university of groningen

faculty of science and engineering

A prediction model for LOD reduction subject to eccentricity and lighting

Bachelor's Thesis Computing Science

July 2024

Author: Tait van Strien
Student Number: S4398467
First supervisor: Cara Tursun
Second supervisor: Jiri Kosinka

Abstract

Level-of-detail (LOD) rendering is a technique in computer graphics that defines several 'levels' of a model's complexity, allowing a lower complexity representation and, therefore, less computationally expensive rendering. Traditionally, developers use LOD techniques to achieve higher performance without losing perceived visual fidelity as a function of distance. New technologies such as Virtual Reality (VR), Augmented Reality (AR), eye tracking, and widescreen displays are far more demanding for high-quality rendering due to their increased field of view. The Human Visual System (HVS) has non-uniform sensitivity across the visual field; how we perceive objects in the peripheral vision differs from the central vision (fovea) to the periphery. As these new technologies occupy a larger proportion of field of view (FOV) and our perception across the display is non-uniform, There is an opportunity to use LOD rendering as a function of eccentricity, changing the detail of a model across the FOV such that we can achieve a lower computational cost without a loss of perceived visual fidelity. This concept is known as foveated rendering using LOD and is often used in accompaniment with eye trackers to render the peripheral vision at a lower LOD than the fovea.

The variation across the HVS is affected by various visual factors, such as colour, texture, the lighting parameters in the virtual scene, and temporal changes. As such, we have created an experiment framework and conducted preliminary experiments into the effects of ambient strength, specular strength, lighting position, eccentricity and level of detail reduction on the perception of detail, developing a logistic regression model for visibility of LOD reduction.

Contents

List of Figures

List of Tables

1 Introduction

1.1 Background

A mainstay of modern rendering engines, level of detail meshes allow high-fidelity environments to be rendered with a minimal performance impact. Typically, mesh simplification techniques are used as a function of distance from the camera to an object in three-dimensional (3D) space. However, recent developments in VR and AR have introduced a demand for portable, high-fidelity computing, as evidenced by the development of headsets with wide fields of view. These headsets and their wide field-of-view displays are very computationally demanding and, as such, require more aggressive optimisation relative to lower field-of-view displays.

The Human Visual System (HVS) and the perception of visual stimuli vary dramatically over the field of view. Our perception of colour, texture, lighting and temporal changes varies throughout our field of view. As such, we can exploit these differences in perception to reduce the geometric complexity of objects in 3D space without a loss of visual fidelity, thereby reducing processing requirements and improving performance. We can use this to improve the performance in wide field-of-view displays by using LOD reduction at higher eccentricity.

In this project, we take inspiration from existing work by Surace et al. [34]. Their research into the perception of the level of detail in increasing eccentricity, which investigated the relationship between the perception of LOD reduction through mesh simplification of an object, is particularly relevant to our study. We extend their experimentation to investigate the effects of lighting conditions, leading us to a few questions we sought to answer with our experiment:

- How do Specular highlights and ambient strength affect the visually tolerable amount of LOD mesh reduction across different eccentricities on the screen?
- How does light angle affect the visually tolerable amount of LOD mesh reduction across different eccentricities on the screen?
- How does eccentricity affect the visually tolerable LOD?

1.2 Aim of the Paper

This paper specialises in analysing the effects of ambient lighting strength, specular highlights and lighting position on the perception of the level of detail in peripheral vision. Using this data, we fit a machine-learning model for the effects of lighting conditions on the visual saliency of mesh reduction. As a result of these findings, researchers can discern new optimisation techniques for 3D rendering, improving performance in these portable computing environments and allowing developers to create higher fidelity 3D worlds without sacrificing

computational performance.

1.3 Proposal

We propose an experiment framework to collect perceptual data from human participants and build a prediction model for the visually tolerable amount of LOD reduction for an arbitrary mesh as a function of the distance to the gaze position on a screen and given parameters of the virtual light source. This experiment involves presenting the participant with two intervals of an object under a lighting condition at an eccentricity at two different levels of detail and asking them to identify the higher fidelity image. With this experiment, we can develop a model for the effects of lighting conditions on the visual saliency of mesh reduction. As a result of these findings, researchers can discern new optimisation techniques for 3D rendering, improving performance in these portable computing environments and allowing developers to create higher fidelity 3D worlds without sacrificing computational performance.

1.4 Paper Structure

Section 2 provides a comprehensive overview of the state-of-the-art in peripheral perception, level-of-detail techniques, and the increasing relevance of VR and AR. Section 3 meticulously details the preprocessing steps to prepare the 3D models and algorithms before the experiment was conducted. Section 4 outlines the experiment, its implementation, and key components, demonstrating the thoroughness of our approach. Section 5 offers a meticulous analysis of the collected data and the creation of the logistic model. Section 6 discusses our findings from the experiment and the results of the analysis. Finally, section 7 concludes our findings, highlighting the limitations and future work explored in section 8.

2 Related Work

2.1 Level of Detail reduction using geometric simplification

As an object moves further from the camera, it occupies fewer pixels on the screen. As such, it requires fewer polygons to represent an object at a distance with equal fidelity to a close object. To take advantage of these variations in perception, many algorithms have been developed to approximate object meshes as a function of the area of the screen taken by the object [10]. Vertex clustering is a simple LOD algorithm, first introduced by Rossignac [30], which assigns a weight to every vertex in a model, such as a higher weight for vertices attached to large faces. The algorithm then clusters vertices from triangles into lines or points, creating a simplified model. As Schroeder et al. [31] described, Vertex decimation utilises vertex removal considering the vertex's location in the object, such as being on a corner or boundary. However, the distortion from the original object can be noticeable on large terrains. As such, Garland et al. [14] introduce the concept of a greedy insertion algorithm to simplify a terrain mesh through a multi-pass approximation and simplification of the terrain. This laid the groundwork for Quadric error metrics, where the cumulative surface distortion is measured by a series of vertex merge operations, iteratively merging vertices until it reaches a desired level of simplification. [15] Both Quadric error metrics and vertex clustering inspired Brodsky et al. [7] to create a greedy reverse simplification framework that utilises 'clusters', representing a surface patch on the input model and a single vertex in the output model. These 'clusters' are then repeatedly split according to their surface normal variation error metric until the desired number of vertices is achieved. Later developments by Lindstrom et al. [21] suggest using features of the image space to drive simplification, allowing the collapsing of edges ranked by an estimation of the effects on the image, taking normals, colour and texture into consideration. These rendering methods have become common in physics and game engines such as Unity's level of detail [38] and the unreal engine nanite mesh [37]. While these techniques successfully improve performance for objects in 3D space, they do not factor in eccentricity.

2.2 Advent of VR

The advent of VR and AR introduced new design principles for peripherally degraded displays and displays with a far greater field of view, which has a higher computational cost [41]. VR and AR adoption has followed a steadily rising trend of adoption with the introduction of more affordable consumer headsets from companies such as Sony [32], Meta [23] and Apple [6]. As such, there is more demand than ever for higher fidelity imaging at lower performance requirements. However, Virtual and Augmented Reality is far more demanding due to a higher field of view, a requirement for lower latency and hardware limitations [40]. Achieving higher fidelity imaging while maintaining framerates has long been one of the most pressing problems facing virtual and Augmented Reality.

2.3 Optimisation using the human visual system

The Human Visual System (HVS) opens opportunities for optimisation in rendering. VR and AR use displays that occupy a far larger area of our field of view; however, our vision cannot interpret detail in every aspect of our field of view as processing across the visual field is not homogeneous due to the structure of the eye. The cone photoreceptor density peaks in the fovea and declines toward the periphery [33], changing how we perceive stimuli at differing eccentricity.

With increasing eccentricity, the interpretation of graphics in the HVS changes. Colour is one example; perception in the fovea is strongest and declines towards the periphery [16] [41]. To take advantage of this, techniques to reduce the level of detail in the periphery have been created.

