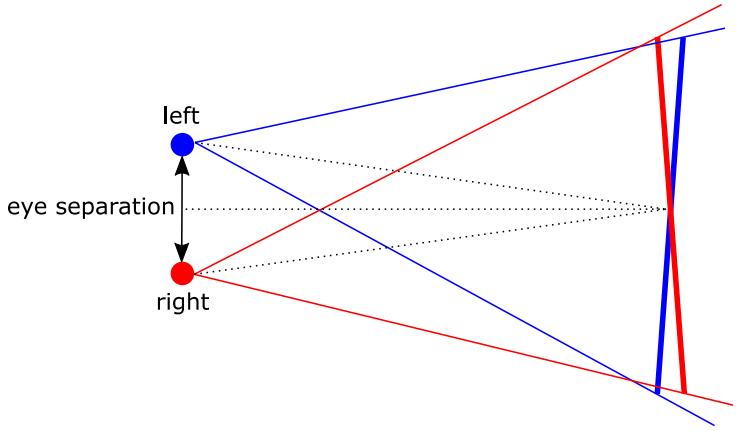


Figure 2.8: Toe-In Stereo

2.3.2 Off-Axis Stereo

The eye's rays are parallel with a fixed distance. Because of the parallel axis the projection plane is identical and therefore vertical parallax does not

occur. The disadvantage is that a standard camera cannot be used because of the asymmetric frustum. The frustum has to be manually computed.

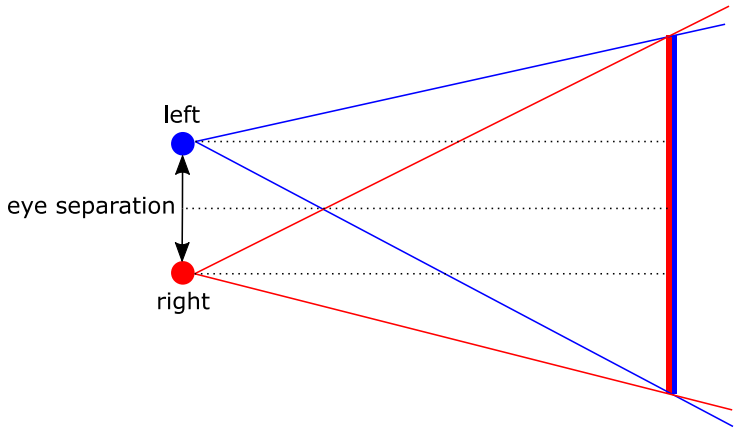


Figure 2.9: Off-Axis Stereo

2.4 Radial Lens Distortion

We already heard that lenses are needed in order to feel immersive for virtual reality applications. The downside of using lenses is that they come with image distortions. When looking through the lens the image is magnified. Magnifying an image means that pixels position on the image with a given distance to the center of the image are moved further away from the center. The amount of shifting the pixels out increases with increasing distance to the image center. This means the effect of the radial lens distortion is almost not recognizable in the center but can be easily seen towards the edge of the image. The distortion can be corrected by the use of Brown's distortion model [9] which is more commonly know as Brown-Conrady model based on earlier work by Conrady [10].

2.4.1 Brown-Conrady Model

In general the Brown-Conrady Model corrects both radial as well as tangential distortion. Tangential distortion is also known as decentering distortion. While the radial distortion is only moving points further away or closer to the center which means simply scaling the distance to the center, tangential

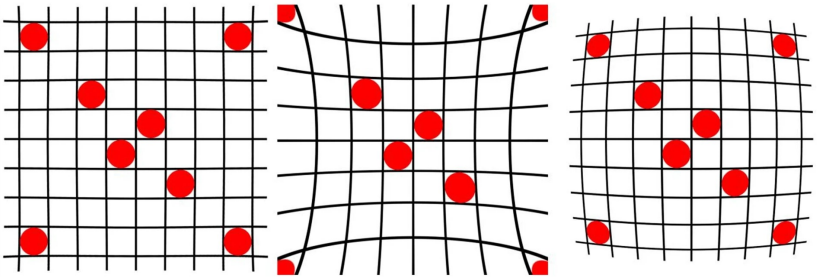


Figure 2.10: (Left) Undistorted (Middle) Pincushion Distortion (Right) Barrel Distortion

distortions moves the point to the left or to the right maintaining the same radial distance but changing the angle opened by the point and the x or y axis. For this thesis the focus lies on the radial distortion and therefore tangential distortion is not further taken into account.

The radius is zero at the image center and grows towards the edge of the screen. This means that if coordinates do not have the position (0,0) in the middle of the screen the coordinate position of the center has to be subtracted from the point of interest, respectively the point the distortion has to be calculated for. The Brown-Conrady model makes use of the following formula, assuming the center of the image is at position (0,0) for x and y coordinate:

$$r_{\text{distorted}} = r * (1 + k_1 * r^2 + k_2 * r^4 + ...)$$

where the result $x_{\text{distorted}}$ is the image point as radially distorted by the lens. The distorted radius is smaller than the original radius for negative distortion coefficients which results in a barrel distortion while it's larger for positive coefficients resulting in a pincushion distortion. The factor the original radius is multiplied with is essentially a taylor series approximating the actual optics of a lens with as many parameters k_n as wanted in order to increase accuracy. For this thesis only two lens distortion parameters were used which already yields quite good accuracy. Whereas k_1 is multiplied by the radius with lower power therefore having more impact on the distortion when the radius is rather small which results in controlling the amount of inner distortion. For k_2 it's the other way round which means k_2 control the amount of outer distortion. The original radius r can be computed as follows

$$r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$$

where x and y are the coordinate positions of the affected pixel and (x_c, y_c) is the center of the image which has to be subtracted in order to maintain (0,0) as center values. The value of the radius itself has to be normalized in order for the coefficients to be in the correct scale. This results in the radius being in the range from -1 to 1.Typical values of the distortion coefficients k_1 and k_2 can be found in a later section describing the head mounted displays which were used for the user study.

2.4.2 Lens Distortion in VR

The lenses in virtual reality headset increase our field of view and therefore they magnify the image which means image positions with a specific radius are moved out to an even larger radius. As explained this results in a pincushion distortion. As the user of the HMD does not want to look at a distorted image a compensation is applied in software. The output on the display which is seen through the lens is distorted in a way that it perfectly cancels out the lens distortion resulting in a normal undistorted image. Figure 2.11 shows the describe procedure.

For specific radial lens distortion coefficients k_1 and k_2 one can compute the pincushion distortion with the use of the Brown-Conrady model as described above. The inverse of the radial distortion which is needed for compensation is not available in closed form. With the formula the inverse would result in

$$r = \frac{r_{\text{distorted}}}{(1 + k_1 * r^2 + k_2 * r^4 + ...)}$$

but the radius r which is of interest does also appear in the divisor which is the fraction the original radius was multiplied with in order to get the distorted position.

One possible way to overcome this problem is to iteratively refine the distorted point until convergence as illustrated in Algorithm 1.

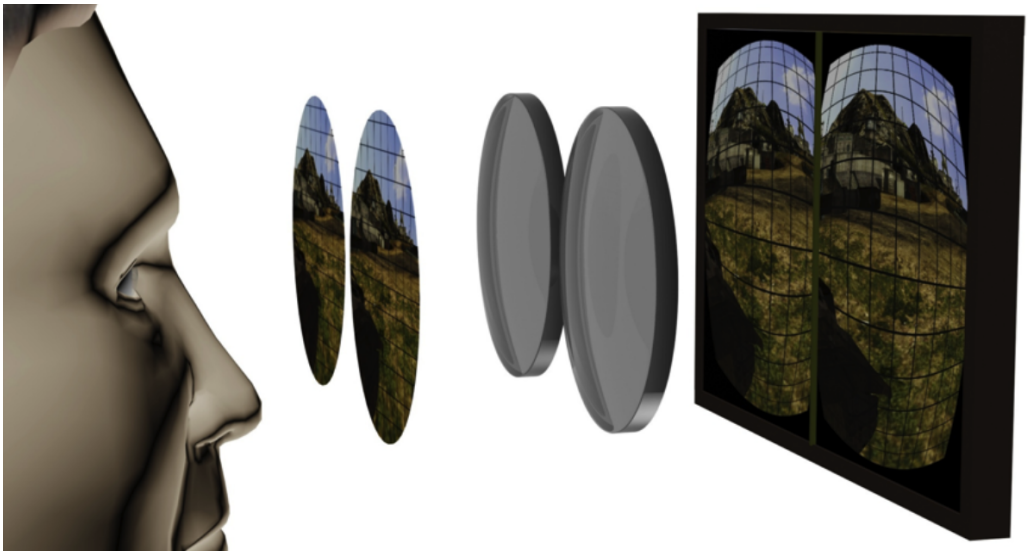


Figure 2.11: Barrel distorted image on the display cancels the pincushion distortion of the lens resulting in perfectly aligned output - Reprinted from Daniel Pohl. Virtual Reality Blog [11]

Algorithm 1 Iterative reverse lens distortion

```
1: function ComputeInverseR(r_distorted, k1, k2)
2:     r_undistorted = r_distorted
3:     while r_undistorted not converging do
4:         r = ||r_undistorted||
5:         d_r = (1 + k_1 * r^2 + k_2 * r^4 + ...)
6:         r_undistorted = r_distorted / d_r
7:     end while
8:     return r_undistorted
9: end function
```

The second method is to simply save a lookup table which saves pairs of values containing the distorted r value and the fraction it was multiplied with. This way the radius we want to use for the images shown on the display can be acquired via the look up table which is way faster than the first approach. For this thesis those calculations will be done twice for each object visible in the viewing frustum for each frame. As performance could be an issue there obviously the second approach with the lookup table was chosen.

This way a position on an undistorted image with radius r is first moved towards the center by a fraction f and shown on the display. By viewing the display via the lens this radius is moved away from the center by a fraction f resulting in the original radius and therefore undistorted as in the original image.

2.4.3 Rendering Lens Distortion

After getting to know how the barrel distortion is mathematically applied the following section will show how to apply distortion in respect to rendering. In this case the three most common methods are briefly described below.

2.4.3.1 Texture lookup in fragment shader

For this approach the whole scene is rendered to a large texture. In the fragment shader a lookup into this texture is performed in order to obtain the color value in its undistorted location. As this is done for every pixel in the image this is a quite taxing approach.

2.4.3.2 Texture to grid assignment

The second method is similar to the first one but the lookup is not performed for every pixel in the image but for each vertex of the grid. The quality of course depends on the number of vertices used for the grid.

2.4.3.3 Distorting geometry

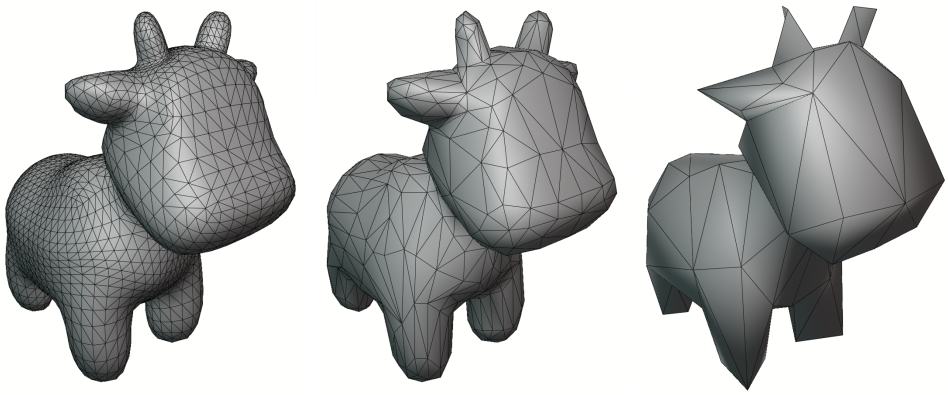
The last and most complex approach is distorting the geometry itself as it is being rendered. This requires and extremely high level of geometry tessellation in order to avoid significant aliases.

The framework used for this thesis implemented the second approach. The lens distortion effect is applied as a post processing step. Everything is rendered normally and in the end the final image is saved into a large texture which is then used for the grid morphing. - perceptual models - eye tracking techniques - human visual vision

2.5 Level of Detail

As already discussed in previous chapters one compulsory requirement for VR applications is a constant and high frame rate. On the other hand a rich and highly detailed graphical world supporting the realism of immersion is quite important as well. The computational power of the graphics hardware is limited, therefore a tradeoff between the mentioned points has to be made. One way to keep the frame rate high while maintaining an attractive graphical world is by regulating the amount of detail used to represent the virtual world.

Three dimensional meshes in computer graphics are often very complex which means they consist of a very high number of polygons describing the surface and shape of the geometry. The higher the number of polygons the higher the computational power required to render the object. Levels of detail are used to overcome this issue. In this case lowering the level of detail means varying the geometric resolution resulting in simplified less complex meshes by reducing the number of polygons.



(a) 100% polygons (b) 20% polygons (c) 3% polygons

Figure 2.12: Mesh simplifications

There are different approaches when to use less detailed version of the object, e.g. smaller or distant objects or objects which are placed in unimportant parts of the screen. Whenever a specified criteria to use a less detailed version is met the resulting level is used for rendering.

2.5.1 Level of Detail Frameworks

In general three different basic frameworks for managing levels of detail exist [12]. The next section briefly describes those three approaches and their advantages as well as disadvantages.

2.5.1.1 Discrete

Discrete level of detail describes the traditional approach originally proposed by James Clark in 1976 [13]. For each object in the scene a certain number of LODs are pre-computed and the selection of the correct level is done at runtime. The most common criterion for LOD selection is the distance to the camera. The larger the distance to the camera the smaller the object appears on the screen and therefore less detail is needed.

Because the creation of the LODs is done as an preprocessing step the creation process is completely decoupled from the rendering and therefore does not have to meet any real-time or specific performance constraints.

There are cases where the object itself does not really suit this approach. If objects are very large the distance to the camera is computed once for the entire object, whereas the distance can vary a lot depending on which point of the object is taken into account. A solution for this would be to subdivide the object and select LODs for each subdivision. This results in several different LOD selection within one large object. The best example here is to imagine a large terrain object which obviously somehow has to be split in order to select less detailed version for far regions.

Although the level of detail frameworks described in the next chapters come with advantages the traditional approach is still the by far most commonly used one in practice. It's the simplest approach and works best with most current graphics hardware. Discrete LOD is used in this thesis.

2.5.1.2 Continuous

Continuous level of detail continuously tries to find the perfect LOD at runtime. This means the grade of detail is not chosen from any number of pre-computed models but directly calculated. This way every object in the scene uses exactly the number of polygons it should use. This results in overall better resource utilization. While with the traditional approach it can be happen that visual popping effects can be recognized when switching

between different LODs, this is not the case for continuous level of detail as the detail is adjusted gradually and therefore reducing visual pops.

2.5.1.3 View-dependent

The last of the three frameworks is an extension of continuous level of detail where the current viewing parameters are also taken into account for the LOD selection. The best selection should be made for the current view. One criterion could again be the distance to the eyepoint. Another criterion could result in showing silhouette regions of object with more detail than inner regions. One last example deals with the aspect of the user's focus. More detail is needed where the user focuses his vision whereas less detail is needed in the periphery.

The most popular application for view-dependant level of detail which almost requires this technique is in terrain rendering. Depending on the view a large terrain spans multiple levels of detail.

Continuous and view-dependant LOD provide better fidelity but also require more memory and increased run-time performance.

2.5.2 Generating Levels of Detail

In order to produce different levels of detail for the mesh of an object the mesh has to be simplified. The geometry of each mesh is represented by vertices while the connectivity is represented by edges connecting the vertices. By removing vertices or edges the mesh is simplified. Mesh decimation in general is a greedy iterative approach to gradually reduce the complexity of the mesh as shown in Algorithm 2.

Each decimation performed simplifies the mesh by a small amount. This is done until no further decimations can be performed due to a error threshold describing the desired amount of simplification. Different mesh decimation operators exist which are briefly described in the following sections.

2.5.2.1 Vertex Removal

One vertex is selected and completely removed from the mesh. As the vertex is removed the edges connected to this vertex are also removed resulting in a hole. This hole is then filled with new triangles via re-triangulation.

