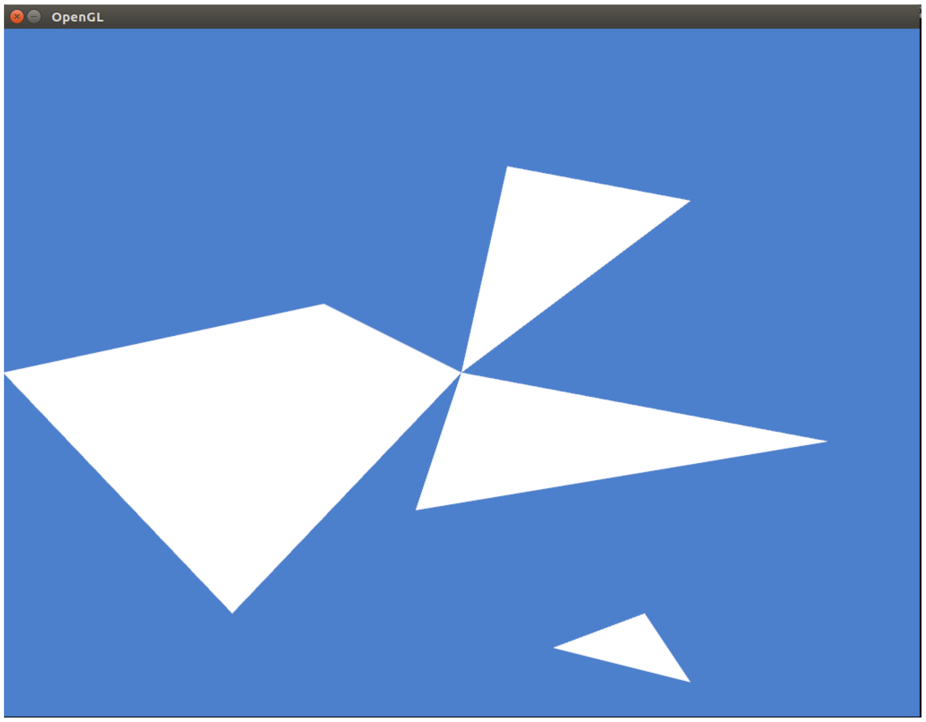
TDT4195 OpenGL Lab 1

Rendell Cale

Task 1

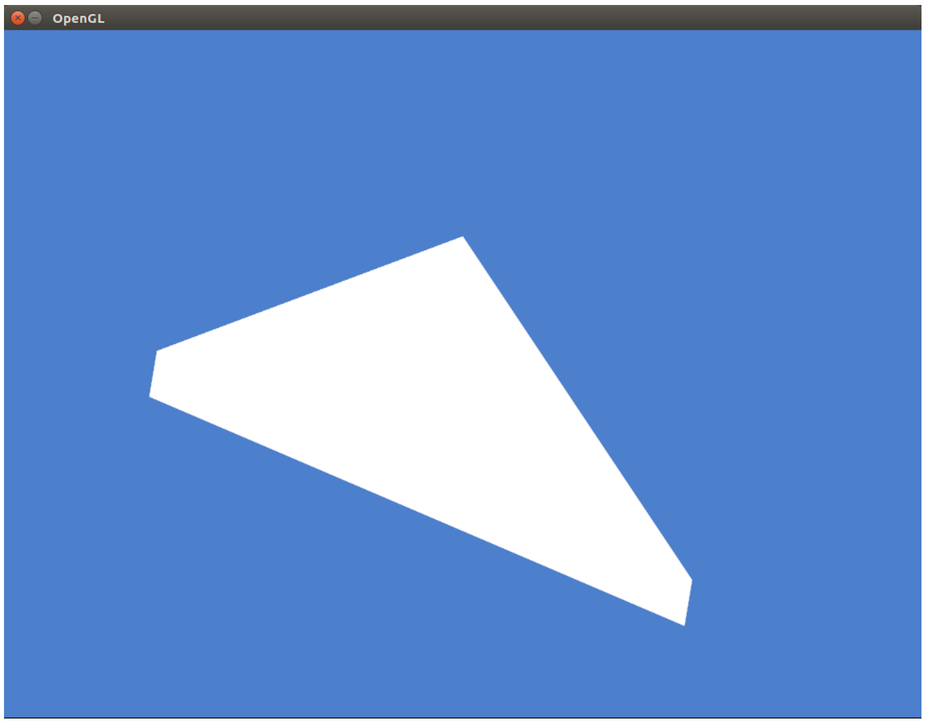
c



Task 2

a

i



Clipping. Two of the vertices are outside the 2x2x2 box centered at the origin which means they wont be rendered. This is quite apparent when it happens in the z-dimension. ### ii Clipping occours when we render something outside the clipping region. ### iii The purpose of clipping specifically in the z-direction, is to eliminate objects that are too far away, too close, or behind the camera.

b

i

After changing the order of the vertices, the triangle disappears! ### ii It happens because the triangle is facing the other direction. ### iii The normal of the surface of the triangle is given by the right hand rule, so when the vertices are listed counter-clock wise, then normal points toward the "camera". But when we flip two vertices we are drawing it clock wise and thus we see the backside of the triangle, which wont render.

c

i

The depth buffer provides a way to render objects in 3D dimensions, where some objects might be partially covered or even transparent.

ii

A shader object is a compiled shader.