Foveated rendering is a technique in which eccentricity is a factor in selecting the visual quality of the rendering, thus reducing required computation and increasing performance. Early works into this concept assumed participants held focus at the centre of a display [13], a technique called fixed-foveated rendering. With the increased prevalence of eye-tracking devices and software, these experiments were confirmed in later works where eccentricity layers were utilised to produce a perceptual model of resolution reduction. Sony's most recent VR headset, the PSVR 2, uses a version of foveated rendering to improve the performance of supported titles. The Unity engine developers found that "with foveated rendering, the PSVR 2 can be up to 2.5x faster, without any perceptual loss compared to equivalent image quality through standard stereo rendering" [5]. This performance gain can additionally be achieved with little loss in perceived quality through understanding visual salience for colour [11], luminance [39] and topographical methods [27].

Murphy et al. [26] present a nonisotropic model-based rendering technique for gaze-contingent mesh simplification to address this need for eccentricity-based rendering, resulting in significant performance improvements. Reddy [29] builds on this idea and introduces a method for LOD in the image space rather than the geometric space, resulting in a more visually accurate image for what the user sees. Later, Surace et al. [34] began investigating the effects of varying virtual environment conditions, applying a simple LOD technique and assuming the worst-case scenario for a set of textures. However, this model is limited as a single technique calibrates it, is only studied over limited textures, and does not factor in lighting conditions [18]. As such, we propose a purpose-built flexible experimental framework explicitly designed for 3D psychophysics vision experimentation built on top of existing frameworks. While these models display improved performance over traditional rendering, their limits under various conditions, such as lighting and colour, are currently understudied. As such, additional experimentation and investigation into the limits of acceptable LOD reduction and a framework to perform these experiments are required.

2.4 Existing experimental frameworks

Existing frameworks for psychophysics exist as libraries for MATLAB [36] and Python [35]. However, these frameworks are intended for all behavioural sciences. As such, they are flexible but require additional program construction to be utilised in our area of interest. PsychoPy is a Python library and, as such, is quick to develop due to Python's easy syntax; however, its support for 3D models is limited. Pysychotoolbox is a similar framework built for Matlab, making working natively with experimental data easier. Brooks et al. [8] has created a framework for Unity that allows data collection using Unity's 3D object designer and shaders. Alternatively, should we require a more customisable approach, Regl is a library for simply interacting with webGL using javascript [28]. All these frameworks support exporting to webGL, allowing us to build the experiment on various hardware.

We propose a framework for studying the effects of textures, lighting, shaders, contrast, and geometric simplification in peripheral vision. As such, we can find the limits and nuances of geometric simplification within the image space. This will allow us to investigate the effects of lighting conditions on the perception of mesh simplification and create a framework that can be used for future experimentation.

3 Model Pre-processing

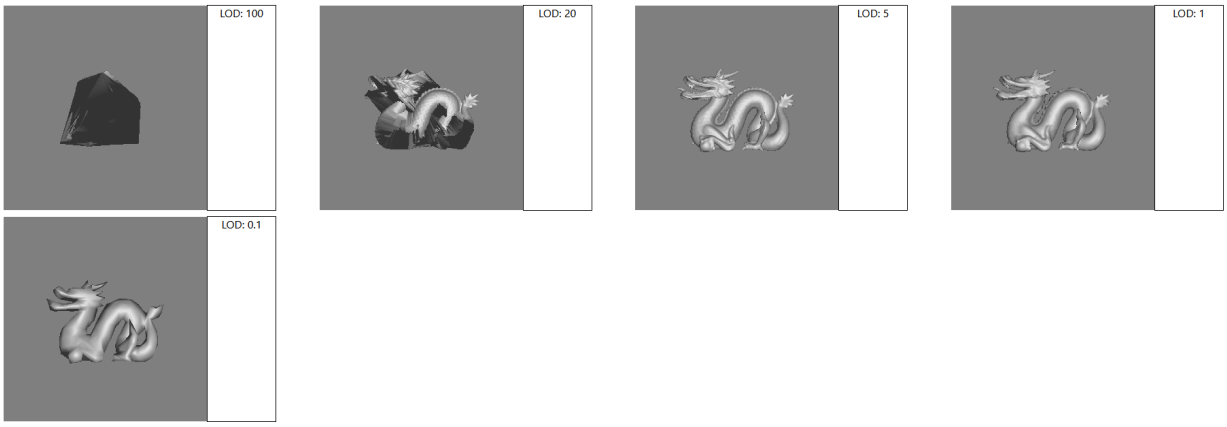

Figure 1: renderings of the original dragon model after going through model cleaning

Before experimenting, we required a few 3D models to simplify and evaluate lighting conditions. The models downloaded from the Stanford model repository [4] were 3D scans of real-world models. As a result, the scans contained many overlapping vertices and some fragments. To resolve this, we imported the meshes into pymeshlab, centred the mesh on the origin, and cleaned the model by removing duplicate and unreferenced vertices. After the meshes were cleaned, we used pymeshlab's implementation of M. Garland and P. Heckbert quadric mesh simplification by vertex decimation [15] to produce meshes at the reduced levels of detail for the two models, as shown for both bunny and dragon models in Fig 3 and 2.

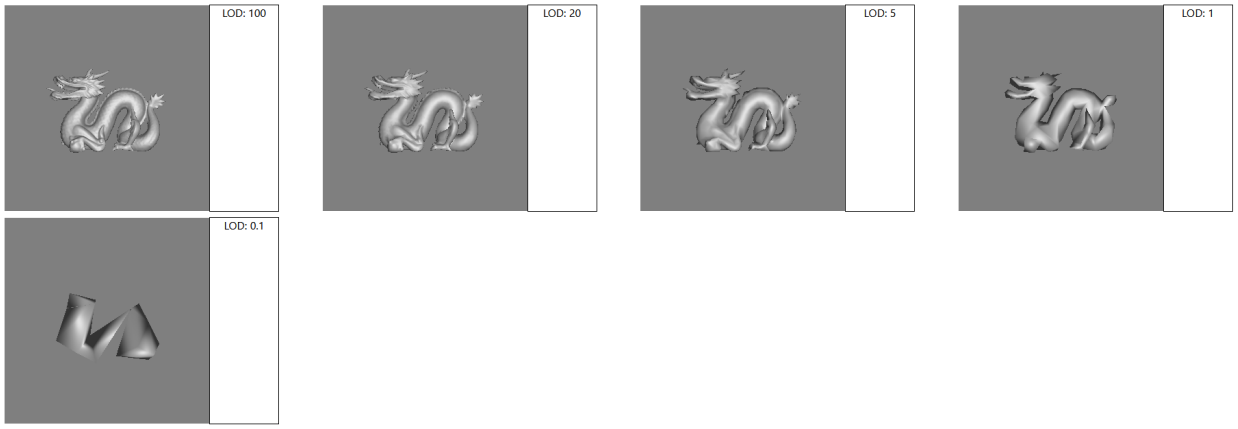

Figure 2: renderings of the levels of detail of the dragon model

The algorithm is based on the iterative contraction of vertex pairs and error quadrics. The basic algorithm is as follows:

1. Compute the Q matrices for all the initial vertices

2. select all valid pairs

3. Compute the optimal contraction target $\bar{x}$ for each valid pair (v_1, v_2). The error $\bar{v}^T(Q_1 + Q_2)\bar{v}$ of this target vertex becomes the cost of contracting that pair.

4. Place all the pairs in a heap keyed on cost with the minimum cost pair at the top.

5. Iteratively remove the pair (v_1, v_2) of least cost from the heap, contract this pair, and update the costs of all valid pairs involving v_1.

For the initial Q value, each vertex must accumulate the planes for the triangles which meet at that vertex and have an initial error estimate for each vertex of 0. As each vertex lies in the planes of all its incident triangles.

We encountered a problem with cleaning and simplifying the dragon model where we struggled to identify the vertices causing artefacts in the dragon model, shown in Fig.1. As such, we set our base level of detail for the dragon model to be 5% of the original scan.