Algorithm 2 Mesh decimation

```
1: for all regions in the mesh do
2:     Measure error after decimation
3:     queue(region, error)
4: end for
5:
6: while reduction possible do
7:     Get best decimation in queue
8:     if error < limit then
9:         Perform decimation
10:        Re-evaluate error metrics
11:    end if
12: end while
```

2.5.2.2 Edge Collapse

This approach was first introduced by Hoppe et al. [14] in 1993. By collapsing an edge in the mesh the two vertices connected by the edge are merged. This operation removes all triangles which share the collapsed edge. The two merged vertices result in one final vertex. The inverse of the edge collapse is a vertex split. If the final vertex is one of the initial vertices the operation is called half-edge collapse and is actually a special case of the vertex removal. Vertex removal and edge collapse both require the mesh around the vertex to be manifold. If a mesh is manifold one edge can only be shared by one or two faces and faces connected to a vertex can only form an open or closed fan. Furthermore both operations preserve local topological structures which means the layout of the wireframe stays the same. So both operations have some properties in common.

Edge collapse is a quite simple operation but care has to taken because mesh foldovers can happen. This means that if an edge is collapsed the resulting vertex connects to another vertex via an edge which is crossing another edge. This did not happen before the edge collapse because the position on the vertex was somewhere else. As this results in a crease in the surface of the geometry it can be detected by measuring the change of the normals of the triangles of the mesh. A large change, usually larger than ninety degrees is a sign for a foldover.

2.5.2.3 Vertex Clustering

A cluster is applied and all vertices in the cluster will be represented by one final vertex. Different from the first two techniques vertex clustering can also handle non-manifold meshes. It also allows topological changes. In general it is a very fast and robust approach but it is hard to control the simplified mesh and may not achieve best quality.

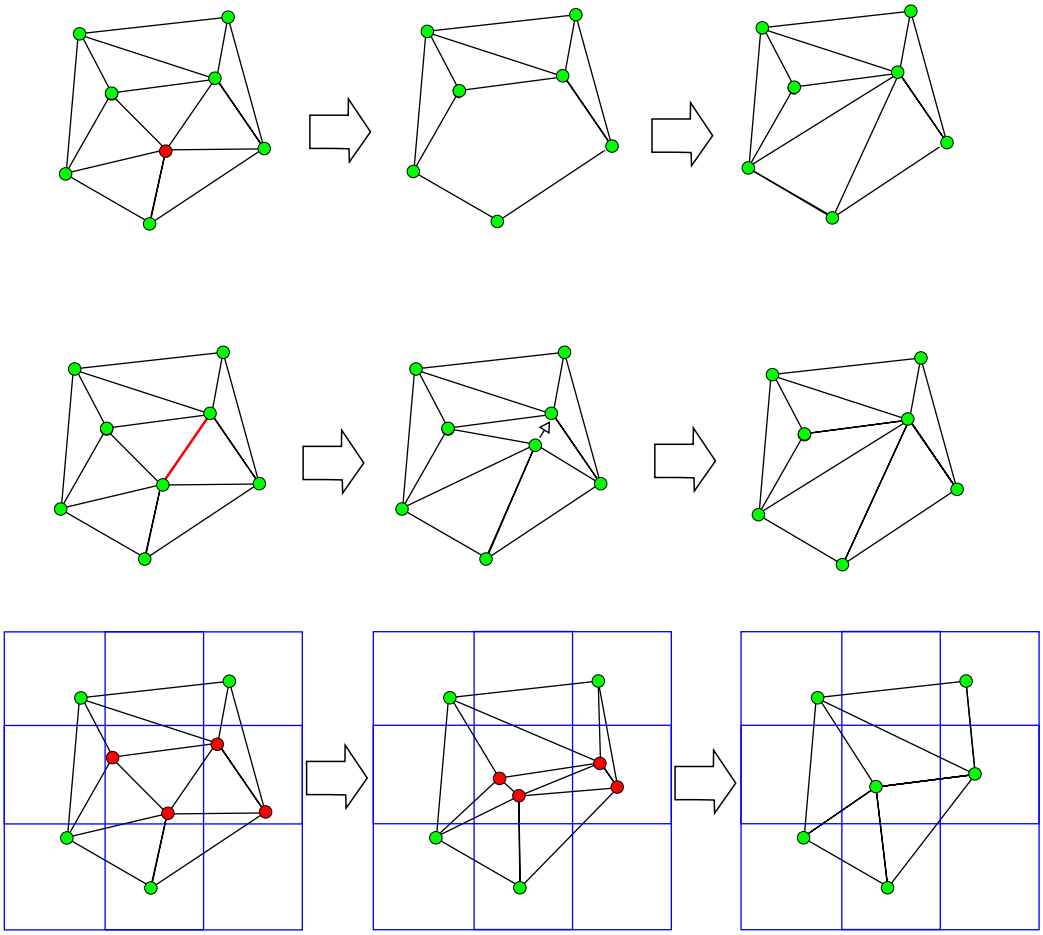


Figure 2.13: (Top) Vertex Removal (Middle) Edge Collapse (Bottom) Vertex Clustering

2.5.3 Choosing Level of Detail

After discussing how LODs are created this section deals with the selection of a desired amount of detail at run-time. Which metrics influence the selection and how the measure of difference between original model and any LOD is performed.

2.5.3.1 Fidelity Metric

If the selection of LODs should not be done by hand as it was commonly done in the past a certain metric is needed to automatically choose the amount of mesh simplification.

Fidelity-based simplification
The user can choose a desired fidelity of the simplification and then the algorithm should choose the number of polygons accordingly without violating the set constraint.

Budget-based simplification
In this case the user can set a specific number of polygons which should not be exceeded. Then the fidelity of simplification is being maximized without violating the constraint of maximum polygons.

In most cases a combination of fidelity-based as well as budget-based simplification is the best choice.

2.5.3.2 Measuring Error

Now that the two approaches are known the question is how can fidelity be measured? Most algorithms use some sort of geometric error which means the geometric difference between the original model and a certain level of detail. The perceptual error on the other hand should be of higher interest as it tells if the simplified model looks the same as the original one.

Geometric Error
There are various different approaches from which two were picked in order to understand the basic principles. One method is the vertex-vertex distance, where the distance of each vertex connected to the vertex which is being

removed is measured and the max of all measured distances is taken as error metric. This technique was first introduced by Rossignac and Borrel in 1993 [15].

Another metric is resulting from surface-surfaces distances. In this case the error is the maximum of all distances between the surface of the original mesh and the simplified one.

Perceptual Error
As already mentioned it may be more interesting to investigate the visual differences. This is also where the work of this thesis has its focus on. The selection of LODs depends on their scores reached which are measured by the Structural Similarity Index, from now on abbreviated by SSIM. This metric is further described in later chapters.

2.5.3.3 LOD Selection

The last section in this chapter about LOD management deals with the question of when the switching to simpler meshes should happen. When and under which circumstances is less detail sufficient? The following listed factors can all influence the LOD selection:

- Size
  One of the first things that come to ones mind as a factor for LOD selection is the size of the object. Smaller things most probably need less detail than larger objects. Size can be measured in e.g. pixels occupied.

- Distance
  Another obvious factor is the distance to camera of the scene. If the object is far away it is probably less important and can be rendered with less detail. With increasing distance the object also gets smaller which is what was described in the previous factor. But object can of course have different sizes with equal distance (e.g. an elephant and a mouse).

- Priority
  Object can also be prioritized depending on their semantic importance in the scene. Is one specific object more important to the user than

others? As an example objects in the periphery could be less important than the ones in the area of focus.

There are of course a lot of other factors influencing the LOD selection but the above mentioned ones are one of the most common and should give a good idea of what could be an aspect choosing less detailed meshes.

LOD Selection for VR
Another aspect this work should take into account is an additional factor for LOD selection especially suited for virtual reality applications. As the user should most likely have the focus in the middle of the screen, which is even more true for VR as head movements are more probable than only moving the eyes, using eccentricity for LOD selection has already been proposed a long time ago [16].

Most of the approaches taking the eccentricity as an additional factor have an psychological background, dealing with the user's attention on the screen. What is proposed within this thesis is not a psychological approach but a technique making use of the optics of the lenses in VR systems. As already heard in previous chapters lenses in VR HMDs introduce distortions which are compensated in software. The compensation distorts the final rendered image in a way that object at the edge of the screen are a little smaller than the ones in the center. As mentioned before the size of the object could be one factor for the LOD selection and smaller objects would need less detail which can now also be applied by the eccentricity in VR headsets. How much smaller the objects get via the distortion and how the shrinking can be calculated is part of one of the next chapters dealing with the implementation of this idea.

2.6 Structural Similarity Index

As already mentioned in previous chapters there are certain measures to rate LODs and use the desired one for the current state of the rendering. For this thesis the measure used is the Structural Similarity Index, from now on abbreviated by SSIM. This metric was first introduced in 2004 by Z. Wang, A.C. Bovik, H.R. Sheikh and E.P. Simoncelli [17] and gained significant popularity in the field of image processing since then.

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{2.1}$$

μ_x	the average of x
μ_y	the average of y
σ_x^2	the variance of x
σ_y^2	the variance of y
σ_{xy}	the covariance of x and y
c_1, c_2	constants

where the values of the constant variables where chosen to be $c_1 = 0.0001$ and $c_2 = 0.0009$ in our implementation. The purpose of those constants is to stabilize the division with a weak denominator.

The SSIM index consists of three different comparison metrics which are described below.

2.6.1 Luminance

$$l(x,y) = \frac{2\mu_x\mu_y + c_1}{\mu_x^2 + \mu_y^2 + c_1} \tag{2.2}$$

In this case c_1 is important to overcome instability when μ^2 for x and y is close to zero.

2.6.2 Contrast

$$c(x,y) = \frac{2\sigma_x\sigma_y + c_2}{\sigma_x^2 + \sigma_y^2 + c_2} \tag{2.3}$$

2.6.3 Structure

$$s(x,y) = \frac{\sigma_{xy} + c_3}{\sigma_x\sigma_y + c_3} \tag{2.4}$$

with $c_3 = \frac{c_2}{2}$.

The combination of theses three metrics results in the following equation:

$$SSIM(x, y) = [l(x, y)]^{\alpha} \cdot [c(x, y)]^{\beta} \cdot [s(x, y)]^{\gamma} \tag{2.5}$$

where alpha, beta and gamma are weights for each part of the metric as described above. With even weighting e.g. $\alpha = \beta = \gamma = 1$ the formula reduces to the form as given by equation 2.1.

2.6.4 SSIM in Images

The previously explained equations to calculate the SSIM score always take two inputs x and y which are compared. For this thesis the inputs are two slightly different images with the same dimensions in height and width. There are different approaches to calculate the similarity score between two input images. The difference between each pixel in image x versus the corresponding pixel in image y can be compared but it is recommended to calculate the score over a window of pixels. A window size of 10x10 pixels was used for this work. Additionally the weight of each pixel was not equally distributed in the window but a Gaussian kernel was applied which leads to the pixel in the center of the windows being the most important one with the largest weight. Furthermore in the end the average of all pixels was taken as the final score.

2.6.4.1 SSIM for Objects in Images

As for this work the interest was not the difference between two images but the difference between two objects in those images only the pixels occupied by the corresponding objects were taken into account for the calculation of the SSIM score as described above.

2.6.5 Comparison to MSE

Another metric commonly used in image processing is the mean squared error, abbreviated by MSE. This metric simply takes the squared difference for each pixel in two images and uses the mean as a result. This results in the formula given in 2.6.

$$MSE(x, y) = \frac{1}{n} \sum_{i=1}^{n} (x - y)^2 \tag{2.6}$$

The disadvantage of MSE is that there is no relation to real visual differences in two comparing images. That's where SSIM outperforms MSE which can also be seen in figure 2.14.

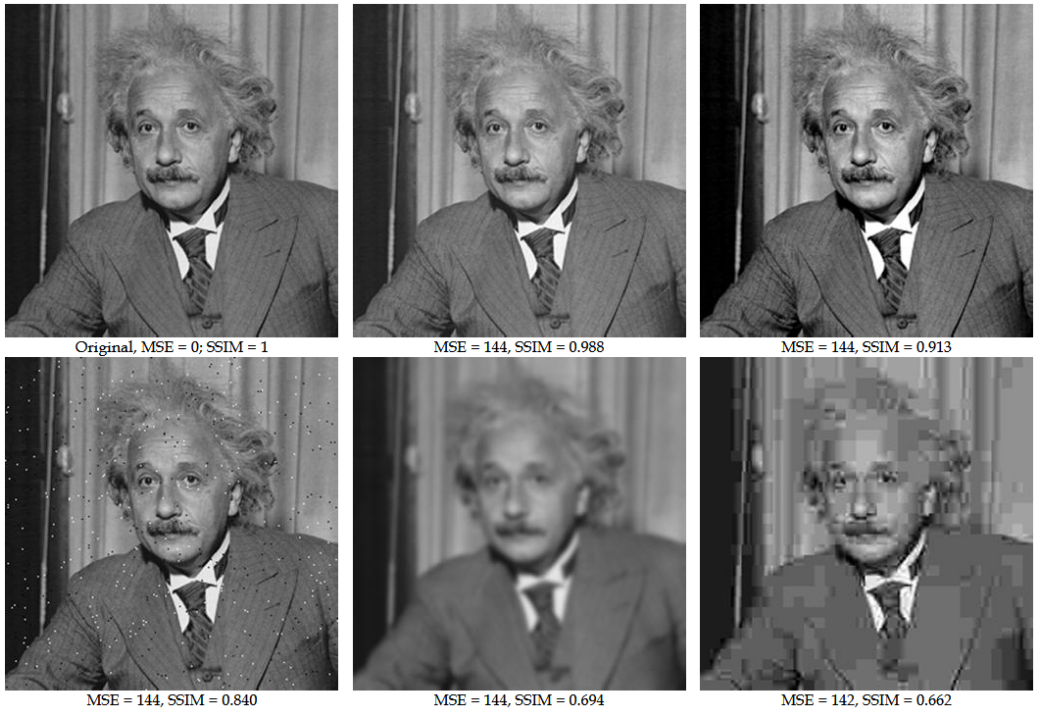


Figure 2.14: All images with same MSE but different values for SSIM - Reprinted from Z. Wang. et al. The SSIM Index for Image Quality Assessment [18]

The first image in figure 2.14 is the original one and in all others some sort of distortion was applied. All distorted images have the same value for MSE. The values for SSIM were different and closer to real visual differences. The following distortions were applied to the images:

- mean-shifted
- contrast-stretched
- salt-pepper noise

- blurred
- JPEG compressed

2.7 Vector Streaming Framework

All the work for this thesis was implemented in the vector streaming framework [19] which is mainly used by the researchers of the institute of computer graphics of the technical university of Graz. This section should give a brief overview of some important parts of the framework.

The framework itself is written in C++ and uses the Vulkan API [6] for rendering. OpenGL Mathematics (GLM) [20], which is a mathematics library especially for graphics software, as well as the Graphics Library Framework (GLFW) [21], which is commonly used for vulkan development providing and API for creating and setting up windows and similar things, are included in the frameworkg.

2.7.1 Basic Structure

The framework consists of multiple main components each with its own responsibilities in order to create the final rendering. Those components are listed below with a short description of the corresponding tasks.

- Scene Manager
  The scene manager is the main component and is responsible for loading and setting up all other manager listed below.
- Entity Manager
  Each object in the scene is entitled as entity. It consists of a desired number of models each with its own mesh and a material as well as a model matrix defining the position in the scene. While rendering the entity manager iterates through all entities and check their visibility via view frustum culling and then sets the correct level of detail for each visible object according to the current LOD selection criteria. This is exactly where the main code of this work can be found.
- Model Manager
  This component is responsible for managing all the models of one entity.

One entity can have arbitrary number of models which correspond to the same entity, but each additional model with less detail in term of polygons in the mesh, compared to the base model.

- Material Manager
  Each entity has a material. A material can be used by multiple entities and the material manager is responsible for the mapping. In general there are three different material properties which differ in the used shader for calculating the final color. The first one uses simple phong shading and mainly depends on the given base color of the object. The next material is used to render metallic effects. The third shading technique more complex and requires image based lightning to introduce specular effects.

- Light Manager
  Another important thing for each scene is the lighting setup. That's what the light manager is responsible for. It manages the scene's lighting which consists of three different light types which are spot lights, point lights and directional lights. Furthermore it is responsible for the image based lighting which is important for the object's materials as explained above. Another thing which is done by the light manager is setting up an environment. This is done by a cube map with a specific image for each side of the cube.

