CS4102 3D Rendering 160026526 29 April 2021

• Overview:
The aim of the practical was to learn and work with techniques commonly used in the rendering of 3D objects.

• Design:
Basis of application
The application will be implemented using Processing which allows for the creation of PVectors that stores xyz-coordinates of each point for creating the triangles which will be used as a mesh for the face projection. The PVectors will also be used to store the colour values. The program runs in Fullscreen with a recommended screen resolution of 1920x1080. Clicking on the triangle produces the synthetic face based by calculating the ratios between each point, which correspond to a different face mesh, and applying the calculations to the average face. The ratios are also displayed on the screen. An executable was created for Windows and Linux for the program.

Reading the data
All the data is stored in csv files which can be read in processing using the loadTable function. For the base specification, only the mesh, sh0-3, tx0-3, shEV and txEV csv files were loaded. The data can be read by getting the floats at the specific coordinates.

Triangle to manipulate the face
A equilateral triangle was drawn onto the screen and a mouse left click function was implemented to save the position in the triangle along with drawing a point in the triangle. This will later be used when calculating the ratios to for the offsets for face variations. When implementing the left click functionality, it was required for the clicks to only apply when in the triangle so a triangle collision detection was implemented using the formula from Jeffrey Thompsons collision detection site which was commented in the code.

Calculating user offset for face
To calculate the user offset, the distance of the point the user has chosen from each of the three triangle points was calculated and if the distance of any of the points is 0, this would mean the ratio for that point was 1 while the other two were 0. If this was not the case, the ratio would be calculated by,

$$\text{sum} = 1/d1 + 1/d2 + 1/d3$$
$$r1 = 1/d1/\text{sum}$$
$$r2 = 1/d2/\text{sum}$$
$$r3 = 1/d3/\text{sum}$$

where r1, r2 and r3 are the respective ratios for each of the points in the triangle which will be used for the user offset when calculating the face.

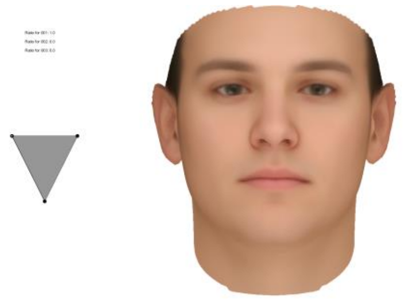
Calculating face
When calculating the face, each row in the mesh csv was iterated over and the three columns were used to locate the three points in which will be used to form triangles. The points were then located for the average face and the three face offsets. The points were then calculated by multiplying each offset face point with its weight along with the user's ratio and adding them all with the average face points. Painter's algorithm will be used later which paints the furthest object first so the three points in the triangle was sorted to allow for the location of the point with the smallest z value. Since the values of the face were large, it was scaled downwards, and the signs were swapped due to processing coordinate system starting from the top left of the screen to the bottom right instead of

the usual bottom left to top right coordinate system. The same method was used to find the colour of each triangle. An average colour was also calculated by adding each of the three colours and dividing by three to get a constant colour for the whole triangle when painting. All the triangles were then added into an ArrayList of Triangles.

<u>Triangle class</u>
The triangle class will be used when drawing the triangles to the screen to display the face. Each triangle will have three PVectors which contains the three xyz-coordinates of the triangle along with a PVector to hold its average colour and a float that will contain the furthest z-coordinate for the triangle which will be used when comparing for Painter's algorithm. A comparable was written in the triangle class which will be later used to compare and sort the z-values. Back in the main method, Collections was used to sort the list of all triangles (allTri) based on the furthest z-value of each triangle, producing a list of triangles with z-values from the furthest to the closest.

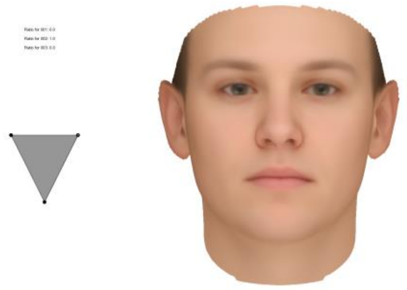
<u>Drawing the face</u>
To draw the face, the ArrayList of all triangles are iterated over and the strokes and fill were set to each triangle colour. The triangles were then drawn with the appropriate colours with flat shading using each of the points, along with offsetting the x and y values to ensure that the face was properly projected close to the centre of the screen. Each triangle was drawn only using the xy-coordinates as the z-coordinates did not matter when looking at the face from a front perspective and the z-coordinates had already been used for the sorting. The image was orthogonally projected in a front view.

<u>Screenshots of the application</u>
• Ratio of 1 for the first face



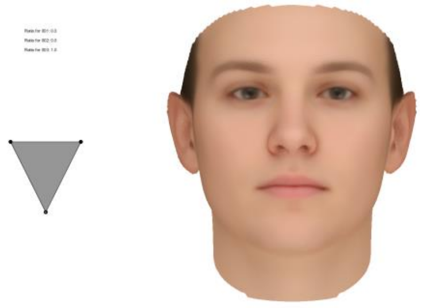
Ratio for 001: 1.0
Ratio for 002: 0.0
Ratio for 003: 0.0

• Ratio of 1 for the second face



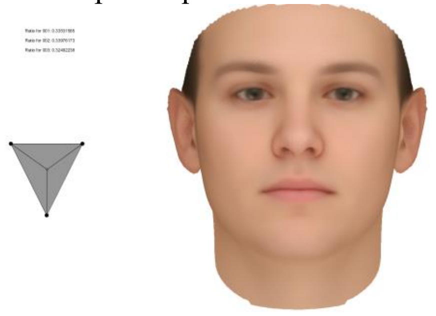
Ratio for 001: 0.0
Ratio for 002: 1.0
Ratio for 003: 0.0

• Ratio of 1 for the third face



Ratio for 001: 0.0
Ratio for 002: 0.0
Ratio for 003: 1.0

• Attempt at equal ratio for each face



```
Ratio for 001: 0.33531988
Ratio for 002: 0.33976173
Ratio for 003: 0.32482238
```

• Discussion and Evaluation:
Looking at the time space complexity for Painter's algorithm, the time complexity depends on the sorting algorithm used to order the polygons, in this case based on the z-values. The sorting used was the Java comparator which has a time complexity of O(nlog(n)). The worst-case space complexity for Painter's algorithm will be O(n+m) in which n is the number of polygons, in this case triangles and m is the number of pixels that must be filled.

When sorting the z-values, the furthest value of each triangle was taken when used to compare. This was a basic form of sorting the polygons but can be improved with a depth buffer (Z-Buffer Method) or using a depth sorting method.

• Conclusion:
In conclusion, the application was able to read the csv files, take user input as offset values, process and sort the triangles which were used to create the synthetic face and displayed using Painter's algorithm using the appropriate graphics design techniques such as flat shading and orthogonal projection. The basic specification of the project can be considered a success.