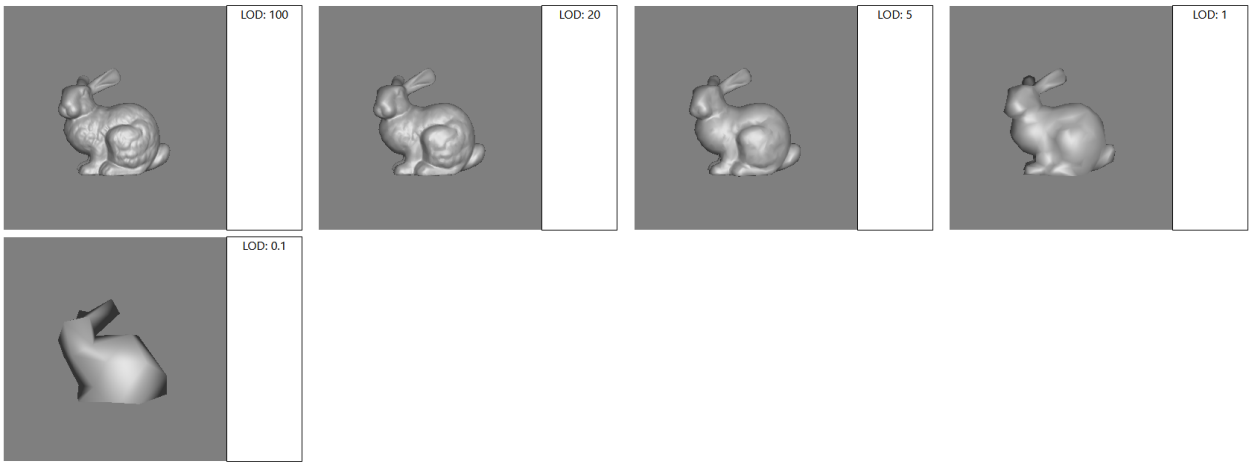

Figure 3: renderings of the levels of detail of the bunny model

While there are many methods for producing models with different levels of detail, such as vertex clustering [30] or progressive meshes [17], We chose mesh simplification by vertex decimation [15]. It is computationally efficient and minimizes the error introduced during the simplification process by using a quadratic error metric, ensuring the geometric difference between the original and the simplified mesh is as close as possible. When processing the models, we used the default parameters for pymeshlab, with the target percentage (targetperc) being set to the percentage reduction, as demonstrated in Tab. 1.

variable	value	description
targetperc	1.0-0	The desired final size of the mesh as a percentage of the initial size.
qualitythr	0.3	Quality threshold for penalizing bad shaped faces.
preserveboundary	false	The simplification process tries not to affect mesh boundaries during simplification.
boundaryweight	1	The importance of the boundary during simplification, one means that the boundary has the same importance as the rest.
preservenormal	false	Avoid all the collapses that should cause a topology change in the mesh
optimalplacement	true	Each collapsed vertex is placed in the position minimizing the quadric error
planarquadric	false	Add additional simplification constraints that improves the quality of the simplification of the planar portion of the mesh
planarweight	0.001	How much we should try to preserve the triangles in the planar regions
qualityweight	false	Use the Per-Vertex quality as a weighting factor for the simplification
autoclean	true	After the simplification, an additional set of steps is performed to clean the mesh (unreferenced vertices, bad faces, etc).
selected	false	Dimplify only selected faces.

Table 1: Table of settings used for the quadric mesh simplification by vertex decimation

4 Materials and Methods

4.1 Experiment

Variable	Type	Range	Tested Values
Eccentricity	Independent continuous	0-55	10,30,50
Ambient Strength	Independent continuous	0-1	0.4, 0.7
Specular Strength	Independent continuous	0-1	0, 0.3
Camera Distance	Independent continuous	$\pm\infty$	75, 150
Level of Detail (LOD)	Independent continuous	0-100%	20%, 5%, 1%, 0.1%
Light position	Independent nominal	(X,Y,Z): $\pm\infty$	X: 100,-100, Y:100,Z:0
Correct Response	Dependent nominal	0,1	-

Table 2: Table of independent and dependent variables

Stimuli We used two models from the Stanford 3D scanning repository; namely the Stanford Bunny (Fig. 4) and Dragon model (Fig. 5). As these are raw 3D scans, preprocessing of the models was required to render them without artefacts. The two models in the experiment were rendered with a neutral grey background using a perspective projection camera. The perspective camera was created using the display size parameters, resulting in a vertical field of view of approximately 37 degrees.

Task The participant was seated in a headrest 50cm away from the display (Fig. 6) and was presented with a two-interval forced choice (2IFC), as described by Kingdom and Prins [19]. In each trial, the participant was shown two intervals containing the same model, with one interval containing the full level of detail and another interval containing a reduced level of

Figure 4: Stanford Bunny model

Source:
https://graphics.stanford.edu/data/3Dscanrep/bunny-scanalyze-sh.jpg

Figure 5: Stanford Dragon model

Source:
https://graphics.stanford.edu/data/3Dscanrep/dragon.jpg

detail, with all other conditions being the same between intervals. This was repeated for a variety of lighting conditions across two camera distances and three eccentricities, 10°, 30°, 50°, as shown in Fig. 7. These eccentricities were calculated as an angle from the eye position of the participant to the position on the display, as shown in Fig. 6. For each interval, a square fixation point was rendered in the centre of the screen, and a grey screen was used to reset the participant's vision between intervals. After being shown both intervals, the participant was prompted to select which interval they believed contained the full quality model by pressing a left or right key on a small smart keypad. After all variables for the first model were tested, the model was switched. To avoid bias, the order of the full-quality and reduced-quality models was randomised in addition to the order of trials.

Additionally, the choice of which model to use for the first set of trials was made using a coin flip. To reduce fatigue, the participant was asked whether they felt tired at 15 minutes and was given an additional 5-minute break between the two models. For each model, 192 trials were conducted, in total 384 for both models. On average, the experiment took 50 minutes for each participant.

Hardware The experiment was performed using a Samsung Odyssey G9 49" Curved Display (LC49G94T) with a resolution of 5120x1440 and refresh rate of 239.76 Hz with a peak luminance of 160 cd/m^2. Additionally, to take input from the participant, we used a Vaydeer 9-key smart keypad (JP11011) with the centre left and centre right keys remapped to the selection of interval 1 and interval 2, respectively. Additionally, SR Research head support was used to ensure consistent distance from the display.

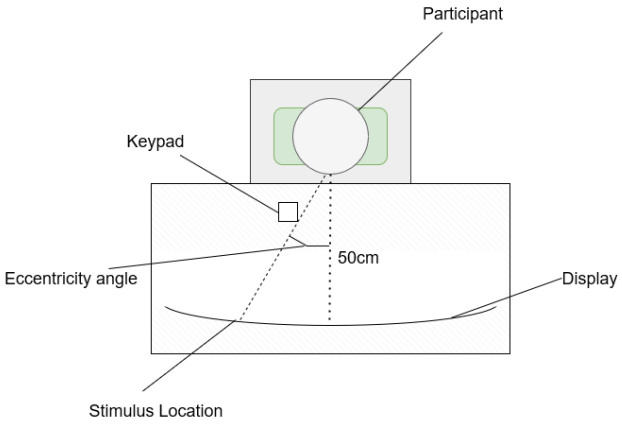

Figure 6: Setup of the experiment showing the relative locations of the participant to the display, keypad and stimulus

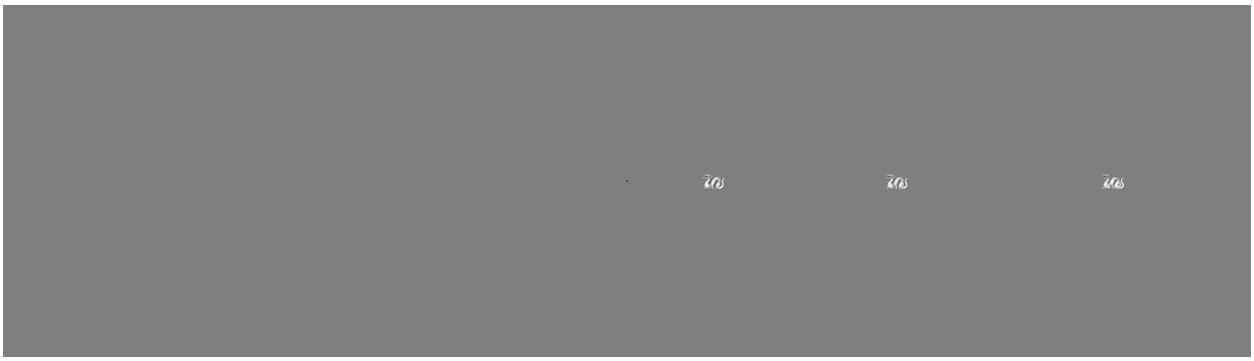

Figure 7: Dragon model displayed at eccentricities 10, 30 and 50

Participants 15 participants. All participants had normal or corrected-to-normal vision.