- Display
  For the final output three different kinds of displays can be chosen. One is an OpenVR display which uses the connected headset for rendering. The other two are both for rendering the scene directly on the computer screen. One is a mono display which outputs the renderings in the usual way as seen from one viewport while the stereo display outputs the scene seen from two slightly different viewports which are the two eyes. What can be seen when using the stereo display is what is actually seen when wearing a VR headset. Furthermore the stereo display uses a grid warping technique to render the barrel distortion effect.

2.7.2 Scene Setup

Every scene in the vector streaming framework is set up via configuration files describing the scene. A desired number of .cfg files can be concatenated to

produce the final output. Those files store data which is used to be loaded for every component explained in the previous section. A sample configuration file can be seen in figure 2.15.

```
configs = {
    ibl = "environments/ibl_sun_temple.cfg"
    sky = "sky.cfg"
}

scene = {
    ambient_light = {
            r = 0.5
            g = 0.5
            b = 0.5
        }
}

rendering = {
    view_frustum_culling = 1
    fovy = 110.0
    znear = 0.1
    zfar = 1000.0
}

output = {
    width = 1440
    height = 1600
}
```

```
scene = {
    materials = {
        spot_phong = {
            texture = "lod_scene/models/spot/spot_texture.png"
        }
    }
    entities = {
        spot = {
                base = {
                    mesh = "lod_scene/models/spot/LOD_1.0.obj"
                    material = "spot_phong"
                    submesh_id = 0
                }
                level_1 = {
                    mesh = "lod_scene/models/spot/LOD_0.5.obj"
                    material = "spot_phong"
                    submesh_id = 0
                }
                level_2 = {
                    mesh = "lod_scene/models/spot/LOD_0.3.obj"
                    material = "spot_phong"
                    submesh_id = 0
                }
        }
    }
}
```

Figure 2.15: Sample Scene Configuration File

The first part of the configuration file references other files to setup the image based lighting. Afterwards an ambient light with the specified color is declared. The next section has some rendering related information such as the field of view. Then the size of the output windows is specified. On the right side of figure 2.15 one entity is declared which should then be visible in the final rendering. It is an entity called "Spot" with a base model and two more levels of detail each with their own meshes and a material which is specified above. The material uses the standard shader which is phong shading and has an image used for texturing the object.

2.7.3 OpenVR

The framework is also able to render scenes for specific head mounted displays. The interface used to access the hardware is an API and runtime called OpenVR [22]. It allows the access of different headsets without requiring the application to have specific knowledge of the connected hardware. It uses SteamVR [23] in order to detect and connect to headsets.

Chapter 3

Method

After getting to know all the details about VR in general and why we need lenses in HMDs and which advantages as well as disadvantages they come with, this section should present a model to increase the performance of VR renderings by taking into account the effects of the lens distortion compensation. As already mentioned a barrel distortion is applied in software which distorts the image in a way that objects with a larger radial distance to the center of the lens are smaller in size, because of this fact the level of detail required decreases with increased eccentricity while maintaining the same distance to the camera.

3.1 System Diagram and Components

Figure 3.1 describes the relations of all the different factors which are used to determine the desired LOD at runtime. There are three main groups which are parameters of the display in the headset, runtime parameters as well as an input file which should be available for each object in the scene.

3.1.1 Display Parameters

There are three different types of display parameters which all contribute to the selection of the correct LOD.

- **Resolution**
  The resolution plays a major role because with higher resolution there

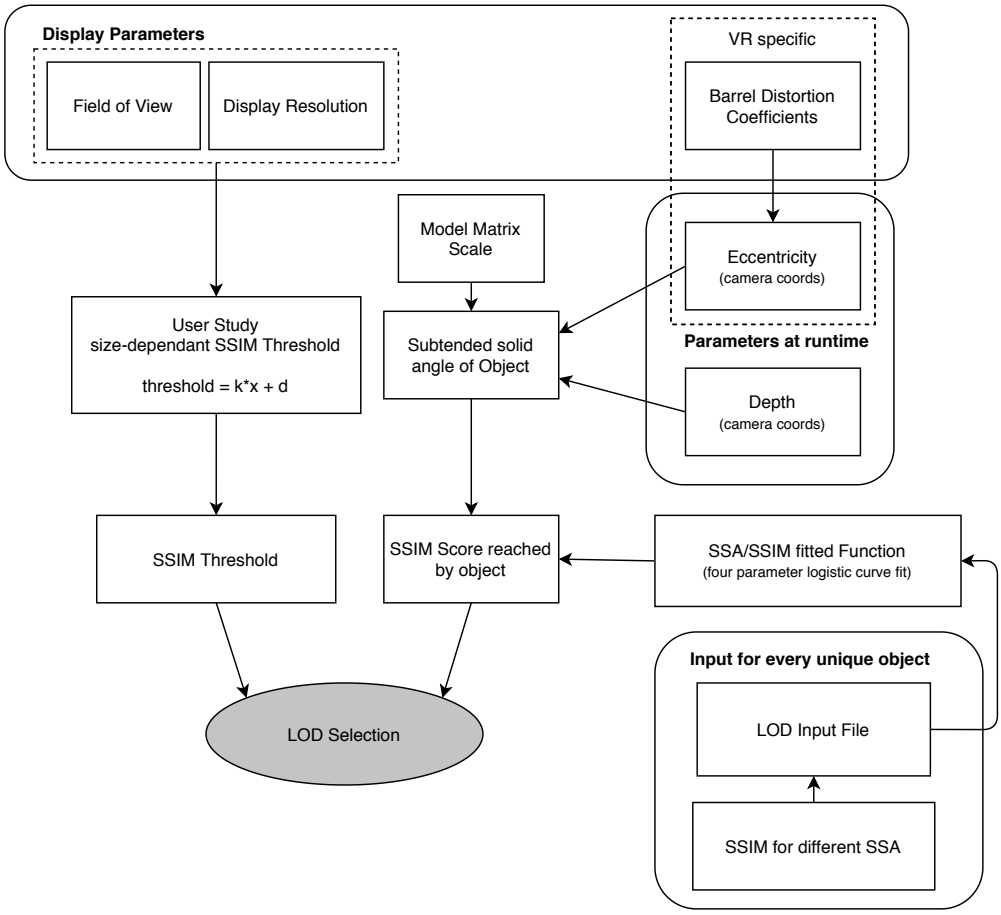


Figure 3.1: LOD Selection System Diagram

are more pixels available to render the object. More pixels mean that the object can be rendered with more detail.

- **Field of View**
  When the field of view is increased more information has to fit on the same screen which means less pixels are available per degree of field of view. While maintaining the same resolution an increase in field of view results in less pixels available for a certain object and therefore it can be rendered with less detail.

- **Distortion Coefficients**

The most important factor for this work is the VR specific one describing the amount of distortion needed to compensate the distortion introduced by the lens. The coefficients itself were already explained in earlier chapters.

3.1.2 Parameters at Runtime

The following parameters are acquired at runtime because they are depending on the viewing parameters of the camera for the current scene. As also seen in the system diagram the model can be scaled via the model matrix, which obviously changes its size and therefore the LOD selection. Non rigid transformation though can also be applied via the model matrix but will cause problems because shape of the object in completely changed. On the one hand this results in a new bounding sphere and on the other hand this would also change the SSIM scores used later for the LOD selection.

- **Depth**
  Describes the distance between the object and the camera. This distance is calculated once per object. Every object has a bounding sphere and the point of the bounding sphere closest to the camera is taken for further calculations.

- **Eccentricity**
  The eccentricity is the distance to the center of the screen expressed in either radial distance or in degrees of an angle.

3.1.3 Object specific Input

Another thing needed in order to evaluate the LOD is an input file describing the behaviour of the SSIM scores. This file is evaluated for each object in the scene and has to be available at runtime. More information about the creation of this and the containing data is described in section 3.3. In general the object specific input serves for the evaluation of the reached SSIM score for the current settings and for each level of detail for the specific object.

3.1.4 Structural Similarity Score Threshold

As a last step for the evaluation of the sufficient level of detail needed for the current runtime parameters as well as the current display configuration

the score reached by a specific object is compared to a SSIM threshold. This threshold is not a fixed value but changes with decreasing subtended solid angle given by a linear function.

3.1.4.1 Size-dependant Threshold

After several observations of different objects and settings a conclusion was made that a fixed threshold is not sufficient. A linear relation was expected and therefore a function of the form

$$y = k * x + d \tag{3.1}$$

was used to calculate the SSIM threshold for different sizes respectively subtended solid angles. To evaluate the parameters k and d a user study was performed which is described in more detail in section 5.1.

3.2 Creating LODs

In previous chapters certain different techniques to simplify a mesh were explained in order to create levels of detail of any desired object. The mesh simplifications respectively the creation of LODs were done using Blender 2.82. Blender has different modifiers which can be applied on a meshes and one of them is a modifier called Decimate [24].

3.2.1 Blender Decimate

This modifier has three different types of mesh simplification approaches to choose from.

- **Collapse**
  Vertex collapse techniques merging vertices together as described in section 2.5.2.2.
- **Un-Subdivide**
  This is more or less the reverse step of a subdivide modifier which is trying to remove edges introduced by a subdivide operation. This is mainly used for geometry with flat surfaces.

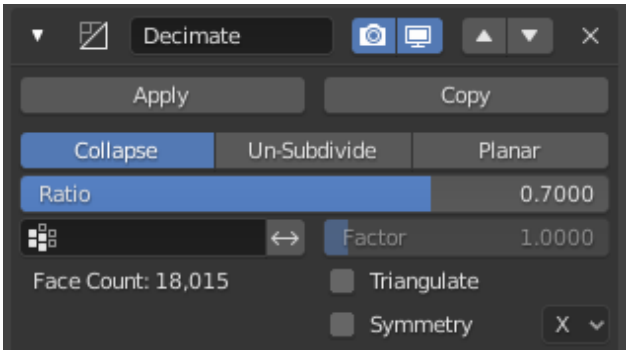


Figure 3.2: Blender Decimate Modifier

- **Planar**
  An angle limit can be set for this approach and geometry forming angles between surfaces higher than the given setting is dissolved. This achieves good results mainly for almost flat surfaces.

For this work the first decimate type is suited well and was therefore chosen to create the LODs. When merging vertices together the collapse type also takes the overall shape of the mesh into account.

The collapse approach has one parameter called ratio which is used to control the amount of mesh decimation. It is in the range from zero to one. Setting it to one results in the mesh being unchanged and setting it to e.g. 0.7 results in a mesh where seventy percent of the original number of faces remain. As seen in figure 3.2 the remaining face count is already displayed while configuring the parameters for the simplification.

The modifier can also be applied on different sub parts of the mesh. This is done via a vertex group but was not needed in our case. Another option is to keep triangulated geometry which results from the decimation process but as our input meshes are already triangulated this option does not have any effect. One more setting is aiming to maintain symmetry along a desired axis but this setting was not used either.

3.2.2 Automation via Python Script

The process of applying the decimate modifier was automated by running a python script which can be directly fed into blender. The script takes each mesh in the blender scene and creates one output obj file for each decimate

ratio defined in the script. The output files can then be directly used LODs for the given object.

3.3 Object Specific Input

The system diagram that was explained before needs an object specific input which will be the topic of this section. First the procedure in order to create those input files as well as the files content is discussed followed by explanations which LODs to pick and which ones to discard for the final rendering framework.

The general intention is to create one input file per object describing the behaviour of the SSIM scores over a certain amount of given subtended solid angles. This is done for each level of detail generated from the original model. The input file is then used at runtime in order to get the scores reached by every LOD for the current viewing parameters. The scores are then compared to the current SSIM threshold and a LOD selection can be made.

3.3.1 SSIM Inputfile Test

The SSIM score in images as already discussed in section 2.6.4 is the mean over all scores for each pixel of interest which are the ones related to the object. As one criteria is that switching to models with lower complexity should start when the object is smaller than one sixteenth of the screen size, the creation of the input file data also started at this specific subtended solid angle which is calculated as shown in equation 3.2. Furthermore the object is placed exactly in the middle of the screen.

$$SSA(\frac{1}{16}) = 2\pi(1 - \cos(\arcsin(0.25 * \tan(\omega))) \tag{3.2}$$

By starting at this subtended solid angle another advantage is that the bad stretching effects of perspective projection do not kick in as the object is in the screen's center and rather small. The distance between the camera and the object is now increased in linear depth steps. The score is evaluated for every distance.

It is possible that objects are more complex on a certain side than on others. For this reason, every object is rotated in order to be seen from

different viewing angles to reveal otherwise unseen more complex areas. The object is rotated around four different axes with a step size of ten degrees per axis resulting in 144 scores per distance and LOD. The chosen axes were a horizontal and a vertical one as well as combinations of those resulting in two diagonal axes.

It was observed that the gathered scores for the different axes were fluctuating quite a lot. This is probably due to rendering artifacts especially for larger depth where the object really small. What was desired in theory was the worst scores of all rotations as this way it is not possible to see the object from a worse viewing angle and the quality of the chosen LOD will always be sufficient. In order to avoid scores the 25^{th} percentile was taken. This measure yielded good results and removed a huge part of the noisy values.

The percentile is defined as follows

$$x_p = \begin{cases} 0.5(x_{\text{np}} + x_{\text{np}+1}), & \text{if } np \in \mathbb{Z} \\ x_{\lfloor \text{np}+1 \rfloor}, & \text{otherwise} \end{cases} \tag{3.3}$$

with n being the number of values in the pool and p the percentile of interest. In case of the input file test procedure the interest was in the 25^{th} percentile which means $p = 0.25$ from a pool of $n = 144$ values. Resulting in $n * p = 36$ which is an integer therefore the score used is the average of the 36^{th} and the 37^{th} score.

3.3.1.1 Inputfile Data

After generating all the data as described with the procedure before an input data file consists of three values per line which are the following:

- Level of Detail
- Subtended Solid Angle
- Structural Similarity Score

This means that scores are available for every level of detail of the object and a certain amount of different subtended solid angles which is everything needed for the LOD selection. The distance between camera and object of interest was linearly increased leading to a higher number of data points for smaller subtended solid angles. When the distance between camera and

object is doubled the size of the object in terms of width and height is halved but the number of pixels which corresponds to the subtended solid angle is a fourth of the original one. This leads to the following relation: $SSA = \frac{1}{depth^2}$. The data points were fitted which is subject to the next section.

3.3.1.2 Fitting the Data

In order to use the generated data for the LOD selection framework it was fitted with the use of four parameter logistic regression [25], abbreviated 4PL.

$$y = d + \frac{a - d}{1 + (\frac{x}{c})^b} \tag{3.4}$$

with the four parameters denoted a, b, c and d as well as the input x which is the subtended solid angle resulting in the value y being the score. The result of the fitting as well as the data points generated by the test procedure can be seen in figure 3.3.

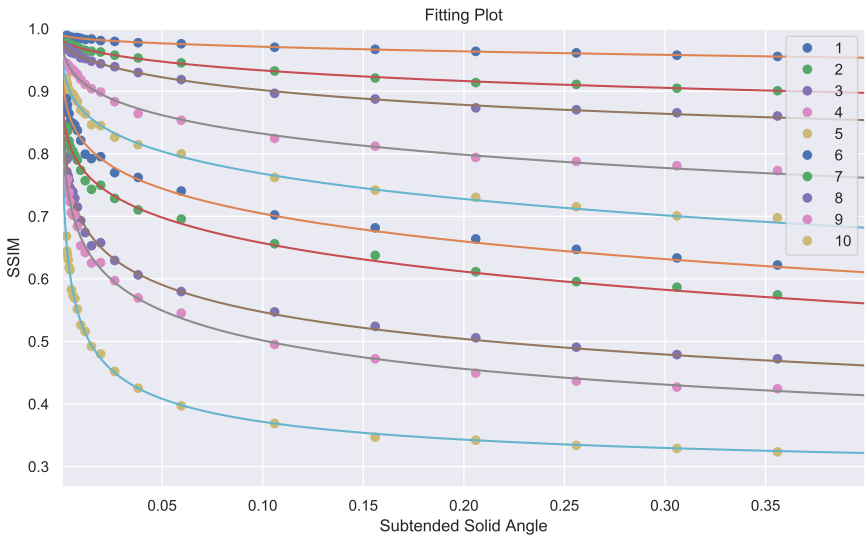


Figure 3.3: Sample LOD Input File with 4PL Fitting

3.3.2 Elimination Criteria

A certain number of LODs can be created as explained in section 3.2 with different reduction ratios. The reduction ratio is not directly relatable to the score reached by the LOD. Therefore a fixed amount of LODs is generated with a certain step size for the reduction ratio and only a subset of those is used for the LOD framework. In order to choose the subset the object is set to have a size of one sixteenth of the screen size and the reached score is evaluated. A certain step size is SSIM score is set, for example five percent and the subset of LODs is picked accordingly. Figure 3.3 already shows only a picked subset.

