TDT4195 Exercise 1

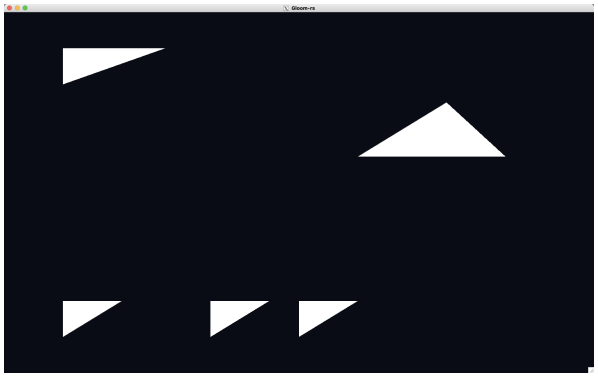
Henrik Horpedal

September 8, 2024

1 Heading

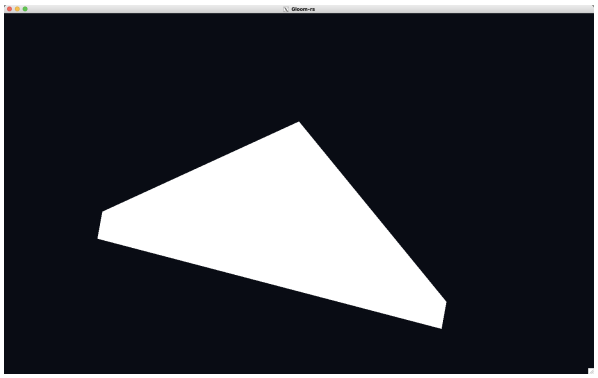
1.1 Task 1c

Five distinct triangles:



1.2 Task 2a

This was the result:



1.2.1 What is the name of this phenomenon?

What we see here is a result of a step in the 3D graphics pipeline known as clipping.

1.2.2 When does it occur?

It occurs when an object intersects with the clipping boundaries. The part of the object that sticks out from this volume is clipped out.

1.2.3 What is its purpose?

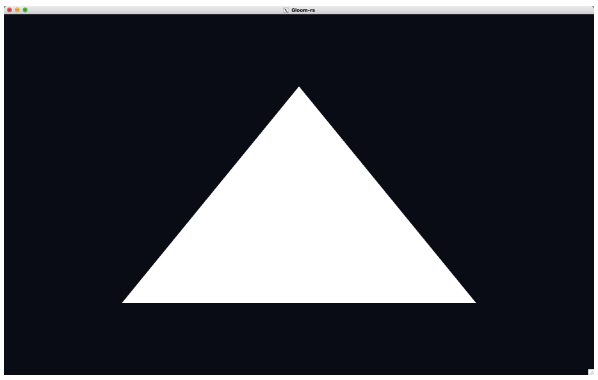
The clipping volume defines the volume of space that the camera can see. There is no point in keeping an object that is outside what the camera will see. Therefore, by removing them, it makes the scene less complex and increases performance.

1.3 Task 2b

I used the following vertices to specify the triangle in this task:

$$v_0 = \begin{bmatrix} -0.6 \\ -0.6 \\ 0 \end{bmatrix}, \quad v_1 = \begin{bmatrix} 0.6 \\ -0.6 \\ 0 \end{bmatrix}, \quad v_2 = \begin{bmatrix} 0 \\ 0.6 \\ 0 \end{bmatrix} \tag{1}$$

When the vertex indices are specified in ascending order, `[0,1,2]`, the triangle appears. It also appears with `[2,0,1]`:



But it is not displayed when using `[2,1,0]` or `[0,2,1]`:

The reason for this is known as back-face culling. Back-faces refer to polygons where the angle between its normal-vector and the view-vector is greater than 90°. The normal vector is, in this case, calculated based on the order in which the vertices of

the triangles are specified in the index buffer. Therefore, the triangles that are defined clockwise will be culled and not visible.

1.4 Task 2c

1.4.1 1

The depth buffer is a two-dimensional array with the same size as the frame-buffer. Each element is linked to each pixel in the frame and represents the distance to the nearest primitive at that pixel. While loading the frame, it is continuously updated if a primitive is closer. It is used to determine what surfaces are hidden by other primitives. If a circle were to move leftward without clearing the depth buffer, I would imagine objects behind the circle would be visible, even though they are supposed to be covered up.

1.4.2 2

If there are multiple objects in front of each other, the depth buffer is updated multiple times as fragments are processed. The fragment shader can be executed several times for the same pixel.

1.4.3 3

The two most common types of shaders are vertex shaders and fragment shaders. The vertex shader operates on the individual vertices of a polygon and typically performs geometric transformations like transforming from 3D space to screen space. The fragment shader is responsible for assigning the color to a fragment and assigning a depth value.

1.4.4 4

Because you can re-use the vertices. Imagine specifying a pyramid—you only need to specify the top vertex once and not create a new vertex with the same coordinates each time. By doing this, you save the number of times the vertex shader is called, which leads to better performance.

1.4.5 5

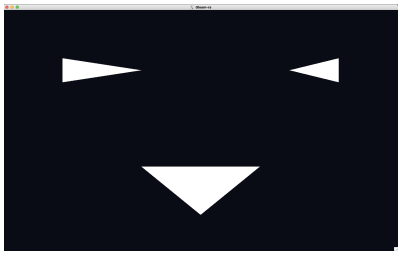
If we want to not only store 3D positions but also texture coordinates with an additional two components. If we, for some reason, only wanted the texture, we could define an offset that skips over the 3D coordinates. The offset would be the size of the datatype the 3D coordinates are defined with, multiplied by three.

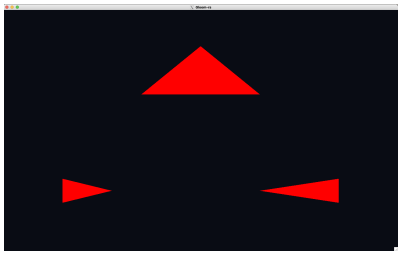
1.5 Task 2d

The triangles before coloring and transformations:

After:

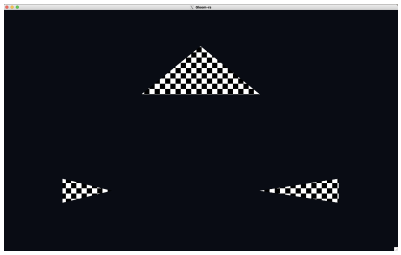
To change the color to red, I just used `1.0f, 0.0f, 0.0f, 1.0f` in the fragment shader. To flip the scene horizontally and vertically, I multiplied the x-coordinate and y-coordinate by -1.





1.6 Task 3

I used `gl_FragCoord` combined with modulo to decide whether the pixel should be black or white.



NTNU