4.2 Software design

4.2.1 Technology Stack

A **Python** library implementation of the functions used by the **Meshlab** open-source mesh processing tool called **pymeshlab** [25] was used to generate the models at different levels of detail. These models were then added to the data file of our **Express.js** server [12] to serve the model files dynamically. This backend server is necessary due to the large size of the model files and improves the performance over static loading in react. To implement the interface, we used **react JS** [24], which contained most of the program logic for the interface and the communication between the model server and our webGL instance. The webGL state was managed using the functional web-gl Library **react-regl** [1], which is a compatibility layer for **regl** [28]. Additionally, for analysis, libraries from **Scikit** [3] and

imblearn[2] were used for their logistic regression and SMOTE functions, respectively.

4.2.2 Key Components

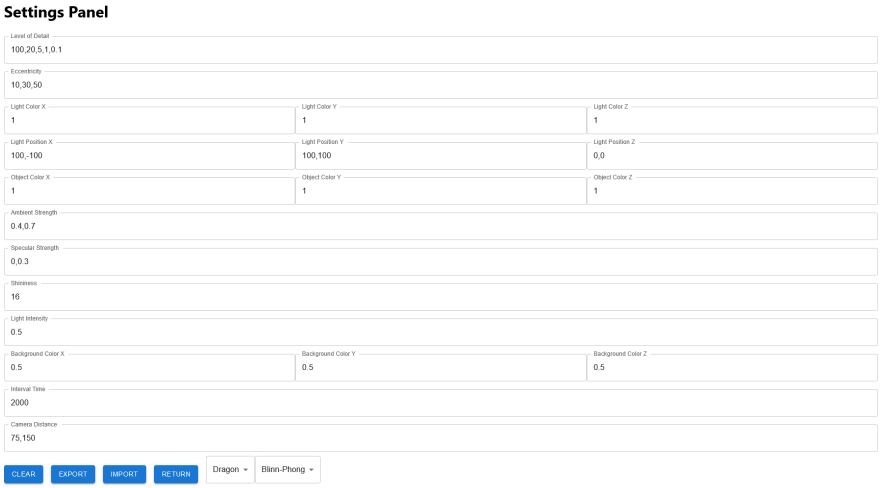

Figure 8: Screenshot of the experiment settings interface

To configure the experiment, a user interface was built in React, storing the experiment settings in a shared context across the React interface. This can be seen in Fig.8, where settings can be saved to a JSON file for import and export. The configured settings were then used to render the models in each interval, each of which is displayed similarly to Fig.9

Figure 9: Render of the dragon model at 30°

4.2.3 Randomisation

An optimised version of the Fisher-Yaytes shuffling algorithm, the Durstenfeld shuffle, was used to shuffle the order of lighting conditions for the experiments. This algorithm was chosen because it produced an unbiased permutation.

```
function shuffleArray(array) {
  for (let i = array.length - 1; i > 0; i--) {
      const j = Math.floor(Math.random() * (i + 1));
      [array[i], array[j]] = [array[j], array[i]];
  }
  return array;
}
```

4.2.4 Calculating Eccentricity

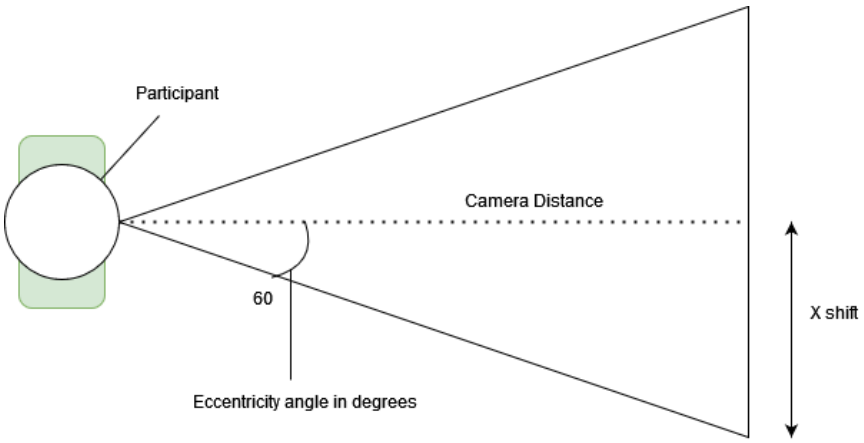

Figure 10: Calculation of eccentricity conversion factor

To get a view-based eccentricity, we set the experiment up with the user 50cm from the display in a headrest, as shown in Fig. 6. We have a curved monitor with a 1000R curve (a circle with a radius of 1000mm). As such, we needed to map the screen coordinates to the eccentricity angle of the participant, the x-shift, as demonstrated by Fig. 10, we calculated a constant through the equation:

$$\tan\theta = \arctan\left(\frac{x_{\text{shift}}}{\text{camera distance}}\right)$$

We found that at 60°, the model must be moved 815 units to align with a camera distance of 1000. As such, we created our conversion factor α as:

$$\alpha = \arctan\left(\frac{815}{1000}\right) \cdot \frac{3}{\pi}$$

Multiplying α with an eccentricity β (in radians) gives us the angle needed for the 3D virtual space. From which the virtual x-shift is calculated as:

$$x_{\text{shift}} = \tan(\alpha \cdot \beta) \cdot \text{camera distance}$$

5 Analysis

To analyse the effects of the variables on whether the participant could identify which model was of higher quality, we computed several comparison metrics against the correct response. The correct response, a dichotomous variable, is obtained as the result of each trial, where the value true represents the correct identification of the interval that contained the higher quality model and false the selection of the lower detail model.

5.1 Correlation methods

The **SciPy** implementation of the point biserial correlation by Joseph Lev[20] was used between each discrete variable and whether the participant correctly identified the higher quality model, correct response. The point biserial correlation is a particular case of the Pearson correlation coefficient. The two correlation coefficients are mathematically equivalent, where the output is between 1 and -1. A Pearson correlation coefficient measures the linear correlation between two data continuous sets, whereas a biserial correlation measures the linear correlation between a continuous and a dichotomous variable. A correlation coefficient, r_{pb} of 1, would indicate a perfect linear correlation where the increase in one variable can be shown as a linear increase in the other variable. This is inversely true for a -1 coefficient. Values approaching 0 imply no correlation.