3.3.3 Appearance Parameters

Every object has a material assigned used to determine the final color of the surface. Different settings can be made to change the material. As every setting slightly changes the appearance of the object it also influences the SSIM score. The following appearance parameters were investigated regarding their influence on the final SSIM score and the described behaviour was observed:

- metallicity
  With increasing metallicity, the score decreased. Higher metallicity leads to more reflective surfaces, which means edges can be seen more clearly, and this was the assumption for the lower scores.

- specularity
  This parameters was showing the same behaviour as the metallicity, with the same assumption as stated above.

- texture frequency
  The higher the texture frequency, the lower the score. An increased frequency leads to a more complex texture, which then leads to more differences between LODs and therefore to a lower score.

- texture contrast
  A higher contrast in the texture means that again the texture will be visually perceived as more complex, which ends up in the same assumptions as for the texture frequency.

The behaviour described above will most likely be the same for every object, therefore a future improvement could be that a correction factor is introduced which depends on the material used and corrects the final score. This was the intention while investigating the material parameters, but in the end the conclusion was made that there was not enough data to build a good correction factor.

Chapter 4

Implementation

In this section all the relevant aspects for the implementation of the model presented in the previous section should be given. Furthermore the equations in order to calculate different parameters are explained in more detail.

4.1 Subtended Solid Angle

The basic idea of this thesis was that the SSIM scores of a certain object increase with increasing distance to the camera which is one criteria of the LOD selection function. The increase in the score is due to the object getting smaller. As already explained because of the compensation of the magnifying effect of the lenses in VR HMDs the objects also get smaller with increasing eccentricity. Therefore is was also assumed that the SSIM scores are getting better with increasing eccentricity. Unfortunately this was not the case. After further investigation the issue was found in the effect of perspective projection which is applied for the renderings. As we want to map a flat 2D screen onto sphere, which is the human eye ball, perspective projection is necessary. In order to do this mapping correctly the object is stretched towards the edge of the screen. The stretching increases with increasing field of view as we want to map the screen to a larger area of the sphere. Instead of increasing scores with increasing eccentricity the scores got worse towards the edge of the screen. When comparing two results with same field of view but different settings for the lens coefficients the desired effect of better SSIM scores was observed.

To be independent of the perspective distortion the solid angle was the

metric used for all further calculations. This is exactly the measure needed as the solid angle in geometry describes the amount of field of view covered by an object. The unit of the solid angle is called steridian. The solid angle is calculated by

$$\sigma = \frac{A}{r^2} \tag{4.1}$$

where A is the area which is cut out of a sphere with radius r. Therefore one steridian equals an area cut out of the sphere with the size of $A = r^2$. As the whole surface of a sphere is given by $4\pi r^2$ the entire sphere has a solid angle of $4sr$.

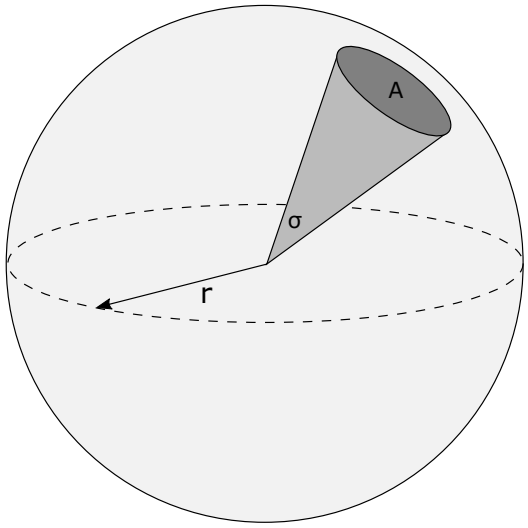


Figure 4.1: Solid Angle

The simplest approximation of the subtended solid angle of an object is based on a bounding sphere. Therefore the size of every object in the rendered scene is given by the subtended solid angle covered by the bounding sphere of the object.

4.2 Bounding Sphere

In order to evaluate the distance to the camera as well as the eccentricity, the distance to the center of the screen, a bounding sphere is used for each

object in the scene. This sphere should obviously completely cover the object. Each sphere Z is defined by its center $z = (z_x, z_y, z_z)^T$ and a radius r_z. Also important is that the values calculated to define the sphere are given in 3D camera coordinates.

Finding the smallest fitting bounding sphere is a popular mathematical problem more commonly known as smallest-circle problem, initially proposed by James Joseph Sylvester in 1857 [26]. There are different algorithms trying to solve this problem but for this work it was sufficient to come up with a more simple approaches. A tighter better fitting bounding sphere would not have an huge impact in this case.

Therefore the implementation calculates the center of the bounding sphere in two different ways explained below. The sphere with the smaller radius respectively with the better fit is taken for further calculations.

4.2.1 Center z

The following two approaches for the calculations of the center of the bounding sphere were taken into account.

4.2.1.1 Average of Vertices

The center of the sphere is calculated by the average of all vertices of the model. If the vertices are equally distributed over the whole mesh of the object this yields good results.

$$z = \frac{1}{n} * \sum_{i=0}^{n} v_i \tag{4.2}$$

With n being the number of vertices in the mesh.

4.2.1.2 Min-Max of Vertices

This approach is better suited if the vertices are not equally distributed. Consider a mesh consisting of two parts one with higher density of vertices and one with lower density, then the min max approach will result in a bounding sphere with smaller radius. The center is calculated by evaluating the minimum and the maximum of every vertex for each axis and then taking

average as the final result.

$$z = \frac{1}{2} * \forall v \in V \begin{pmatrix} \min(v_x) + \max(v_x) \\ \min(v_y) + \max(v_y) \\ \min(v_z) + \max(v_z) \end{pmatrix} \tag{4.3}$$

4.2.2 Radius r_z

The two mentioned approaches only differ in the calculation of the center of the bounding sphere but the calculation of the radius r_z is the same. The radius is given by the largest distance from the center to any vertex.

$$r_z = \max_{\forall v \in V} |z, v| \tag{4.4}$$

4.3 Computing Eccentricity

This section deals with the computation of the eccentricity as well as the amount of shrinking of the object introduced by the compensation of the lens distortion. As already mentioned the eccentricity serves as second criterion for the LOD selection function.

4.3.1 Subtended Solid Angle

First of all the subtended angle of the desired object is calculated. The bounding sphere results in a circle as silhouette. With the radius of the bounding sphere a cone can be formed seen from the eye point. The angle μ_z resulting from this can be calculated by the following equation:

$$\mu_z = arcsin \frac{r_z}{|z|} \tag{4.5}$$

The subtended solid angle of a cone is the area of a spherical cap on a unit sphere and is calculated as shown in equation 4.6.

$$\sigma_z = 2\pi(1 - \cos(\mu_z)) \tag{4.6}$$

4.3.2 Distortion Correction

The next step is to calculate the amount of reduction of σ_z resulting from the lens distortion compensation. The eccentricity of an object is calculated by the center of the bounding sphere, denoted α_z.

$$\alpha_z = \arctan(\frac{\sqrt{z_x^2 + z_y^2}}{z_z}) \tag{4.7}$$

4.3.2.1 Lookup of r^{-1}

Already explained in earlier chapters the lens distortion compensation function, a barrel distortion, is reducing the size of the objects towards the edge of the screen. The distance of a point to the screen's center is multiplied by a factor describing this reduction. This reduction factor is stored in lookup table which has to be available for the current HMD's distortion coefficients.

4.3.2.2 Conversion Functions

Before continuing with further calculations it is necessary to understand how to convert the angle α of a given point to a radial distance r and vice versa. The mentioned angle and radial distance of a point can be seen in figure 4.2.

Angle to Radius
As the lookup table takes the distance of a point to center of the screen, the radial distance, and the object's size is determined using angles a conversion has to be applied given in equation 4.8.

$$\alpha 2r(s) = \frac{\tan(\alpha(s))}{\tan(\omega)} \tag{4.8}$$

Radius to Angle
After applying the effect of the barrel distortion to the given radial distance a conversion back to angles is needed in order to proceed.

$$r2\alpha(s) = \arctan(r(s) * \tan(\omega)) \tag{4.9}$$

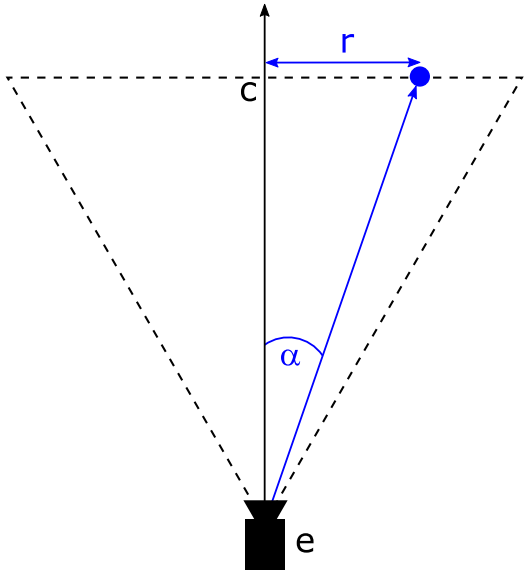


Figure 4.2: Conversion from Angle to Radius and vice versa

One last parameter is needed to calculate the correction factor for the lens distortion effect. This is the amount of distortion applied to the radial distance which was already explained in algorithm 1 in a previous section.

The distorted angles α^+ and α^- are calculated with the difference of those two values being exactly the subtended angle of $2\mu_z$ in an undistorted way. The values $(\alpha_z + \mu_z)$ and $(\alpha_z - \mu_z)$ are the boundary values of the object. They are first converted from angle to radial distance in order to scale them accordingly by the reverse lens distortion function r^{-1}. After the scaling the values are converted back to angles and now describe the new size of the object.

$$\alpha^+ = r2\alpha(r^{-1}(\alpha 2r(\alpha_z + \mu_z))) \tag{4.10}$$

$$\alpha^- = r2\alpha(r^{-1}(\alpha 2r(\alpha_z - \mu_z))) \tag{4.11}$$

The last step is to scale the previously calculated subtended solid angle of the object by the amount if shrinking introduced by the barrel distortion.

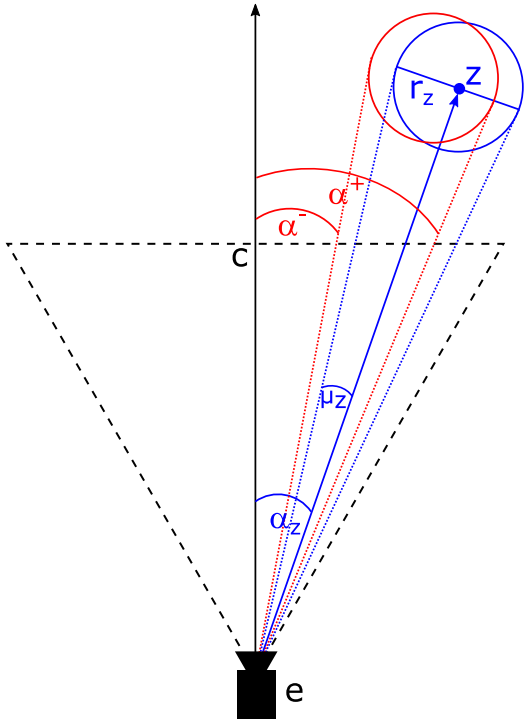


Figure 4.3: Distortion Correction

This is done by calculating the ratio between the new boundary values and the original ones.

$$\sigma_z' = \sigma_z \frac{\alpha^+ - \alpha^-}{2\mu_z} \tag{4.12}$$

The resulting subtended solid angle σ_z' is then used for the LOD selection function to evaluate the amount of detail which is sufficient for the current parameters.

When the object is positioned in the center of the screen the barrel distortion does not have any effect which results in α^+ and α^- being equal to $2\mu_z$ which means the initial subtended solid angle σ_z is multiplied by one resulting in no distortion correction.

4.4 LOD Selection Function

The previously calculated parameters lead to the following LOD selection function:

$$LOD(O, \sigma, e) = max_i(f(\sigma) <= M(O_i, \sigma')) \tag{4.13}$$

This equation evaluates the desired level of detail of an specific object O which covers the subtended solid angle σ at the current eccentricity e by checking all LODs O_i of the current object against a SSIM threshold given by $f(\sigma)$ where the score reached is given by $M(O_i, \sigma')$ which is considering the lens optics by shrinking the subtended solid angle σ' by the amount of distortion applied towards the edge of the screen.

4.5 LOD in Vector Streaming Framework

After setting up the window, loading the scene and initializing all the managers as described in section 2.7 the application maintains in the render loop until the window is closed and therefore the application is terminated. The render loop is a continuous execution of two functions which are first updating the scene and the animations according to the current timestamp and then rendering the scene.

In the second part of the render loop each manager in the vector streaming framework takes care of the parts it is responsible for. For example the entity manager culls the view frustum in order to check which entities are visible. Another thing the entity manager does at this point is checking the level of detail of each object in the scene. So this is exactly where all the calculations which were previously explained are done and the final LOD per object is evaluated. If the LOD of an object changes the the visibility of the old model is set to false while the new one is set to be visible.

For now there is no specific algorithm implemented to switch between the LODs which is one point for future extensions, for example by smoothly blending LOD representations. Hard LOD switching introduces the negative effect of visual pops but as this work focuses on when to switch LODs and not on how to switch them as well as the performance improvements this was sufficient.

Chapter 5

Evaluation

This chapter is divided into two different parts. The first part deals with the performed user study while the second part discusses the results of the performance evaluation and the savings resulting from the foveation aspect which was taken into account.

5.1 User Study

In order to evaluate the thresholds for the LOD selection, the already mentioned parameters k and d for the linear relation between the size of the object the SSIM score, a user study was performed. The first section explains the procedure of the user study while the second section gives an overview of the used hardware and the software setup. Finally the results of the study are discussed.

5.1.1 Procedure

The general idea was to present different objects and let the user decide how much the model can be simplified without seeing a significant difference compared to the base model. Furthermore this assumption should be valid for all distances to the camera, respectively for all sizes of the object.

For the duration of the whole study each object was rotating so the user is able to see it from different angles. The user was also able to stop the rotation in order to focus on different areas of the object. The certain object to evaluate was the only thing visible in the scene besides a small window

showing the instructions and current progress. Furthermore the object was presented on a neutral gray background with no complex lighting but only a normal ambient light. In order to compare the LODs to the base model the object always fades between the two. Fading was desired to avoid seeing the typical popping effects.

For each object the user had to go through two phases. The settings evaluated in both phases should always be as bad as possible and as good as needed.

5.1.1.1 Phase 1

Phase one was used to evaluate the d value for the linear relation. This was done by presenting the object at the size of one sixteenth of the entire screen size. This is exactly the size where the use of lower levels of detail should start. The user was then able to go through the ten LODs which are available for each object. When a level of detail with almost no difference to the base model was found the user accepted the decision and was led to phase two.

5.1.1.2 Phase 2

The aim of the second phase was to find the slope setting for the linear relation. The slope settings lets the user decide how much detail can be lost with increasing depth while still remaining the main assumption of almost not seeing a difference to the full resolution model. In this phase the user has to try different slope settings and the check if the setting is sufficient for the whole range of distances. When a good setting was found the next object was shown and phase one was executed again.

5.1.2 Setup

This section will briefly describe the used HMDs as well as the objects presented during the user study.

5.1.2.1 Head Mounted Displays

The evaluation of all the objects for the user study was performed on two different head mounted displays. This was done in order to see how the results change when different hardware is used. The differences are discussed

in a later section while the purpose of this section is to describe the used hardware and list the specifications.

HTC Vive Pro

The first headset used is the HTC Vive Pro [27] which was introduced in January 2018 as an upgraded model of the previous Vive. The more recent version has higher resolution displays and is a bit lighter in terms of overall weight. Further specifications are listed below.

HTC Vive Pro Specifications	
Display Technology	OLED
Resolution	1440x1600 pixels per eye
Refresh Rate	90 Hz
Field of View	110°
Lens Type	Fresnel
Price	700€ *

* price for HMD only (as of July 2020)

Table 5.1: HTC Vive Pro

Oculus Rift S

The Oculus Rift S [28] was the second headset used in the user study. It is officially available on the market since May 2019. The Oculus Rift S is also a newer version of the previous Oculus Rift. Again the display resolution is a tiny bit higher in the newer version but what is more interesting is that the newer version has an LCD panel while the older one was equipped with an OLED display. Table 5.2 lists further specifications.

Oculus Rift S Specifications	
Display Technology	LCD
Resolution	1280x1440 pixels per eye
Refresh Rate	70 Hz
Field of View	NA
Lens Type	Fresnel
Price	450€ *

* price for HMD + controller (as of July 2020)

Table 5.2: Oculus Rift S

5.1.2.2 Tested Objects

The user had to go through the test phases for a total of ten different objects. For the objects itself the aim was to have a good variation of different kind of aspects like the following:

- geometry: round object and objects with rather flat surfaces as well as objects with complex high curvature meshes
- texture: differing in texture contrast and texture frequency
- material: a variation of materials with different effects like specularity or metallicity
- polygon count: meshes with varying number of polygons of the original base model

In figure 5.1 eight of the tested objects can be seen. Two objects were shown with the same geometry but without any texture or special material. The object were presented with a white to gray base color and normal phong shading. This was done in order to check the influences of the appearance on the geometry itself.

5.1.3 User Profiles

A total number of twenty four users was tested. None of them had any idea what the user study was about nor got any information or instructions in advance. All users were at the time of the user study working for the Institute

Figure 5.1: Objects used in the user study

of Computer Graphics of the TU Graz. Some statistics about the user can be seen in table 5.3.

User Statistics	
Total Number of Participants	25
Age	23 to 43
Males	21 of 25
Females	4 of 25
Corrected Vision	14 of 25
Computer Experience	4.84* on average
VR Experience	3.08* on average

*rating ranging from 1 (lowest) to 5 (highest)

Table 5.3: Oculus Rift S

5.1.4 Results

The aim of this section is the discussion of the results gained from the user study. All outputs from the user study were processed using Statistical Pack-

age for the Social Sciences (SPSS) [29] by IBM. First thing to investigate was if there is a remarkable difference between the two different headsets.

5.1.4.1 Normality Test

Before continuing with further evaluation steps the acquired data has to be tested for normality. A lot of statistical tests for further investigations have the prerequisite for the data to be normally distributed. Two different kinds of normality tests were executed for each parameter from the user study which were the initial starting threshold at one sixteenth of the screen size d and the corresponding slope setting k which evaluates the SSIM threshold for larger distances.

To test the data for normality two different approaches were used. The first one is called Kolmogorov-Smirnov [30] and the second one is called Shapiro-Wilk [31]. The results of both are shown in table 5.4.

	HMD	Kolmogorov-Smirnov			Shapiro-Wilk		
		Statistic	df	Sig.	Statistic	df	Sig.
d	rift	0.135	240	0.00	0.885	240	0.00
	vive	0.142	240	0.00	0.842	240	0.00
k	rift	0.201	240	0.00	0.913	240	0.00
	vive	0.167	240	0.00	0.936	240	0.00

Table 5.4: Normality Tests for different HMDs

The output above was created by SPSS and the important column is denoted $Sig.$ which corresponds to p the probability of the hypothesis. In this case the hypothesis was that the data is normally distributed. In statistical application a threshold value to reject the hypothesis is $p < 0.05$. This is the case in the performed normality tests and therefore the hypothesis is rejected which means the data for the parameters k and d is not normally distributed.

Further investigation concluded that both test probably failed because the data is skewed, especially for the parameter d. The distribution plots show that the data is not perfectly normally distributed but almost has the correct shape.

When looking at the descriptives of the data in table 5.5 the skewness can be seen. The difference between the column statistic and std. error is

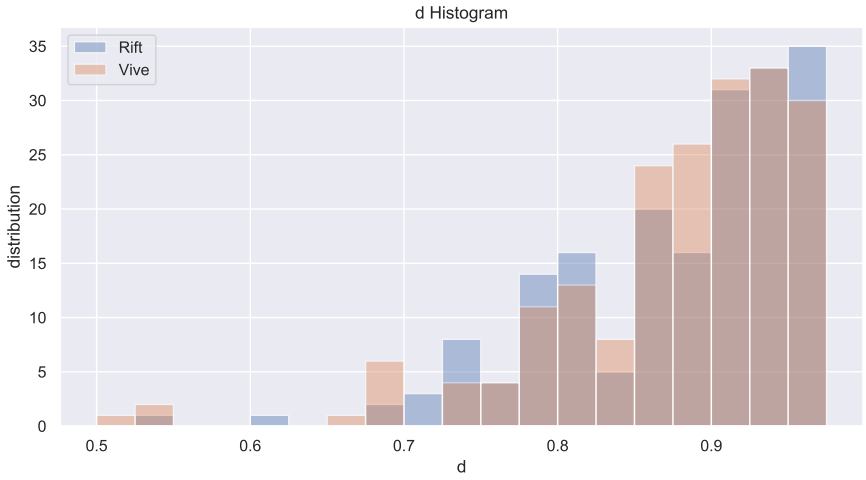


Figure 5.2: Distribution of Parameter d for both HMDs

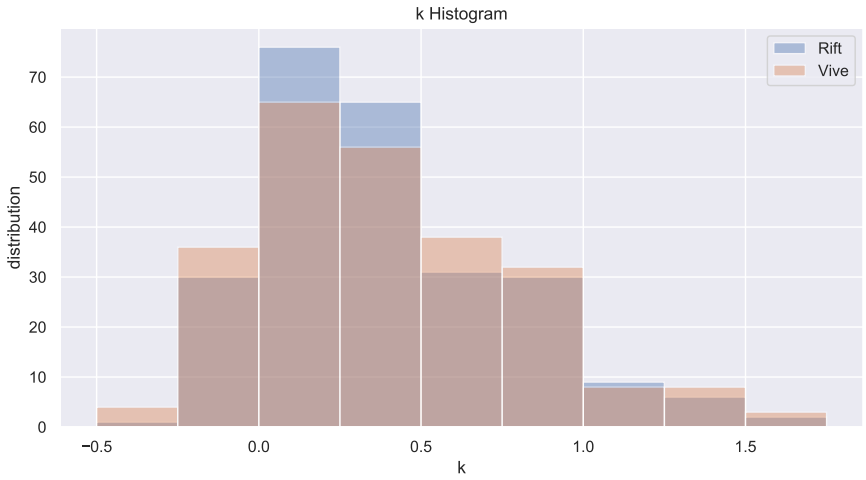


Figure 5.3: Distribution of Parameter k for both HMDs

quite high which is a good indicator why the normality tests failed. For the parameter d this obviously makes sense as the values the user picked have a high chance to be close to one but definitely cannot be higher than one.

	HMD		Statistic	Std. Error
k	rift	Mean	0.2677	0.02526
		Std. Deviation	0.39139	
		Skewness	0.852	0.156
		Kurtosis	0.597	0.313
	vive	Mean	0.2792	0.02719
		Std. Deviation	0.42130	
		Skewness	0.619	0.157
		Kurtosis	-0.006	0.313
d	rift	Mean	0.9050	0.0053
		Std. Deviation	0.08365	
		Skewness	-1.201	0.157
		Kurtosis	1.474	0.313
	vive	Mean	0.8971	0.0061
		Std. Deviation	0.0958	
		Skewness	-1.783	0.157
		Kurtosis	4.384	0.313

Table 5.5: Descriptives of the user study data

5.1.4.2 Difference between HMDs

To see if there exists a difference between the headsets based on the two evaluated parameters the first idea was to do an Analysis of Variances (ANOVA) [32]. The result showed that there is no significant difference between the headsets but as normality is a prerequisite in order to perform ANOVA the result was not valid. The same prerequisite of normality holds for another popular approach which is the T-Test [33]. A non parametric test has to be performed which does not have the prerequisite of normality.

Wilcoxon
Wilcoxon [34] also showed that there is no difference between the headsets which can be seen by the values $Sig.$ in table 5.6. As it was with the normality

tests a value of Sig. and respectively p smaller than 0.05 indicates that the hypothesis of recognizing a difference can be rejected and an alternative hypothesis is accepted which says there is no difference between the HMDs.

Test Statistics[a]		
	d Vive - d Rift	k Vive - k Rift
Z	-0.567[b]	-0.341[c]
Asymp. Sig. (2-tailed)	0.571	0.733

a Wilcoxon Signed Ranks Test
b Based on positive ranks
c Based on negative ranks

Table 5.6: Normality Tests for different HMDs

Personal HMD Rating
Throughout the whole user study, general opinions about the HMDs were collected. The majority of users considered the Oculus Rift S as the better headset in terms of comfort and ease of use. For the Vive the opinion was that it is way harder to find the sweet spot, if found though users reported a very good quality. Furthermore the quality at larger eccentricities was not that good for the Vive.

5.1.4.3 Correlation between k and d

The next thing to investigate was the correlation of k and d. The assumption was that the parameter k depends on the previously chosen parameter d.

In figure 5.4 a scatter plot for the two parameters can be seen. This plot also helps to understand correlations between two variables. When fitting a linear line through the data a positive or negative correlation can be easily seen. When the slope of the linear fit is rather low, resulting in a horizontal line, there is almost no correlation. This seems to be the case for the given data.

To confirm the assumption made from the plot some statistical tests were performed again.

Pearson's r
A popular method to check for correlation between two variables is Pearson's

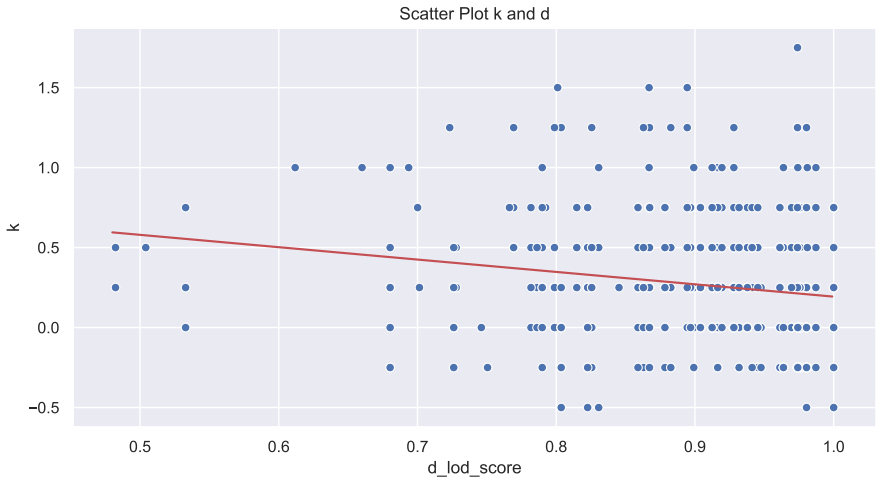


Figure 5.4: Scatter Plot for Parameters k and d

r [35] but again this test has the prerequisite of normality and was therefore no option.

Kendall's tau b and Spearman's rho

Two common non parametric methods to test for correlation are Kendall's tau [36] and Spearman's rho [37] which were both performed. In general correlation coefficients are in the range of minus one to plus one. Negative values mean that there is a negative correlation and the opposite for positive values. In general values close to zero indicate that there is not correlation and absolute values close to one show a high correlation. A classification for the strength of the correlation is shown in table 5.7.

Correlation Coefficients	
0.0 - 0.3	negligible correlation
0.3 - 0.5	low correlation
0.5 - 0.7	moderate correlation
> 0.7	high correlation

Table 5.7: Categorization of Correlation Coefficients

The results of the non parametric tests for correlation are listed in table

5.8 representing a typical correlation matrix. According to the information given above the correlation between d and k is rather low and therefore negligible.

		d	k
Kendall's tau	d	1.0	-0.164
	k	-0.164	1.0
Spearmans's rho	d	1.0	-0.220
	k	-0.220	1.0

Table 5.8: Correlation Coefficients for Parameters k and d

For both tests the correlation is significant at the 0.01 level which was the case as the values were almost zero. This results in a significant correlation but the correlation itself is so low that it is negligible.

5.1.4.4 Appearance Parameters Influences

The SSIM score of every tested object does not only depend on the reduction ratio to produce levels with different amount of detail but also on other aspects as the geometry of the object or different material properties. This is also the reason why some models in the user study were duplicated with the only difference in having another material.

One example from the dataset for the user study is the elephant which was shown textured and untextured. The SSIM score for the textured elephant were lower because the differences between each pixel were higher due to a more complex texture. The human eye on the other hand could probably see more difference in the simpler shaded model as the texture is too complex to see differences.

This led to the general assumption that a correction factor should be implemented. This factor should improve the scores of complex material parameters because those differences can hardly be seen by the human eye. As the SSIM score in general should deliver results closer to the human perception this aspect could possibly seen as a little downside. In order to create a correction term based on material parameters the results of the user study were scanned for correlations between the parameters k and d and some aspects of material parameters.

5.1.4.5 Conclusion User Study

After discussing all the results and investigating correlations between different parameters the conclusion was made to take the average of all user study results as final result for further evaluation steps.

The average values are $\boldsymbol{d = 0.90}$ and $\boldsymbol{k = 0.27}$.

5.2 Performance Evaluation

In order to evaluate the performance savings as a result by the level of detail implementation three different LOD modes were compared. The first one being *disabled* meaning the only model is the base model and no other levels are used. The second one does only take the distance to the camera into account, denoted as *depth only* and the third mode which takes both the distance to the camera as well as the distance to the center of the screen (eccentricity) as criterion for the level of detail selection. The last mode is called *depth and eccentricity*.

Additional Information on FOV

As mentioned in earlier sections the field of view of the used HMD is one relevant specification which influences the result of the LOD selection. The real FOV is not always stated by the manufacturing company of the headset. Lately FOV became a key aspect for VR headset and therefore for the values stated it is unclear which FOV it really is. For marketing reasons, the higher the field of view the better, often the diagonal FOV is given. What is of interest for this work is the horizontal and the vertical field of view.

To be independent from any company revealed values the field of view was measured. This was done via a small tool which places a target on the screen which is then moved to the edge of the screen by moving the head. When the object is exactly at the edge the values of the current field of view can be read.

As the field of view differs from person to person due to different distances between eyes and lens as well as the eyes' position within the HMD it was tried to get the largest field of view possible. Even though some people may experience a smaller field of view objects at the edge of the screen are still rendered even though they are not seen by the user anymore. Therefore what is of interest is the field of view of the screen itself not of a specific person.

The values acquired for the two headsets of the user study are listed in table 5.9.

HMD	measured Field of View
rift	86-88°horizontal 90°vertical
vive	94-96°horizontal 94-96°vertical

Table 5.9: Measured field of view values for both HMDs

Acquiring Distortion Coefficients
The next parameters relevant for the LOD selection in order to correctly run the third LOD mode also considering the eccentricity are the lens distortion coefficients. Those values are definitely not of interest for anyone desiring to buy a VR headset. That is the reason why it is very hard to obtain those values for a large amount of hardware.

The distortion coefficients for the Oculus Rift S were found as parameters in the Oculus Software Development Kit [38].

For the HTC Vive there is the possibility to up and download a configuration file which is later used for the rendering by SteamVR. This configuration file does also contain information of the distortion coefficients used. The file contains coefficients for all three color channels in order to automatically consider chromatic aberration. The average of all color channels was taken as input for the LOD selction framework. The results for both headsets can be seen in table 5.10.

HMD	k1	k2
rift	0.22	0.24
vive	0.2063	0.0576

Table 5.10: Lens Distortion Coefficients

5.2.1 Setup for Performance Evaluation

First the used hardware components and specifications are described, and then the setup of the different scenes used in the rendering framework is presented.

5.2.1.1 Hardware

All performance tests were executed on a machine equipped with the hardware components listed and further described in table 5.11. There were no applications other than the performance tests running in order to produce comparable results.

Component	Name	Description
GPU	NVIDIA GeForce RTX 2060 SUPER	8GB GDDR6
CPU	Intel Core I5 9600K	3.70 GHz
RAM	Corsair Vengeance LPX	16GB DDR4

Table 5.11: Hardware component description

5.2.1.2 Scene Setup

For the scene setup, a variation of two parameters was tested. The first parameter is the density of the scene, which means the number of objects in the scene varied. The second parameter is the size of the scene the objects are placed in, which will be described in more detail later.

The scenes were created in Unity [39], which is a very common and well known tool, and then exported as gltf files, which can then be converted and imported into the Vector Streaming framework. In the Vector Streaming framework, a simple forward rendering approach was used for the performance evaluation.