To calculate the point biserial coefficient, we divide the data set into Group 1, Y_T, which contains True values, and Group 2, Y_F, which contains False values for the correct response. In these two groups, we calculate the mean value $\bar{Y}_T$ for the continuous variable in Group 1 and the mean value $\bar{Y}_F$ for the continuous variable in Group 2. We then define N_T and N_F as the number of observations coded True and False. The distribution of these two can be seen in Fig. 12. Finally, we define N as the total number of observations and s_y as the standard deviation of all the observations in Y_T and Y_F. Utilising these variables, we can then calculate the point biserial coefficient, r_{pb}:

$$r_{pb} = \frac{\bar{Y}_T - \bar{Y}_F}{s_y}\sqrt{\frac{N_T N_F}{N(N-1)}}$$

	correct response 0	**correct response 1**
light position X -100	698	2374
light position X 100	709	2363

Table 3: Cross tabulation of correct response and the X component of light position

We used fixed positions for the light position, with the variation being two values for the X component. As such, we classified this as a dichotomous variable. A Chi-squared independence test is more appropriate than a biserial correlation coefficient for two dichotomous variables as it is designed for categorical data instead of continuous variables. The Chi-squared value represents the difference between the expected and observed frequencies under

the assumption of the null hypothesis; the two variables are independent. The greater the Chi-squared value, the stronger the difference between the expected and observed values, indicating a relationship between the two variables. We used the Chi-squared value, χ^2, and the p-value, the probability of obtaining test results at least as extreme as the ones observed. These were used to determine the significance of the effects on the trial outcome. The two variables were cross-tabulated in a contingency table to calculate these values, as shown in Tab. 3. We then calculated the χ^2 test-statistic using the formula:

$$\chi^2 = \sum_{i=1}^{n} \frac{(O_i - E_i)^2}{E_i}$$

Where χ^2 is the cumulative Chi-squared test statistic, O_i is the number of observations of type i, in our case i is correct response 0 or correct response 1, E_i is the expected count of type i, and n is the total number of cells in the table (4). Under the null hypothesis, the expected frequency for each cell in the table is calculated, assuming the variables are independent. The expected frequency (E_{ij}) for each cell is determined by:

$$E_{i,j} = \frac{(\text{row total for cell}(i,j)) \cdot (\text{column total for cell}(i,j))}{N}$$

To determine the p-value, the χ^2 value needs to be compared to the Chi-square distribution with an appropriate level of degrees of freedom, the number of independent values or quantities that can vary in the analysis without breaking any constraints. The degree of freedom, df is defined as:

$$df = (\text{number of rows} - 1) \cdot (\text{number of columns} - 1)$$

With 2 rows and 2 columns in Tab. 3, our degrees of freedom is 1. Using **SciPy**'s chi2_contingency function, we can look at the appropriate Chi-squared distribution table and find the p-value of 0.761. In our tests for statistical significance, the null hypothesis H_0 states between the two variables, the independent variable and dependent correct response, as mentioned in Tab. 2, are independent of each other and the distribution of one variable does not affect the distribution of the other. Our alternative hypothesis H_1 states the two variables are not independent and contain an association between them. We took our threshold p-value for rejecting the null hypothesis at 0.05, where the p-value is the probability of obtaining test results at least as extreme as the ones observed under the assumption that the null hypothesis is true.

The correlation of variables in Fig. 11 shows that the most significant factor for participants selecting the lower quality model incorrectly is eccentricity, with a higher eccentricity correlating with a lower chance of choosing the interval containing the higher quality model, a relationship described as inversely proportional. Additionally, the chart shows us that ambient and specular strength had little effect on the trial outcome as their values fall within the expected range of observations for independent variables, demonstrated by their p-values in Tab. 4 being greater than our 0.05 threshold. Only LOD, camera distance and eccentricity have p-values below 0.05, suggesting strong evidence to reject the null hypothesis that

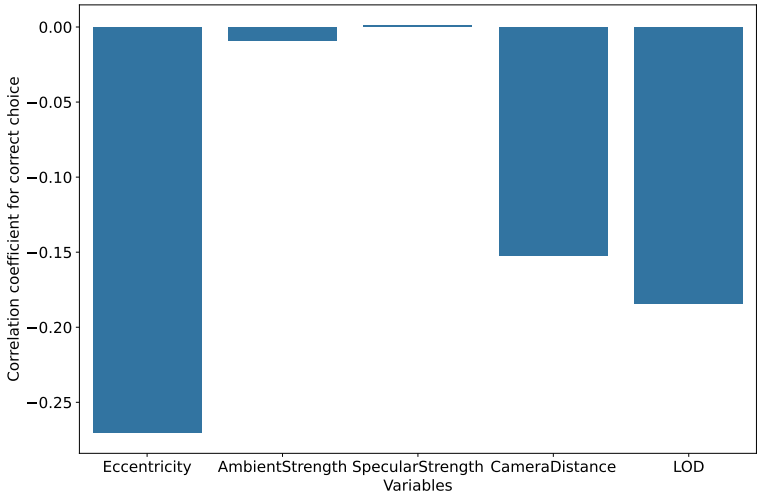

Figure 11: Point biserial correlation of variables effect on correctly identifying the higher quality model

Variable	***p*-value**
Eccentricity	2.00e-103
Ambient Strength	0.486
Specular Strength	0.927
CameraDistance	3.56e-33
LOD	5.41e-48

Table 4: Point biserial correlation coefficient of for each independent variable correct response where the p-value is the probability of obtaining test results at least as extreme as the ones observed

the two variables are independent. Our χ^2 value for light position and correct response was 0.0921, suggesting little difference between our expected values and observed values under the null hypothesis; the p-value of 0.761 confirms the fact that the observations are insufficient to reject the null and the effects of light position on correct response is insignificant.

5.2 Model development

5.2.1 Logistic regression

Logistic regression is a machine-learning tool used as a binary classifier. In our case, it will decide whether the participant can determine the difference between a full and a reduced-quality model under eccentricity, camera distance, and lighting conditions. From our analysis of the independence of variables in the previous section 5.1, we can eliminate some features

from the model before training. Eccentricity, camera distance, and LOD were the only values with a p-value against correct response below the threshold; we can eliminate other variables from the model. In this model, we assume trials are independent of one another, and there is no relation between observations.

5.2.2 Model Standardisation

To standardise the logistic regression model across various 3D models, we created a composite variable for the screen space occupied by the model called 'vertices per pixel'. The vertices per pixel were calculated as a combination of the number of vertices derived from the level of detail model, the camera distance and the bounding box area on the screen. First, the model vertices were transformed using the view (V) and projection (P) matrices of the camera using the formula:

$$\mathbf{v}_{clip} = \mathbf{P} \cdot \mathbf{V} \cdot \mathbf{v}$$

where $\mathbf{v}$ is a vertex in homogeneous coordinates $[x, y, z, 1]$, $\mathbf{V}$ is the view matrix, and $\mathbf{P}$ is the projection matrix. From this, the vertices were mapped to their pixel coordinates on the screen, referred to as normalised device coordinates (ndc):

$$\mathbf{v}_{ndc} = \left[\frac{v_{clip,x}}{v_{clip,w}}, \frac{v_{clip,y}}{v_{clip,w}}, \frac{v_{clip,z}}{v_{clip,w}} \right]$$

where $\mathbf{v}_{clip} = [v_{clip,x}, v_{clip,y}, v_{clip,z}, v_{clip,w}]$.

$$\mathbf{v}_{screen} = \left[\left(\frac{v_{ndc,x} + 1}{2} \right) \cdot W_{viewport}, \left(\frac{v_{ndc,y} + 1}{2} \right) \cdot H_{viewport} \right]$$

Where $W_{viewport}$ and $H_{viewport}$ are the viewport width and height, respectively. A bounding box area for the model was calculated with the minimal and maximal X and Y coordinates.

$$\text{minX} = \min(v_{screen,x}), \quad \text{maxX} = \max(v_{screen,x})$$

$$\text{minY} = \min(v_{screen,y}), \quad \text{maxY} = \max(v_{screen,y})$$

Using the bounding box, the screen space area was calculated.

$$A_{screen} = (\text{maxX} - \text{minX}) \cdot (\text{maxY} - \text{minY})$$

The final vertices per pixel are computed by dividing the number of vertices by the screen space area.

$$\text{Vertices per Pixel} = \frac{N_{vertices}}{A_{screen}}$$

Where $N_{vertices}$ is the number of vertices.

After producing the two feature vectors, the vertices per pixel and eccentricity, predict whether the change would be identified relative to the original model. We used an 80/20 training and test data split to fit the model. We applied a standard scaler provided by **sklearn**, which standardises the features by removing the mean and scaling the features

to unit variance. This is done by applying the fit_transform function to the training data with **parameters**:*Copy= True, with_mean= True, with_std= True.* The function works by computing the mean (μ) and standard deviation (σ) for each feature and applying the following transformation:

$$z = \frac{(x - \mu)}{\sigma}$$

Where x is the training sample from the dataset, producing the new set of feature vectors z. This scaler is then used to transform the training data similarly.