To create the scenes in Unity, a script was used in order to place the desired amount of objects randomly in a sphere with an desired radius. The number of objects as well as the radius of the sphere are the parameters mentioned before. The radius of the sphere is given in world coordinates. The advantage of using a sphere is that the maximum distance as seen from the center of the sphere is always the same which, would not be the case for a cube.

The performance evaluation was executed for the specs of both headsets, which were already used in the user study. As discussed before, the field of view as well as the distortion parameters were changed according to the headset.

To generate the values presented later, the camera was placed in the middle of the sphere. This camera was then smoothly rotated around one

axis. For each of the acquired values, the average of 2500 frames was taken.

5.2.2 Results

This section is organized in three different parts. First, some plots of the performance evaluation are shown and explained in more detail, and then the overall results for both triangle count savings as well as render time savings are presented.

5.2.2.1 Run-Time Plots and LOD Distribution

The first plot in figure 5.5 shows the distribution of levels for two different LOD modes and one specific scene configuration. It can be seen that the distribution of the LOD, which also takes the eccentricity into account, has more values for higher levels, which results exactly in the amount of triangles which can be saved activating this additional criterion for the LOD selection. The chosen scene was the one with a rather small radius for the sphere, which results in selection of lower levels of detail and therefore a distribution located on the left side of the plot.

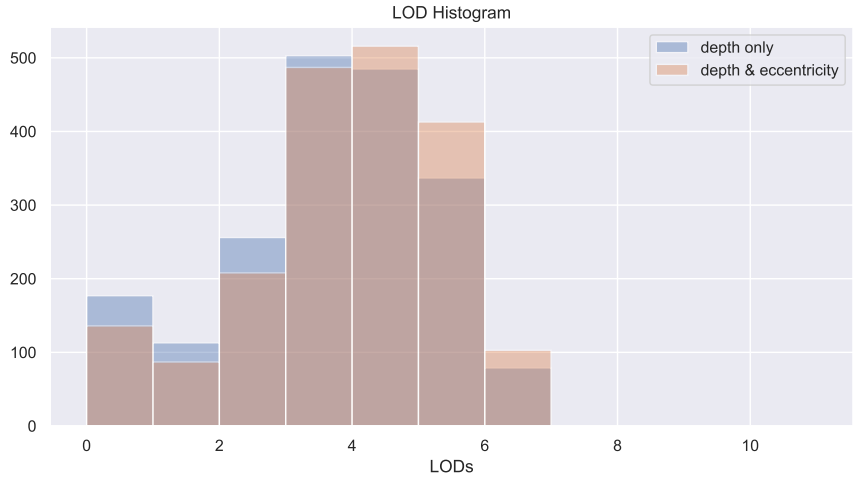


Figure 5.5: Histogram of Levels of Detail

Figure 5.6 shows one part of the frames which are captured by the camera rotation inside the sphere. The y axis gives the values for the average

subtended solid angle of all objects currently in the view frustum of the camera. Again, the two LOD modes are presented, and it can be seen that the values for the LOD mode with eccentricity are lower and therefore result in lower levels of detail. How much lower the average subtended solid angle is compared to the one without eccentricity depends on the barrel distortion coefficients and the amount of object placed closer to the edge of the screen.

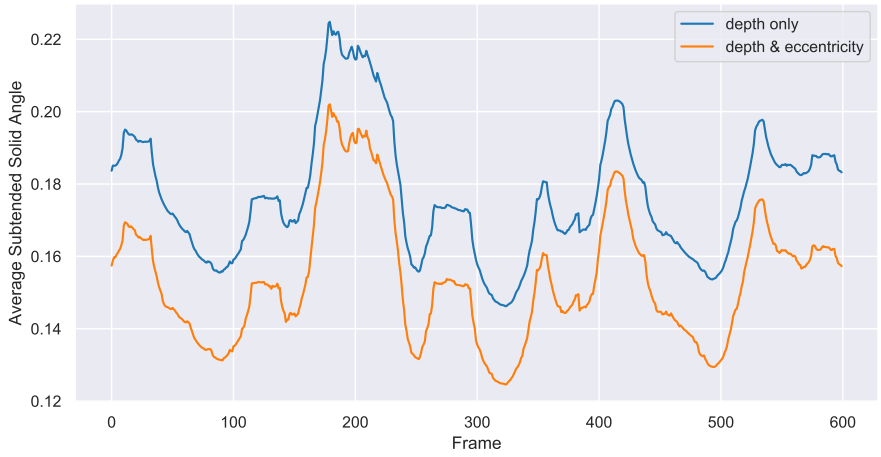


Figure 5.6: Plot of average subtended solid angles in a scene

The last figure in this sections shows the impact of the previously described techniques on render times. Larger amounts of higher levels of detail result in less triangles to be processed and therefore result is lower render times.

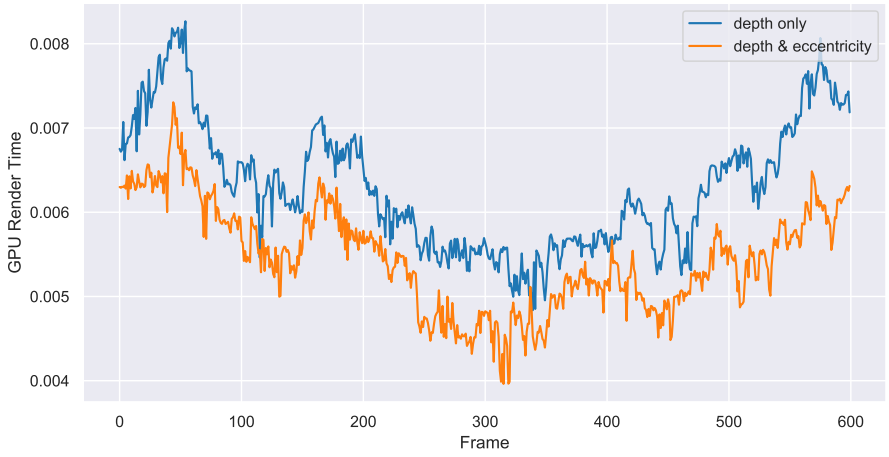


Figure 5.7: Plot of average render times in a scene

5.2.2.2 Triangle Savings

Table 5.12 and 5.13 show the results in respect to saved triangles for different scene configurations, as described in the section before. The values in the table show the differences between the LOD modes. The first value tells how many triangles are saved by enabling LOD with distance as the only criterion. The second one compares the distance based LOD with another mode which also takes the eccentricity into account.

	Size							
	10		15		20		30	
Density	Depth	Ecc	Depth	Ecc	Depth	Ecc	Depth	Ecc
100	92.2%	29.5%	96.8%	23.6%	97.5%	14.8%	99.1%	8.7%
500	91.7%	29.5%	97.4%	24.7%	98.6%	13.2%	98.8%	9.1%
1000	91.9%	29.0%	97.4%	24.0%	98.4%	14.4%	98.6%	9.1%

Table 5.12: Oculus Rift - Percent of saved polygons for different scene configurations and LOD modes

	Size							
	10		15		20		30	
Density	Depth	Ecc	Depth	Ecc	Depth	Ecc	Depth	Ecc
100	92.7%	22.7%	97.4%	19.1%	97.4%	10.4%	99.0%	8.6%
500	92.5%	21.6%	97.7%	18.7%	98.7%	10.1%	99.0%	9.1%
1000	92.7%	21.4%	97.6%	18.1%	98.5%	11.8%	99.0%	8.7%

Table 5.13: HTC Vive - Percent of saved polygons for different scene configurations and LOD modes

In general, it can be seen that with increasing distance, the savings are larger, as the selected levels are also higher. Additionally, the values for the mode with eccentricity decrease with increasing depth, because the difference in number of triangles is lower, the higher the LOD is. For example, level two to level three saves 1000 triangles, whereas level nine to level ten only saves 100.

A difference between the two HMDs can also be seen. The savings for the rift are higher as the ones for the vive. This makes sense, as the distortion coefficients for the rift are larger, which results in more distortion compensation and therefore in smaller objects at the edge of the screen, which lead to higher levels of detail.

What can be observed as well is that the triangle savings do not depend on the density of the scene, which means it does not matter how many objects are placed, the percent of saved triangles will be roughly the same.

Furthermore one has to keep in mind that the values presented can vary a bit as the objects were randomly placed in the scene. Recreating the scenes will obviously result in different values, but in a very similar range as the ones presented.

In table 5.14 and 5.15, the actual values of overall triangles currently in the view frustum of the camera and averaged over all rendered frames are listed.

	Size											
	10			15			20			30		
Density	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc
100	35.8	2.78	1.96	19.5	0.62	0.47	14.3	0.36	0.31	10.3	0.09	0.08
500	175	14.5	10.2	94.9	2.52	1.90	64.7	0.91	0.79	50.5	0.62	0.57
1000	334	27.0	19.2	179	4.71	3.58	132	2.10	1.79	96.2	1.01	0.91

Table 5.14: Oculus Rift - Actual values in millions of polygons for different scene configurations and LOD modes

	Size											
	10			15			20			30		
Density	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc
100	35.6	2.61	2.02	20.8	0.55	0.45	13.8	0.36	0.32	9.29	0.10	0.09
500	178	13.4	10.49	101	2.37	1.93	69.2	0.92	0.83	56.2	0.56	0.51
1000	339	24.9	19.6	189	4.60	3.77	142	2.12	1.87	104	1.02	0.94

Table 5.15: HTC Vive - Actual values in millions of polygons for different scene configurations and LOD modes

5.2.2.3 Render Time Savings

The second part of the performance evaluation deals with the savings in respect to render times. First, the method how the values were acquired is discussed, and then the results are presented and explained.

Rendering Overhead

The first thing which was done was to measure the overhead of the rendering engine itself. This overhead was then subtracted from every value measured for the rendering time. The overhead was calculated by running a scene with zero objects in it.

For the used machine and configuration of the framework, a value of 0.332 milliseconds was established.

Timing Values

In general it is important to mention that GPU timings have to be measured directly on the GPU and not on any other processor, like the CPU. The Vulkan API offers this possibility by configuring a query to return timestamps. There are different query types such as Occlusion, Pipeline Statistics and Timestamps. The query type Occlusion for example can give information about how many fragments pass a certain test, whereas Pipeline Statistics type is able to return the number of vertex shader invocations. Timestamps query, as the name says, returns timestamps and is the important one for this work.

The functionality is activated right before a Vulkan graphics command and queried afterwards. The timestamps can be configured to be taken at a specified pipeline stage, such as the vertex shader. Due to the parallelism of the GPU, this does not mean that it only measures this pipeline stage, as previously processed graphics commands can be executed in parallel in another pipeline stage. Because of this, the render times include all pipeline stages for all the models in the scene. In order to produce meaningful results, it was important to reduce the impact of other programmable stages such as the fragment shader. Therefore the fragment shader simply returns one specific color without further processing.

Results

Table 5.16 and 5.17 show the results which, were acquired for render time savings.

	Size							
	10		15		20		30	
Density	Depth	Ecc	Depth	Ecc	Depth	Ecc	Depth	Ecc
100	91.0%	20.3%	93.8%	9.5%	93.5%	4.6%	95.7%	<1.5%
500	90.8%	20.9%	95.8%	9.7%	97.0%	4.6%	95.3%	<1.5%
1000	91.2%	19.9%	96.1%	10.6%	96.5%	4.9%	96.3%	<1.5%

Table 5.16: Oculus Rift - Percent of saved render time for different scene configurations and LOD modes

	Size							
	10		15		20		30	
Density	Depth	Ecc	Depth	Ecc	Depth	Ecc	Depth	Ecc
100	91.9%	14.4%	94.5%	7.4%	96.7%	4.2%	97.5%	<1.5%
500	91.5%	14.8%	96.1%	7.2%	97.3%	4.0%	96.8%	<1.5%
1000	91.8%	14.7%	96.5%	7.3%	96.7%	3.8%	96.3%	<1.5%

Table 5.17: HTC Vive - Percent of saved render time for different scene configurations and LOD modes

As already described, the values include all pipeline stages. Therefore, it has to be mentioned that, for lower number of triangles, the other pipeline stages do have more effect on the final output value. Other than that, the decreasing amount of render time savings is due to smaller differences in levels, as already explained for the triangle savings.

Furthermore, there is again no considerable difference in term of scene density, which was also the case for the triangle savings described before.

Table 5.18 as well as table 5.19 list the actual values of the evaluation in respect to render time.

	Size											
	10			15			20			30		
Density	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc
100	7.43	0.67	0.53	4.15	0.25	0.23	3.14	0.21	0.20	2.30	0.010	0.099
500	37.0	3.40	2.68	20.5	0.87	0.79	14.3	0.42	0.40	11.5	0.542	0.536
1000	71.1	6.24	5.00	38.6	1.49	1.33	29.3	1.03	0.98	21.9	0.811	0.799

Table 5.18: Oculus Rift - Actual values in milliseconds for different scene configurations and LOD modes

	Size											
	10			15			20			30		
Density	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc	Off	Depth	Ecc
100	7.5	0.61	0.52	4.38	0.24	0.22	3.01	0.10	0.09	2.07	0.052	0.051
500	38.3	3.24	2.76	22.0	0.85	0.79	15.5	0.43	0.41	12.7	0.403	0.399
1000	73.3	6.00	5.12	41.76	1.46	1.35	31.6	1.03	0.99	23.6	0.865	0.856

Table 5.19: HTC Vive - Actual values in milliseconds for different scene configurations and LOD modes

Chapter 6

Conclusions

In conclusion, the SSIM based LOD framework in this work did a quite good job. One of the advantages of the SSIM based approach is that, no matter how many triangles an object has, the simplification of the meshes and the amount of applied simplification can be very well measured with the SSIM metric. This is one aspect of the metric which is really close to the real human visual perception. There is definitely some future work left to determine the behaviour of the metric for different materials, material properties and shading as well as lightning techniques. This aspect was shortly mentioned, but for sure deserves more attention. With more investigation in this direction, the SSIM score may be even more useful to categorize objects.

In terms of performance savings, the approach chosen in this work has some potential to ensure that there is no visual difference in the final renderings compared to renderings without foveation. The savings for the foveation via barrel distortion functions were less than originally expected. The comparison between the foveation function of a human eye and the ones for the two HMDs used in this work can be seen in figure 6.1. There clearly is a huge difference between the functions in the plot and the potential of the foveation function for a human eye is definitely way higher. This is also the reason why this will most likely be the direction of future researches.

This work can also be extended in terms of hardware. As a start the two HMDs, which are the Oculus Rift and the HTC Vive, were evaluated while there are way more HMDs on the market which could be added for the technique presented. With the future development of new hardware for Virtual Reality the HMDs may also have different properties, which could also influence the performance of this technique.

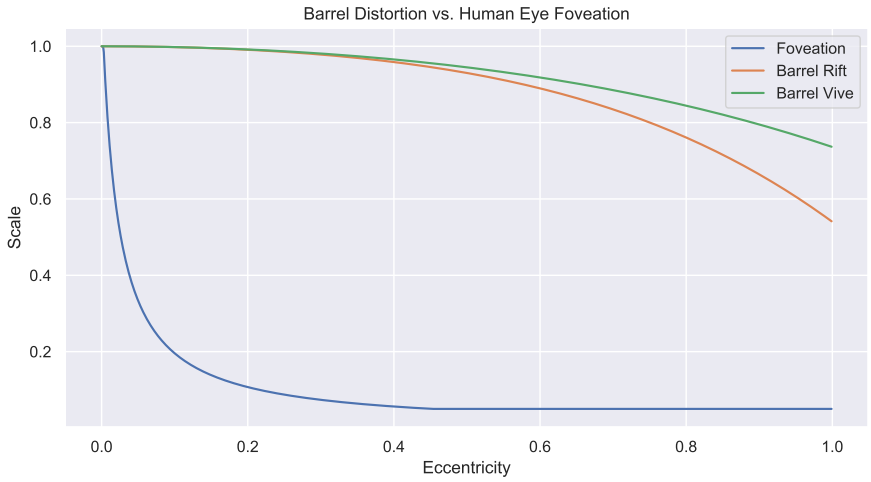


Figure 6.1: Comparison of barrel distortion functions and foveation - Adapted from Guenter et al. Foveated 3D graphics [40]

In the end, render times are a very important factor not only for Virtual Reality, but all real-time renderings in general. Level of Detail is a simple, but very efficient way to boost the performance. This boost can be further enhanced by including a foveation function. There will be room for future research in this direction.

Bibliography

[1] Paul Milgram et al. "Augmented reality: A class of displays on the reality-virtuality continuum". In: *Telemanipulator and Telepresence Technologies* 2351 (1st Jan. 1994). DOI: 10.1117/12.197321.