5.2.3 Addressing Class imbalance

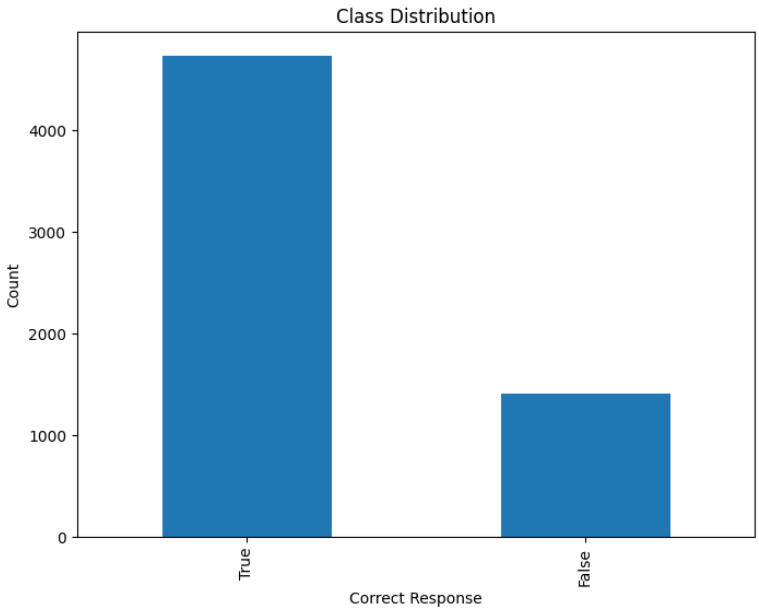

Figure 12: Distribution of correct identifications of the lower level of detail model

The participants very often selected the interval containing the higher quality model, resulting in a class imbalance, as demonstrated in Tab. 12. To address the class imbalance, we used the python **imblearn** implementation of the Synthetic Minority Oversampling Technique (SMOTE) algorithm developed by N.V. Chawla et al. [9] to increase the representation of responses where the lower-quality model was incorrectly classified as the full-quality model. This adjustment is reflected in the contrasting results between the two confusion matrices depicted in Fig. 14 and Fig. 13 where the first has a strong bias to predict true at all times. By introducing SMOTE, we can balance the dataset and reduce the logistic regression bias when choosing the True label for the correct response.

The SMOTE algorithm works by first finding the k-nearest neighbours through Euclidean

distance between each pair of minority class (M) samples $\mathbf{x}_i$ and $\mathbf{x}_j$, given by:

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{m=1}^{M} (x_{im} - x_{jm})^2}$$

For each minority class sample $\mathbf{x}_i$, it randomly selects one of its k-nearest neighbors, $\mathbf{x}_{\text{nn}}$.Then the algorithm generates synthetic samples by interpolating between $\mathbf{x}_i$ and $\mathbf{x}_{\text{nn}}$:

$$\mathbf{x}_{\text{synthetic}} = \mathbf{x}_i + \delta \cdot (\mathbf{x}_{\text{nn}} - \mathbf{x}_i)$$

Where δ is a random number between 0 and 1, i.e., $\delta \sim \text{Uniform}(0, 1)$. When using the **imblearn** library, we used their random seed to generate the number with a seed of 42.

Accuracy	0.669
Precision	0.879
Recall	0.655
ROC-AUC	0.724
F-Score	0.750

Table 5: Accuracy statistics for the logistic model with SMOTE

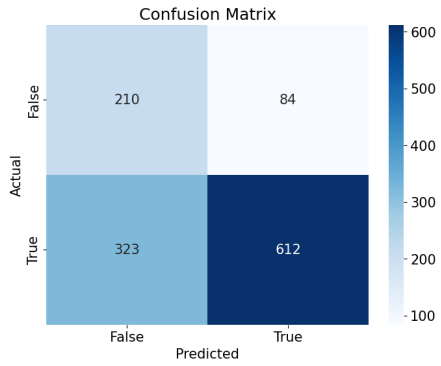

Figure 13: Confusion matrix for the logistic regression model with SMOTE applied

The two tables of descriptive statistics for the model with SMOTE in Tab. 5 and without SMOTE in Tab. 6 show that despite the lower F-Score of 0.75 for the second confusion matrix compared to the 0.85 of the first; we argue that the model utilising SMOTE is a better model due to a higher precision.

5.2.4 Developing the model

Parameters *solver= lbfgs, multi_class=ovr, penalty= l2.,C = 1.0, max_iter= 100, dual= False, tol= 0.0001, fit_intercept= True, random_state= 42*
With our feature vectors in a balanced sampling, we can utilise the **scikit** implementation of logistic regression. Logistic regression describes the probability of correct response being true for a single trial, modelled using a logistic function. The logistic function is a sigmoid curve with the equation:

$$f(x) = \frac{1}{1 + e^x}$$

where x is a linear combination of input features, $\beta_0 + \beta_1 x_1 + \beta_2 x_2 + ...$ with β_0. being the intercept and β 1-n being coefficients and the x values being input features. In training our

Accuracy	0.754
Precision	0.768
Recall	0.971
ROC-AUC	0.724
F-Score	0.857

Table 6: Accuracy statistics for the logistic model without SMOTE

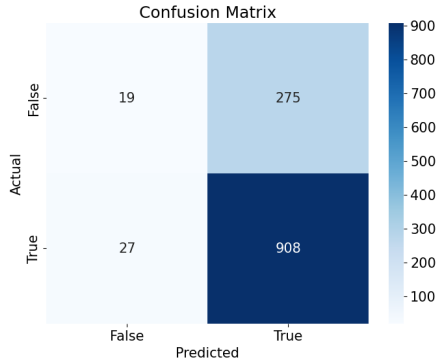

Figure 14: Confusion matrix for the logistic regression model without SMOTE applied

model, we fit the curve to our datapoints using a gradient descent optimisation algorithm, Limited-memory BGFS [22]. The algorithm starts with an initial estimate of the optimal value of each input parameter and interatively refine the estimate to find optimal coefficients. We will not go into how this algorithm works for this project's scope. After fitting the model, we can make predictions.

5.2.5 Model predictions

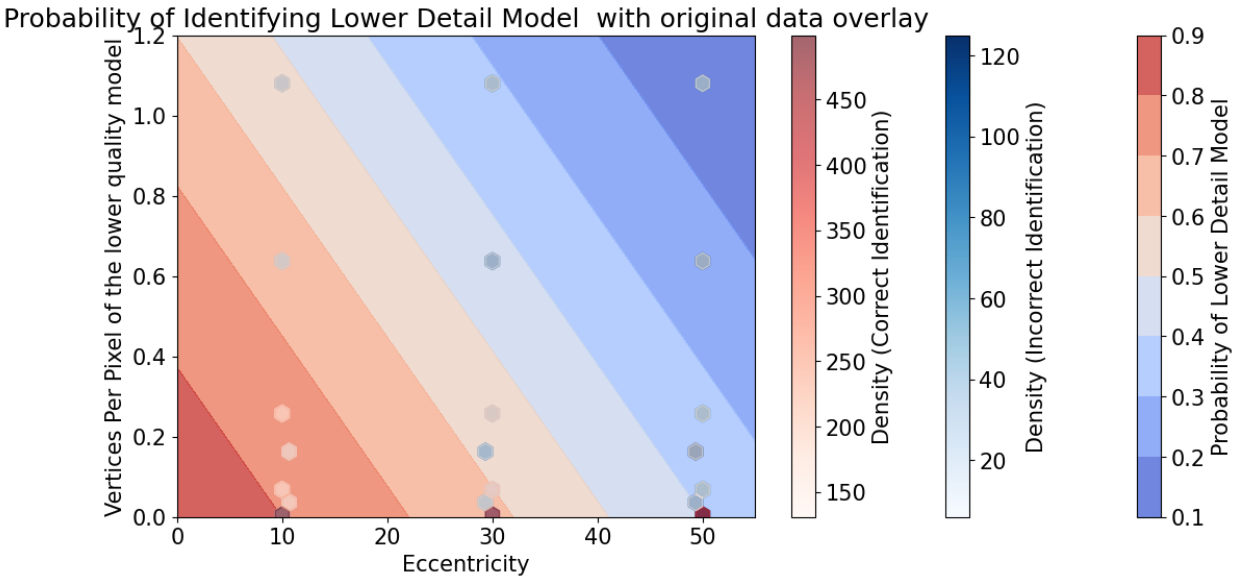

Figure 15: Contour matrix for the 300 predictions with the hex-bin density of the original data correct identification overlaid

Utilising this model, we predicted 300 evenly spaced predictions over the eccentricity range from 0 to 55 and vertices per pixel range of 0 to 1.2, creating a contour plot as shown in Fig. 15. The plot of predicted results shows that the lower the eccentricity from the centre of focus and the lower the quality of the model, the more noticeable it is and the more likely

the participant is to identify which model is higher quality. The overlaid original data, two sets of hexagons ranging from deep red to deep blue, representing the density of correct and incorrect identifications, respectively, broadly agrees with the results of our prediction model. However, when the vertices per pixel are very low at extreme levels, they are much more identifiable in our original data than our model would suggest. A more complicated, non-linear model may be able to have a more accurate prediction for these extreme points however we leave this out of the scope of this project.

6 Results and Discussion

6.1 Overview

Our original research questions were as follows:

- How do specular highlights and ambient strength affect the visually tolerable amount of LOD mesh reduction across different eccentricities on the screen?
- How does light angle affect the visually tolerable amount of LOD mesh reduction across different eccentricities on the screen?
- How does eccentricity affect the visually tolerable LOD?

Regarding our first research question, "How do specular highlights and ambient strength affect the visually tolerable amount of LOD mesh reduction across different eccentricities on the screen?" we conducted experiments testing these variables across two values each for three levels of eccentricity as outlined in Section 4. Subsequently, we analyzed these factors as detailed in Section 5. These results lead us to the following insights:

6.1.1 Specular Highlights

We used a biserial correlation coefficient to explore the association between specular strength and participants' ability to distinguish models with greater detail. The analysis yielded a p-value of 0.927, indicating insufficient evidence to reject the null hypothesis. This suggests that variations in specular strength minimally impact participants' ability to perceive changes in mesh detail across 10°, 30°, and 50° eccentricities.

6.1.2 Ambient Strength

Similarly, our analysis of ambient strength resulted in a p-value of 0.486, again suggesting our data had no significant effect on participants' ability to discern differences in mesh detail across varying eccentricities.

6.1.3 Light Angle

Addressing the question of how light angle affects the visually tolerable amount of LOD mesh reduction, we tested light positions at the model's top left and top right. Statistical analysis (chi-squared test) yielded a chi-squared value of 0.0921 and a p-value of 0.761, indicating no significant influence on participants' perception of mesh detail loss.

6.1.4 Eccentricity

Our final research question, "How does eccentricity affect the visually tolerable LOD?" confirmed that peripheral eccentricity significantly impacts the perception of LOD detail, highlighting the need for optimized LOD strategies based on viewer eccentricity.

In conclusion, our study provides insights into factors affecting visually tolerable LOD mesh reduction:

1. Specular highlights and ambient strength have minimal impact on participants' ability to discern mesh detail, irrespective of eccentricity.
2. Light angle (top left vs. top right) does not significantly affect the perception of LOD mesh reduction, simplifying rendering processes.
3. Eccentricity strongly influences LOD perception, emphasizing the importance of adaptive LOD strategies.

6.2 Implications

Our results confirm the characteristics of the human visual system, with eccentricity being a major factor in the perception of detail. We also revealed the lesser importance of exploring ambient strength, specular strength, and light position, allowing future researchers to focus more on other areas.

6.3 Data Exploration

Figure 16: Demonstration of the training effect before fixing the randomization algorithm.

While initial results suggested a training effect, as shown by the strong trendline in Fig. 16, this was due to issues with our previous randomization algorithm. After implementing the Durstenfeld shuffle 4.2.2, subsequent results demonstrated that the training effect was insignificant, as illustrated by the improved trendline in Fig. 17.

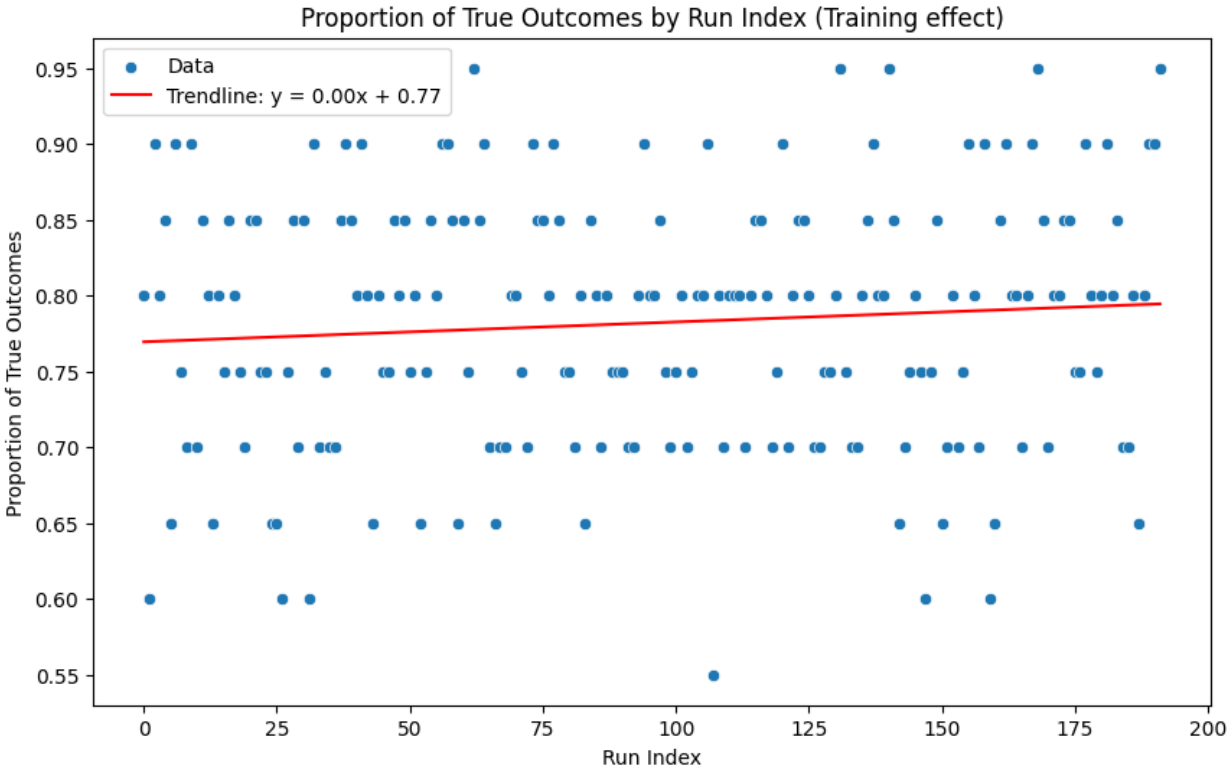

Figure 17: Training effect was shown to be insignificant after fixing the randomization algorithm.

7 Conclusion

This project aimed to investigate the relationships between various lighting conditions and the perception of the level of detail. We have created an extendable framework for experiments involving 3D rendering and peripheral vision on widescreen displays to achieve these goals. Utilising this framework, we created an experiment with 15 participants to study the effects of lighting conditions on the perception of level of detail. Throughout this paper, we have shown through a point biserial correlation and chi-squared independence test that the effects of our tested values for ambient light, specular strength and light position have an insignificant impact on the perception of a reduced level of detail. However, we did confirm the effects of eccentricity on reducing the perception of level of detail. Using our collected data, we created a logistic regression model to predict the visibility of the reduction of vertex density in a lower-detail model relative to eccentricity.