[2] Stanford University Brian A. Wandell. *Foundations of Vision » Chapter 3: The Photoreceptor Mosaic*. URL: https://foundationsofvision.stanford.edu/chapter-3-the-photoreceptor-mosaic/ (visited on 15/12/2020).

[3] *Fresnel Lens and Parabolic Reflectors*. parabolix-light. URL: https://www.parabolixlight.com/fresnel-lens-and-parabolic-reflectors (visited on 15/12/2020).

[4] omnia360. *Motion Sickness: 6 Tipps, damit VR nicht zum Kotzen wird.* URL: https://omnia360.de/blog/motion-sickness-in-virtual-reality/ (visited on 09/01/2021).

[5] David M. Hoffman et al. "Vergence-accommodation conflicts hinder visual performance and cause visual fatigue". In: *Journal of vision* (2008). DOI: 10.1167/8.3.33.

[6] The Khronos Group Inc. *Vulkan API*. URL: https://www.khronos.org/vulkan/ (visited on 20/10/2019).

[7] *Introduction - Vulkan Tutorial*. URL: https://vulkan-tutorial.com/Drawing_a_triangle/Graphics_pipeline_basics/Introduction (visited on 20/12/2020).

[8] *Calculating Stereo Pairs*. URL: http://paulbourke.net/stereographics/stereorender/ (visited on 09/01/2021).

[9] Duane C. Brown. "Decentering distortion of lenses". In: *Photogrammetric Engineering* (1966).

[10] A. E. Conrady. "Decentred Lens-Systems". In: *Monthly Notices of the Royal Astronomical Society* (1919).

[11] *Improved Pre-Warping for Wide Angle, Head Mounted Displays, Daniel Pohl's Virtual Reality Blog*. URL: http://blog.qwrt.de/improved-pre-warping-for-wide-angle-hmds/ (visited on 20/12/2020).

[12] David Luebke et al. *Level of Detail for 3D Graphics*. Elsevier, 31st July 2002. ISBN: 978-0-08-051011-8.

[13] James H. Clark. "Hierarchical geometric models for visible surface algorithms". In: *Communications of the ACM* (1st Oct. 1976). DOI: 10.1145/360349.360354.

[14] *SIGGRAPH '93: Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*. Anaheim, CA: Association for Computing Machinery, 1993. ISBN: 0897916018.

[15] Jarek Rossignac and Paul Borrel. "Multi-resolution 3D approximation for rendering complex scenes". In: (1st Jan. 1993). DOI: 10.1007/978-3-642-78114-8_29.

[16] Thomas A. Funkhouser and Carlo H. Séquin. "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments". In: *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*. SIGGRAPH '93. Anaheim, CA: Association for Computing Machinery, 1st Sept. 1993. ISBN: 978-0-89791-601-1. DOI: 10.1145/166117.166149.

[17] Zhou Wang et al. "Image quality assessment: from error visibility to structural similarity". In: *IEEE Transactions on Image Processing* (2004).

[18] *The SSIM Index for Image Quality Assessment*. URL: https://www.cns.nyu.edu/~lcv/ssim/ (visited on 20/12/2020).

[19] Joerg Mueller et al. "Shading atlas streaming". In: ACM Transactions on Graphics. 4th Dec. 2018. DOI: 10.1145/3272127.3275087.

[20] *OpenGL Mathematics*. URL: https://glm.g-truc.net/0.9.9/index.html (visited on 17/08/2020).

[21] *GLFW - An OpenGL library*. URL: https://www.glfw.org/ (visited on 17/08/2020).

[22] *ValveSoftware/openvr*. 17th Aug. 2020. URL: https://github.com/ValveSoftware/openvr.

[23] *SteamVR – Valve Corporation*. URL: https://www.steamvr.com/de/ (visited on 17/08/2020).

[24] *Decimate Modifier — Blender Manual*. URL: https://docs.blender.org/manual/en/latest/modeling/modifiers/generate/decimate.html (visited on 20/07/2020).

[25] Masseyeff RF et al. *Methods of Immunological Analysis Volume 1: Fundamentals*. New York, NY: VCH Publishers, Inc., 1993.

[26] James Joseph Sylvester. "A question in the geometry of situation". In: *Quarterly Journal of Pure and Applied Mathematics* (1857).

[27] *VIVE™ — Discover Virtual Reality Beyond Imagination*. URL: https://www.vive.com/sea/ (visited on 15/07/2020).

[28] *Oculus Rift S: VR-Headset für VR-fähige PCs — Oculus*. URL: https://www.oculus.com/rift-s/?locale=de_DE (visited on 15/07/2020).

[29] *IBM SPSS – IBM Analytics – Österreich*. 15th Feb. 2017. URL: https://www.ibm.com/analytics/at/de/technology/spss/ (visited on 15/08/2020).

[30] Frank J. Massey. "The Kolmogorov-Smirnov Test for Goodness of Fit". In: *Journal of the American Statistical Association* (1951). DOI: 10.2307/2280095.

[31] S. S. Shapiro and M. B. Wilk. "An analysis of variance test for normality (complete samples)". In: *Biometrika* (1st Dec. 1965). ISSN: 0006-3444. DOI: 10.1093/biomet/52.3-4.591.

[32] R. A. Fisher. "Statistical Methods for Research Workers". In: *Breakthroughs in Statistics: Methodology and Distribution*. Ed. by Samuel Kotz and Norman L. Johnson. Springer Series in Statistics. New York, NY: Springer, 1992. ISBN: 978-1-4612-4380-9. DOI: 10.1007/978-1-4612-4380-9_6. URL: https://doi.org/10.1007/978-1-4612-4380-9_6.

[33] Student. "The Probable Error of a Mean". In: *Biometrika* (1908). ISSN: 0006-3444. DOI: 10.2307/2331554.

[34] Frank Wilcoxon. "Individual Comparisons by Ranking Methods". In: *Biometrics Bulletin* (1945). DOI: 10.2307/3001968.

[35] Karl Pearson and Francis Galton. "VII. Note on regression and inheritance in the case of two parents". In: *Proceedings of the Royal Society of London* (1st Jan. 1895). DOI: 10.1098/rspl.1895.0041.

[36] M. G. Kendall. "A NEW MEASURE OF RANK CORRELATION". In: *Biometrika* (1st June 1938). DOI: 10.1093/biomet/30.1-2.81.

[37] C. Spearman. "The Proof and Measurement of Association between Two Things". In: *The American Journal of Psychology* (1987). ISSN: 0002-9556. DOI: 10.2307/1422689.

[38] *Oculus SDK for Windows — Developer Center — Oculus*. URL: https://developer.oculus.com/downloads/package/oculus-sdk-for-windows/ (visited on 24/08/2020).

[39] Unity Technologies. *Unity Real-Time Development Platform — 3D, 2D VR & AR Engine*. URL: https://unity.com/ (visited on 06/12/2020).

[40] Brian Guenter et al. "Foveated 3D graphics". In: *ACM Transactions on Graphics* (1st Nov. 2012). ISSN: 0730-0301. DOI: 10.1145/2366145.2366183.

Appendix A

Appendix

The Appendix contains all documents used in the user study such as the guidance sheet presented to all participants as well as forms which were filled and signed by all users. Additionally attached are the entire results of the performed user study. These results were used to evaluate the parameters k and d which were then used in all the performance measurements.

User Study Guidance

Please double click user_study.bat now. Afterwards read through the document.

Introduction:
My master thesis is about Levels of Detail (LOD) of objects in VR. LODs represent the same object with simplified meshes. The criteria for switching to lower detail is the distance to the camera.

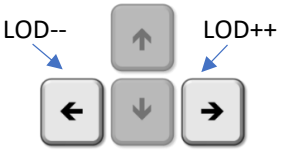
The Structural Similarity Index (SSIM) is used to measure the error between original model and LOD for different distances to the camera. The score is then used for the LOD selection. The aim of the user study is to evaluate the thresholds of the scores which should be reached by the models with less detail.

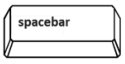
Procedure:
Different objects will be shown to you. They will rotate so they can be seen from more angles. Rotation can be stopped via enter button. The object constantly switches between original model and LOD.

Regarding the thresholds a linear behavior in respect to the distance to the camera is assumed. Therefore, first the starting threshold is evaluated, and then different slope settings are evaluated. For the second phase, the object can be moved further away in order to evaluate the current slope setting.

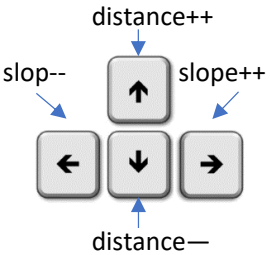
Your Action:
(instructions also visible in the HMD while evaluating)

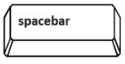
1. Evaluate Starting Threshold



when you are fine with your selection hit 

2. Evaluate Slope Setting



when you are fine with your selection hit 

Decisions can always be undone with the backspace button!

Repeat above steps for each object in the scene (10 in total). After evaluating both settings, the next object appears. The application exits when all object in the scene have been tested. Repeat the above procedure for the next HMD.

Pick up your well-deserved chocolate 😉 Thank you!

Gregor Liebisch
gregor.liebisch@student.tugraz.at

Consent Form

I agree to participate in the study conducted by the ICG – Institute of Computer Graphics and Vision (TU Graz).

I understand that participation in this usability study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session with the study administrator.

Please sign below to indicate that you have read and you understand the information on this form and that any questions you might have about the session have been answered.

Date:_____

Please print your name: _____

Please sign your name: _____

Thank you!

We appreciate your participation.

Gender (select): ☐ m ☐ f

Age: _____

Profession: _____

Corrected Vision (select): ☐ Yes ☐ No

How do you rate your experience with computers in general (1 ... very low, 5 ... very high)?

1	2	3	4	5

How do you rate your experience with VR applications (1 ... very low, 5 ... very high)?

1	2	3	4	5

Notes _____

hmd	model	index	d	k
rift	chair	0	1,0000	0,75
rift	orchid	0	1,0000	0,25
rift	falcon	3	0,8968	0,75
rift	elephant	3	0,8783	0,75
rift	elephant_plain	3	0,8826	1,25
rift	spot_lod	3	0,9378	0,75
rift	mustang	3	0,8672	0,75
rift	dragon	2	0,9412	0,75
rift	buddha	4	0,8010	1,50
rift	buddha_plain	2	0,9453	0,75
rift	chair	3	0,7261	0,00
rift	orchid	3	0,8589	0,75
rift	falcon	3	0,8968	0,25
rift	elephant	2	0,9164	0,50
rift	elephant_plain	1	0,9739	0,25
rift	spot_lod	2	0,9612	0,75
rift	mustang	3	0,8672	0,25
rift	dragon	1	0,9741	0,00
rift	buddha	1	0,9757	-0,25
rift	buddha_plain	1	0,9804	0,00
rift	chair	1	0,8226	-0,25
rift	orchid	2	0,9123	0,00
rift	falcon	3	0,8968	0,75
rift	elephant	3	0,8783	0,25
rift	elephant_plain	1	0,9739	-0,25
rift	spot_lod	2	0,9612	0,75
rift	mustang	2	0,9039	0,25
rift	dragon	0	1,0000	0,00
rift	buddha	1	0,9757	-0,25
rift	buddha_plain	1	0,9804	0,75
rift	chair	1	0,8226	0,25
rift	orchid	2	0,9123	0,00
rift	falcon	2	0,9280	0,00
rift	elephant	2	0,9164	1,00
rift	elephant_plain	1	0,9739	-0,25
rift	spot_lod	1	0,9872	0,50
rift	mustang	2	0,9039	0,00
rift	dragon	1	0,9741	0,75
rift	buddha	1	0,9757	0,25
rift	buddha_plain	1	0,9804	0,00
rift	chair	2	0,7899	1,00
rift	orchid	6	0,7001	0,75
rift	falcon	6	0,6936	1,00
rift	elephant	7	0,6118	1,00
rift	elephant_plain	5	0,7923	0,75
rift	spot_lod	8	0,7232	1,25
rift	mustang	3	0,8672	0,75
rift	dragon	6	0,7693	1,25
rift	buddha	4	0,8010	1,25
rift	buddha_plain	3	0,8991	1,00
rift	chair	1	0,8226	-0,25
rift	orchid	2	0,9123	0,50
rift	falcon	2	0,9280	0,50
rift	elephant	3	0,8783	0,25
rift	elephant_plain	1	0,9739	-0,25
rift	spot_lod	3	0,9378	0,25
rift	mustang	2	0,9039	0,25
rift	dragon	2	0,9412	0,50
rift	buddha	2	0,9318	-0,25
rift	buddha_plain	0	1,0000	0,00
rift	chair	3	0,7261	-0,25
rift	orchid	3	0,8589	0,00

hmd	model	index	d	k
vive	chair	4	0,6803	0,25
vive	orchid	3	0,8589	0,75
vive	falcon	3	0,8968	0,25
vive	elephant	3	0,8783	0,75
vive	elephant_plain	1	0,9739	-0,25
vive	spot_lod	2	0,9612	-0,25
vive	mustang	3	0,8672	1,00
vive	dragon	1	0,9741	1,00
vive	buddha	1	0,9757	0,75
vive	buddha_plain	1	0,9804	-0,25
vive	chair	2	0,7899	0,75
vive	orchid	3	0,8589	0,75
vive	falcon	1	0,9696	0,75
vive	elephant	2	0,9164	0,75
vive	elephant_plain	3	0,8826	0,50
vive	spot_lod	2	0,9612	0,00
vive	mustang	3	0,8672	1,25
vive	dragon	3	0,8944	1,50
vive	buddha	1	0,9757	0,75
vive	buddha_plain	2	0,9453	0,00
vive	chair	2	0,7899	0,50
vive	orchid	2	0,9123	0,00
vive	falcon	3	0,8968	0,75
vive	elephant	3	0,8783	0,50
vive	elephant_plain	2	0,9275	0,25
vive	spot_lod	2	0,9612	0,50
vive	mustang	3	0,8672	0,50
vive	dragon	3	0,8944	0,75
vive	buddha	1	0,9757	0,00
vive	buddha_plain	1	0,9804	0,00
vive	chair	1	0,8226	-0,25
vive	orchid	1	0,9810	0,50
vive	falcon	2	0,9280	0,25
vive	elephant	2	0,9164	0,50
vive	elephant_plain	1	0,9739	-0,25
vive	spot_lod	1	0,9872	0,00
vive	mustang	1	0,9476	0,25
vive	dragon	0	1,0000	0,25
vive	buddha	1	0,9757	0,00
vive	buddha_plain	1	0,9804	0,00
vive	chair	1	0,8226	0,00
vive	orchid	2	0,9123	0,25
vive	falcon	2	0,9280	0,00
vive	elephant	2	0,9164	0,00
vive	elephant_plain	0	1,0000	-0,25
vive	spot_lod	4	0,9194	0,00
vive	mustang	4	0,8036	-0,25
vive	dragon	3	0,8944	1,25
vive	buddha	3	0,8629	-0,25
vive	buddha_plain	0	1,0000	0,00
vive	chair	6	0,5328	0,75
vive	orchid	5	0,7862	0,50
vive	falcon	5	0,7818	0,25
vive	elephant	6	0,6600	1,00
vive	elephant_plain	3	0,8826	0,25
vive	spot_lod	5	0,8674	0,75
vive	mustang	5	0,7505	-0,25
vive	dragon	4	0,8669	0,75
vive	buddha	2	0,9318	0,50
vive	buddha_plain	1	0,9804	0,25
vive	chair	1	0,8226	0,75
vive	orchid	2	0,9123	0,50

rift	falcon	3	0,8968	0,25
rift	elephant	3	0,8783	0,00
rift	elephant_plain	1	0,9739	0,00
rift	spot_lod	1	0,9872	0,00
rift	mustang	3	0,8672	0,00
rift	dragon	1	0,9741	0,50
rift	buddha	1	0,9757	0,25
rift	buddha_plain	1	0,9804	0,00
rift	chair	1	0,8226	0,25
rift	orchid	3	0,8589	0,25
rift	falcon	2	0,9280	1,25
rift	elephant	2	0,9164	0,75
rift	elephant_plain	1	0,9739	0,50
rift	spot_lod	3	0,9378	0,00
rift	mustang	2	0,9039	0,25
rift	dragon	1	0,9741	0,75
rift	buddha	1	0,9757	0,50
rift	buddha_plain	1	0,9804	0,00
rift	chair	2	0,7899	0,25
rift	orchid	2	0,9123	0,25
rift	falcon	4	0,8256	-0,25
rift	elephant	4	0,7987	0,25
rift	elephant_plain	2	0,9275	0,25
rift	spot_lod	4	0,9194	0,00
rift	mustang	2	0,9039	0,25
rift	dragon	4	0,8669	0,75
rift	buddha	2	0,9318	0,00
rift	buddha_plain	1	0,9804	0,00
rift	chair	0	1,0000	0,00
rift	orchid	1	0,9810	-0,25
rift	falcon	1	0,9696	0,00
rift	elephant	2	0,9164	-0,25
rift	elephant_plain	0	1,0000	0,00
rift	spot_lod	0	1,0000	0,00
rift	mustang	1	0,9476	-0,25
rift	dragon	1	0,9741	0,00
rift	buddha	1	0,9757	-0,25
rift	buddha_plain	1	0,9804	-0,25
rift	chair	1	0,8226	0,50
rift	orchid	1	0,9810	-0,25
rift	falcon	1	0,9696	0,00
rift	elephant	0	1,0000	0,00
rift	elephant_plain	0	1,0000	0,00
rift	spot_lod	2	0,9612	-0,25
rift	mustang	1	0,9476	0,00
rift	dragon	1	0,9741	0,00
rift	buddha	1	0,9757	-0,25
rift	buddha_plain	1	0,9804	-0,25
rift	chair	0	1,0000	0,50
rift	orchid	2	0,9123	0,00
rift	falcon	4	0,8256	0,00
rift	elephant	4	0,7987	0,00
rift	elephant_plain	2	0,9275	0,00
rift	spot_lod	2	0,9612	0,00
rift	mustang	4	0,8036	0,00
rift	dragon	5	0,8148	0,25
rift	buddha	2	0,9318	0,25
rift	buddha_plain	2	0,9453	-0,25
rift	chair	2	0,7899	-0,25
rift	orchid	1	0,9810	0,50
rift	falcon	2	0,9280	0,25
rift	elephant	2	0,9164	0,00
rift	elephant_plain	0	1,0000	0,00

vive	falcon	1	0,9696	0,50
vive	elephant	3	0,8783	0,25
vive	elephant_plain	1	0,9739	0,50
vive	spot_lod	2	0,9612	0,50
vive	mustang	2	0,9039	0,25
vive	dragon	4	0,8669	1,50
vive	buddha	1	0,9757	0,25
vive	buddha_plain	2	0,9453	0,50
vive	chair	1	0,8226	0,50
vive	orchid	2	0,9123	0,00
vive	falcon	2	0,9280	0,50
vive	elephant	3	0,8783	0,00
vive	elephant_plain	1	0,9739	0,00
vive	spot_lod	3	0,9378	0,00
vive	mustang	2	0,9039	0,25
vive	dragon	1	0,9741	0,25
vive	buddha	1	0,9757	0,25
vive	buddha_plain	2	0,9453	-0,25
vive	chair	2	0,7899	0,00
vive	orchid	2	0,9123	0,25
vive	falcon	3	0,8968	0,50
vive	elephant	3	0,8783	0,25
vive	elephant_plain	2	0,9275	0,00
vive	spot_lod	3	0,9378	0,25
vive	mustang	3	0,8672	0,25
vive	dragon	3	0,8944	0,25
vive	buddha	2	0,9318	0,25
vive	buddha_plain	1	0,9804	-0,25
vive	chair	6	0,5328	0,25
vive	orchid	2	0,9123	0,75
vive	falcon	5	0,7818	0,75
vive	elephant	3	0,8783	0,75
vive	elephant_plain	1	0,9739	1,25
vive	spot_lod	0	1,0000	0,75
vive	mustang	2	0,9039	0,75
vive	dragon	0	1,0000	0,75
vive	buddha	0	1,0000	-0,50
vive	buddha_plain	0	1,0000	0,00
vive	chair	0	1,0000	0,00
vive	orchid	1	0,9810	0,00
vive	falcon	2	0,9280	0,25
vive	elephant	1	0,9638	-0,25
vive	elephant_plain	0	1,0000	0,00
vive	spot_lod	0	1,0000	0,00
vive	mustang	1	0,9476	-0,25
vive	dragon	0	1,0000	0,00
vive	buddha	0	1,0000	0,00
vive	buddha_plain	1	0,9804	-0,25
vive	chair	4	0,6803	-0,25
vive	orchid	2	0,9123	0,00
vive	falcon	4	0,8256	-0,25
vive	elephant	2	0,9164	-0,25
vive	elephant_plain	0	1,0000	0,00
vive	spot_lod	1	0,9872	0,00
vive	mustang	2	0,9039	0,25
vive	dragon	0	1,0000	0,00
vive	buddha	1	0,9757	0,00
vive	buddha_plain	0	1,0000	0,00
vive	chair	4	0,6803	-0,25
vive	orchid	2	0,9123	0,00
vive	falcon	4	0,8256	0,00
vive	elephant	3	0,8783	-0,25
vive	elephant_plain	2	0,9275	0,25

rift	spot_lod	1	0,9872	0,00
rift	mustang	2	0,9039	0,50
rift	dragon	0	1,0000	0,00
rift	buddha	1	0,9757	-0,25
rift	buddha_plain	1	0,9804	-0,25
rift	chair	1	0,8226	0,25
rift	orchid	1	0,9810	0,25
rift	falcon	3	0,8968	0,50
rift	elephant	4	0,7987	0,25
rift	elephant_plain	1	0,9739	1,75
rift	spot_lod	5	0,8674	0,75
rift	mustang	6	0,7014	0,25
rift	dragon	4	0,8669	0,00
rift	buddha	3	0,8629	0,25
rift	buddha_plain	3	0,8991	0,25
rift	chair	2	0,7899	0,25
rift	orchid	2	0,9123	0,00
rift	falcon	2	0,9280	0,00
rift	elephant	1	0,9638	-0,25
rift	elephant_plain	0	1,0000	0,00
rift	spot_lod	2	0,9612	0,00
rift	mustang	2	0,9039	0,25
rift	dragon	1	0,9741	-0,25
rift	buddha	1	0,9757	0,25
rift	buddha_plain	1	0,9804	0,00
rift	chair	6	0,5328	0,00
rift	orchid	2	0,9123	0,00
rift	falcon	1	0,9696	0,00
rift	elephant	1	0,9638	0,50
rift	elephant_plain	2	0,9275	0,25
rift	spot_lod	3	0,9378	0,50
rift	mustang	3	0,8672	0,25
rift	dragon	2	0,9412	0,50
rift	buddha	1	0,9757	0,25
rift	buddha_plain	1	0,9804	0,25
rift	chair	3	0,7261	0,25
rift	orchid	5	0,7862	0,25
rift	falcon	5	0,7818	0,00
rift	elephant	5	0,7279	0,25
rift	elephant_plain	1	0,9739	0,75
rift	spot_lod	4	0,9194	0,75
rift	mustang	3	0,8672	0,75
rift	dragon	5	0,8148	0,75
rift	buddha	1	0,9757	0,75
rift	buddha_plain	1	0,9804	0,00
rift	chair	2	0,7899	0,25
rift	orchid	3	0,8589	0,25
rift	falcon	4	0,8256	0,25
rift	elephant	4	0,7987	0,25
rift	elephant_plain	2	0,9275	0,25
rift	spot_lod	3	0,9378	0,00
rift	mustang	2	0,9039	0,00
rift	dragon	1	0,9741	0,00
rift	buddha	1	0,9757	0,00
rift	buddha_plain	1	0,9804	0,00
rift	chair	4	0,6803	0,50
rift	orchid	3	0,8589	0,50
rift	falcon	2	0,9280	1,00
rift	elephant	4	0,7987	0,25
rift	elephant_plain	0	1,0000	0,00
rift	spot_lod	4	0,9194	1,00
rift	mustang	4	0,8036	1,25
rift	dragon	2	0,9412	0,25
vive	spot_lod	3	0,9378	0,50
vive	mustang	4	0,8036	0,00
vive	dragon	5	0,8148	0,50
vive	buddha	2	0,9318	0,25
vive	buddha_plain	1	0,9804	-0,25
vive	chair	7	0,4823	0,25
vive	orchid	5	0,7862	0,00
vive	falcon	1	0,9696	0,25
vive	elephant	1	0,9638	0,00
vive	elephant_plain	0	1,0000	-0,25
vive	spot_lod	1	0,9872	0,75
vive	mustang	3	0,8672	-0,25
vive	dragon	2	0,9412	0,50
vive	buddha	5	0,7460	0,00
vive	buddha_plain	0	1,0000	0,00
vive	chair	3	0,7261	0,00
vive	orchid	2	0,9123	0,50
vive	falcon	4	0,8256	0,25
vive	elephant	5	0,7279	0,25
vive	elephant_plain	3	0,8826	0,25
vive	spot_lod	5	0,8674	0,50
vive	mustang	4	0,8036	-0,50
vive	dragon	6	0,7693	0,50
vive	buddha	2	0,9318	-0,25
vive	buddha_plain	3	0,8991	-0,25
vive	chair	3	0,7261	0,25
vive	orchid	0	1,0000	0,75
vive	falcon	0	1,0000	0,75
vive	elephant	8	0,5041	0,50
vive	elephant_plain	1	0,9739	0,75
vive	spot_lod	7	0,7663	0,75
vive	mustang	4	0,8036	0,00
vive	dragon	4	0,8669	0,75
vive	buddha	2	0,9318	0,75
vive	buddha_plain	1	0,9804	0,75
vive	chair	4	0,6803	1,00
vive	orchid	2	0,9123	0,25
vive	falcon	2	0,9280	0,25
vive	elephant	2	0,9164	0,25
vive	elephant_plain	1	0,9739	0,50
vive	spot_lod	2	0,9612	0,50
vive	mustang	2	0,9039	0,25
vive	dragon	2	0,9412	0,50
vive	buddha	1	0,9757	0,25
vive	buddha_plain	3	0,8991	0,00
vive	chair	2	0,7899	0,00
vive	orchid	2	0,9123	0,00
vive	falcon	2	0,9280	0,00
vive	elephant	2	0,9164	0,25
vive	elephant_plain	1	0,9739	0,00
vive	spot_lod	2	0,9612	0,25
vive	mustang	2	0,9039	0,00
vive	dragon	2	0,9412	-0,25
vive	buddha	1	0,9757	0,00
vive	buddha_plain	1	0,9804	0,00
vive	chair	1	0,8226	-0,50
vive	orchid	2	0,9123	0,25
vive	falcon	3	0,8968	0,00
vive	elephant	1	0,9638	-0,25
vive	elephant_plain	1	0,9739	0,25
vive	spot_lod	3	0,9378	0,00
vive	mustang	2	0,9039	0,25
vive	dragon	3	0,8944	0,00

rift	buddha	1	0,9757	0,75
rift	buddha_plain	0	1,0000	0,25
rift	chair	1	0,8226	0,00
rift	orchid	2	0,9123	1,00
rift	falcon	2	0,9280	0,50
rift	elephant	2	0,9164	0,00
rift	elephant_plain	1	0,9739	-0,25
rift	spot_lod	3	0,9378	0,25
rift	mustang	4	0,8036	0,25
rift	dragon	4	0,8669	0,00
rift	buddha	1	0,9757	0,25
rift	buddha_plain	1	0,9804	-0,25
rift	chair	3	0,7261	0,50
rift	orchid	4	0,8306	0,50
rift	falcon	5	0,7818	0,50
rift	elephant	5	0,7279	0,50
rift	elephant_plain	3	0,8826	-0,25
rift	spot_lod	4	0,9194	0,25
rift	mustang	5	0,7505	-0,25
rift	dragon	4	0,8669	0,75
rift	buddha	1	0,9757	0,50
rift	buddha_plain	1	0,9804	0,25
rift	chair	1	0,8226	0,00
rift	orchid	4	0,8306	-0,50
rift	falcon	2	0,9280	0,75
rift	elephant	2	0,9164	0,25
rift	elephant_plain	1	0,9739	0,00
rift	spot_lod	2	0,9612	0,50
rift	mustang	2	0,9039	0,25
rift	dragon	6	0,7693	0,75
rift	buddha	3	0,8629	0,00
rift	buddha_plain	1	0,9804	0,25
rift	chair	0	1,0000	0,25
rift	orchid	2	0,9123	0,75
rift	falcon	1	0,9696	0,50
rift	elephant	1	0,9638	1,00
rift	elephant_plain	0	1,0000	0,00
rift	spot_lod	1	0,9872	0,25
rift	mustang	1	0,9476	0,00
rift	dragon	1	0,9741	0,00
rift	buddha	1	0,9757	0,25
rift	buddha_plain	0	1,0000	0,00
rift	chair	3	0,7261	0,00
rift	orchid	3	0,8589	0,25
rift	falcon	3	0,8968	0,25
rift	elephant	3	0,8783	0,50
rift	elephant_plain	2	0,9275	0,25
rift	spot_lod	4	0,9194	0,50
rift	mustang	4	0,8036	0,00
rift	dragon	6	0,7693	0,50
rift	buddha	2	0,9318	0,00
rift	buddha_plain	2	0,9453	-0,25
rift	chair	3	0,7261	0,50
rift	orchid	2	0,9123	0,00
rift	falcon	3	0,8968	0,25
rift	elephant	1	0,9638	0,50
rift	elephant_plain	1	0,9739	0,00
rift	spot_lod	1	0,9872	0,25
rift	mustang	3	0,8672	0,25
rift	dragon	1	0,9741	0,00
rift	buddha	1	0,9757	0,00
rift	buddha_plain	1	0,9804	0,00

vive	buddha	2	0,9318	-0,25
vive	buddha_plain	1	0,9804	0,00
vive	chair	4	0,6803	0,25
vive	orchid	4	0,8306	1,00
vive	falcon	4	0,8256	1,25
vive	elephant	4	0,7987	1,25
vive	elephant_plain	1	0,9739	1,25
vive	spot_lod	5	0,8674	0,75
vive	mustang	3	0,8672	0,50
vive	dragon	3	0,8944	0,50
vive	buddha	3	0,8629	1,25
vive	buddha_plain	1	0,9804	1,25
vive	chair	1	0,8226	-0,25
vive	orchid	3	0,8589	-0,25
vive	falcon	4	0,8256	0,50
vive	elephant	3	0,8783	0,25
vive	elephant_plain	0	1,0000	0,75
vive	spot_lod	1	0,9872	1,00
vive	mustang	2	0,9039	0,75
vive	dragon	6	0,7693	0,50
vive	buddha	3	0,8629	0,50
vive	buddha_plain	1	0,9804	-0,50
vive	chair	4	0,6803	0,00
vive	orchid	3	0,8589	0,25
vive	falcon	3	0,8968	0,00
vive	elephant	3	0,8783	0,25
vive	elephant_plain	3	0,8826	-0,25
vive	spot_lod	3	0,9378	0,00
vive	mustang	3	0,8672	0,00
vive	dragon	4	0,8669	1,00
vive	buddha	2	0,9318	0,25
vive	buddha_plain	2	0,9453	0,00
vive	chair	4	0,4823	0,50
vive	orchid	3	0,8589	0,75
vive	falcon	3	0,8256	0,50
vive	elephant	4	0,7987	0,50
vive	elephant_plain	4	0,8452	0,25
vive	spot_lod	4	0,9194	0,50
vive	mustang	2	0,9039	0,50
vive	dragon	4	0,8669	1,50
vive	buddha	2	0,9318	0,50
vive	buddha_plain	1	0,9804	0,00
vive	chair	1	0,8226	0,00
vive	orchid	1	0,9810	1,00
vive	falcon	1	0,9696	0,25
vive	elephant	1	0,9638	-0,25
vive	elephant_plain	0	1,0000	-0,25
vive	spot_lod	1	0,9872	-0,25
vive	mustang	4	0,8036	-0,25
vive	dragon	1	0,9741	-0,25
vive	buddha	0	1,0000	0,00
vive	buddha_plain	1	0,9804	-0,25
vive	chair	2	0,7899	0,25
vive	orchid	2	0,9123	0,00
vive	falcon	2	0,9280	0,00
vive	elephant	2	0,9164	0,25
vive	elephant_plain	1	0,9739	0,00
vive	spot_lod	2	0,9612	0,25
vive	mustang	2	0,9039	0,25
vive	dragon	1	0,9741	0,00
vive	buddha	3	0,8629	0,25
vive	buddha_plain	2	0,9453	0,25