8 Future Work

Our experiment is limited in its scope. As such, there is plenty of room for further expansion. We have now developed a framework that is capable of rendering 3D models and collecting data for them; however, it does not include several features that could be used:

- **Further parameters**: Our model has poor performance in the extremes; more specific studies into the extremes of our parameters may reveal additional information that could be used to further our understanding.
- **Textures**: Textured models and would need future expansion to do so. Implementing textures may mask LOD reduction or increase the visibility of artefacts.
- **Additional LOD algorithms**: In our experiment, we only used one level of detail reduction algorithm; expanding this to different LOD algorithms, such as progressive meshes [17], may reveal results that our experiment missed.
- **Silhoutte**: The level of detail reduction is more visible on the silhouette's edges; our model does not account for this.
- **Further refinement of model**: In this paper, we only used one model from our data; a more accurate model may be able to be produced from the data given finer tuning of parameters
- **Testing smaller quality change**: In our experiment, we always used one interval with a full-quality model and a reduced-quality model. The results may be more balanced, and a better model would be produced with a smaller change between the two models.

9 Acknowledgments

I thank Prof. Cara Tursun for her help throughout this project. Her knowledge and assistance have been precious to me. Additionally, I would like to thank Prof. Jiri Kosina for his supervision and ideas in our meetings.

References

[1] GitHub - kevzettler/react-regl: React Fiber Reconciler Renderer for Regl WebGL — github.com. https://github.com/kevzettler/react-regl. [Accessed 04-06-2024].

[2] imbalanced-learn documentation 2014; Version 0.12.3 — imbalanced-learn.org. https://imbalanced-learn.org/stable/. [Accessed 05-07-2024].

[3] scikit-learn: machine learning in Python 2014; scikit-learn 1.5.1 documentation — scikit-learn.org. https://scikit-learn.org/stable/index.html. [Accessed 05-07-2024].

[4] The Stanford 3D Scanning Repository — graphics.stanford.edu. https://graphics.stanford.edu/data/3Dscanrep/. [Accessed 04-07-2024].

[5] Jase A, Nnanna Kama, Robert Thompson, Dave Ruddell, and Robin Bradley. The next generation of VR gaming on ps5, Feb 2023.

[6] apple Inc. Apple Vision Pro, https://www.apple.com/apple-vision-pro/. Accessed: 2024-02-21.

[7] D. Brodsky and B. Watson. Model simplification for interactive applications. In *Proceedings IEEE Virtual Reality 2000 (Cat. No.00CB37048)*, page 286, 2000.

[8] J. Brookes, M. Warburton, and M. et al. Alghadier. Studying human behavior with virtual reality: The unity experiment framework. *Behavior Research*, 52:455–463, 2020.

[9] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. Smote: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, June 2002.

[10] James H. Clark. Hierarchical geometric models for visible-surface algorithms. *SIGGRAPH Comput. Graph.*, 10(2):267, jul 1976.

[11] Xiaoying Ding, Zhao Chen, Weisi Lin, and Zhenzhong Chen. Towards 3d colored mesh saliency: Database and benchmarks. *IEEE Transactions on Multimedia*, 26:3580–3591, 2024.

[12] OpenJS foundation. Expressjs, http://expressjs.com/. Accessed: 2024-06-15.

[13] 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, page 247–254, New York, NY, USA, 1993. Association for Computing Machinery.

[14] Michael Garland and Paul S Heckbert. Fast polygonal approximation of terrains and height fields. 1995.

[15] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[16] Thorsten Hansen, Lars Pracejus, and Karl R. Gegenfurtner. Color perception in the intermediate periphery of the visual field. *Journal of Vision*, 9(4):26–26, 04 2009.

[17] Hugues Hoppe. Progressive meshes. In *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '96, page 99–108, New York, NY, USA, 1996. Association for Computing Machinery.

[18] Hugues Hoppe. *Progressive Meshes*. Association for Computing Machinery, New York, NY, USA, 1 edition, 2023.

[19] Frederick A.A Kingdom and Nicolaas Prins. *Psychophysics: A practical introduction*. Academic Press, 2nd edition, 2016.

[20] Joseph Lev. The Point Biserial Coefficient of Correlation. *The Annals of Mathematical Statistics*, 20(1):125 – 126, 1949.

[21] Peter Lindstrom and Greg Turk. Image-driven simplification. *ACM Transactions on Graphics (ToG)*, 19(3):204–241, 2000.

[22] Dong C. Liu and Jorge Nocedal. On the limited memory bfgs method for large scale optimization. *Mathematical Programming*, 45(1):503–528, Aug 1989.

[23] Meta. Meta quest, https://www.meta.com/nl/en/quest/. Accessed: 2024-03-30.

[24] Meta. React JS: The library for web and native user interfaces, 2024.

[25] Alessandro Muntoni and Paolo Cignoni. PyMeshLab, January 2021.

[26] Hunter A Murphy and Andrew T Duchowski. Gaze-contingent level of detail rendering. In *Eurographics (short presentations)*, 2001.

[27] A. Oliva, A. Torralba, M.S. Castelhano, and J.M. Henderson. Top-down control of visual attention in object detection. In *Proceedings 2003 International Conference on Image Processing (Cat. No.03CH37429)*, volume 1, pages I–253, 2003.

[28] Regl project time. Regl-project/regl: functional webgl, https://github.com/regl-project/regl, 2024. Accessed: 2024-02-21.

[29] M. Reddy. Scrooge:perceptually-driven polygon reduction. *Computer Graphics Forum*, 15(4):191–203, 1996.

[30] Jarek Rossignac and Paul Borrel. Multi-resolution 3d approximations for rendering complex scenes. In *Modeling in computer graphics: methods and applications*, pages 455–465. Springer, 1993.

[31] William J Schroeder, Jonathan A Zarge, and William E Lorensen. Decimation of triangle meshes. In *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 65–70, 1992.

[32] Sony. Playstation VR | live the game with the PS VR headset | Playstation, https://www.playstation.com/nl-nl/ps-vr/. Accessed: 2024-02-21.

[33] Emma E. M. Stewart, Matteo Valsecchi, and Alexander C. Schütz. A review of interactions between peripheral and foveal vision. *Journal of Vision*, 20(12):2–2, 11 2020.

[34] Luca Surace, Cara Tursun, Ufuk Celikcan, and Piotr Didyk. Gaze-Contingent Perceptual Level of Detail Prediction. In Tobias Ritschel and Andrea Weidlich, editors, *Eurographics Symposium on Rendering*. The Eurographics Association, 2023.

[35] PsychoPy Team. Psychopy, https://pypi.org/project/psychopy/, 2023. Accessed: 2024-02-21.

[36] Psychtoolbox Team. Psychtoolbox-3, http://psychtoolbox.org/, 2023. Accessed: 2024-02-21.

[37] Unity Technologies. Nanite virtualized geometry in Unreal engine, https://docs.unrealengine.com/5.0/en-us/nanite-virtualized-geometry-in-unreal-engine/, 2024. Accessed: 2024-02-21.

[38] Unity Technologies. Unity - manual: Level of detail LOD for meshes, https://docs.unity3d.com/manual/levelofdetail.html, 2024. Accessed: 2024-02-21.

[39] Okan Tarhan Tursun, Elena Arabadzhiyska-Koleva, Marek Wernikowski, Radosław Mantiuk, Hans-Peter Seidel, Karol Myszkowski, and Piotr Didyk. Luminance-contrast-aware foveated rendering. *ACM Trans. Graph.*, 38(4), jul 2019.

[40] A. van Dam. VR as a forcing function: Software implications of a new paradigm. In *Proceedings of 1993 IEEE Research Properties in Virtual Reality Symposium*, pages 5–8, 1993.

[41] Benjamin Watson, Neff Walker, and Larry F. Hodges. Managing level of detail through head-tracked peripheral degradation: a model and resulting design principles. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, VRST '97, page 59–63, New York, NY, USA, 1997. Association for Computing Machinery.