A program object is a collections of shader objects which can be executed on the machine.

iii

The two most commonly used shader types are vertex and fragment shaders.

The *vertex shader* does geometric transformations to position the object in the scene and emulate thing like camera placement.

The *fragment shader* applies colors and textures to the rasterized image.

iv

The vertex attributes can be used as inputs to the vertex shader, specyfying how the vertices should be tranformed and interpreted. The index buffer specify how the vertices should be connected together and is independent of how the vertices are transformed.

v

Shader input variables are what the CPU can use to change what the vertex shader does.

Vertex attributes are used on the CPU side to transfer data to the GPU.

d

i

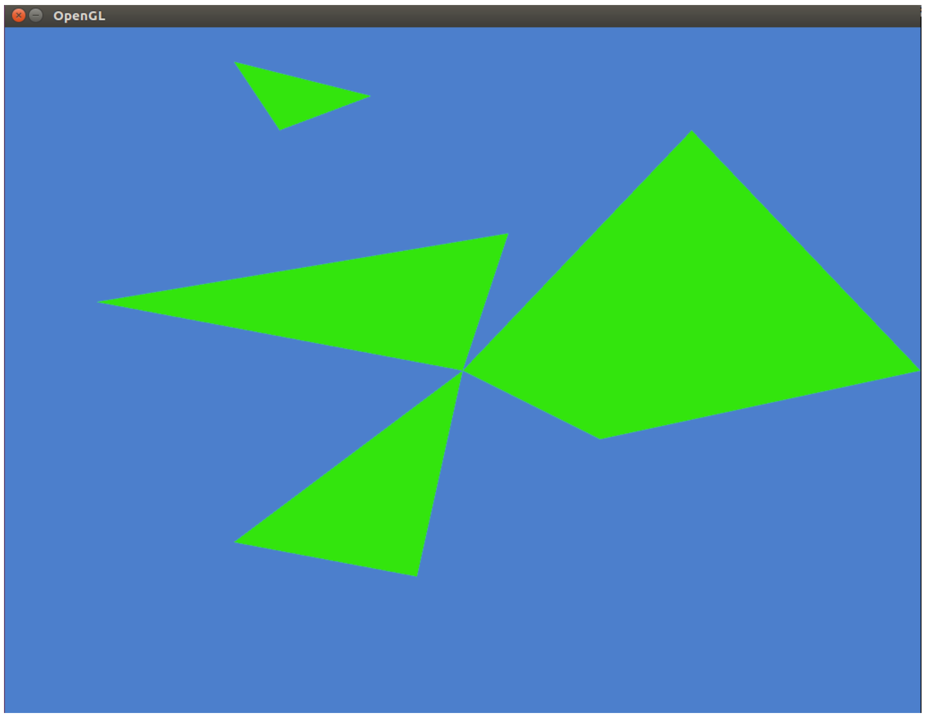
Flipped the x and y axis in the vertex shader with:

```
gl_Position = vec4(-position.xy, position.z, 1.0f);
```

ii

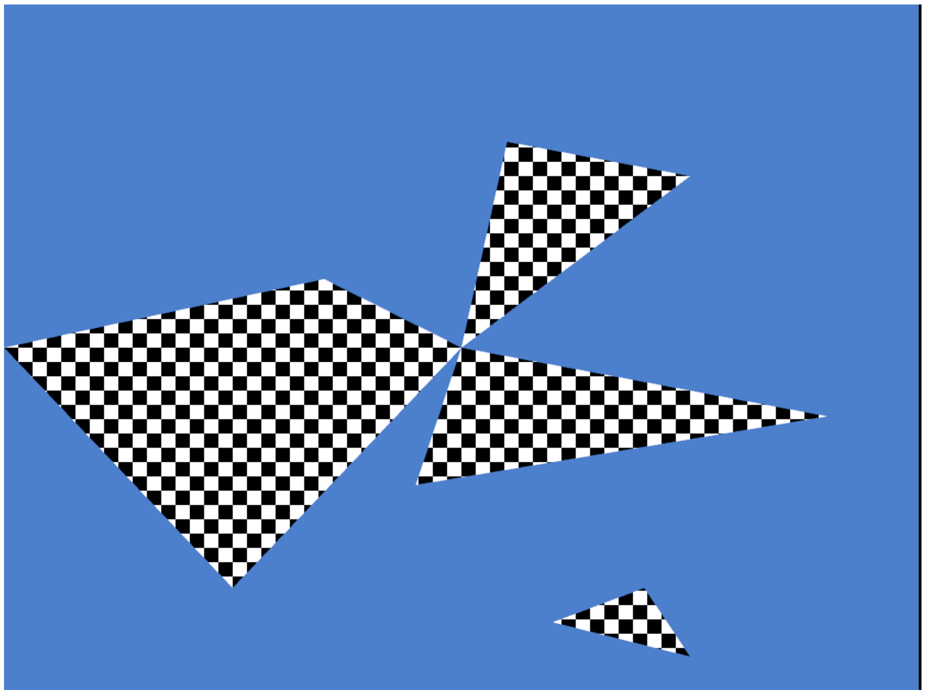
Made the colour of the object green in the fragment shader with:

```
color = vec4(0.2f, 0.9f, 0.05f, 1.0f);
```



Task 3

a

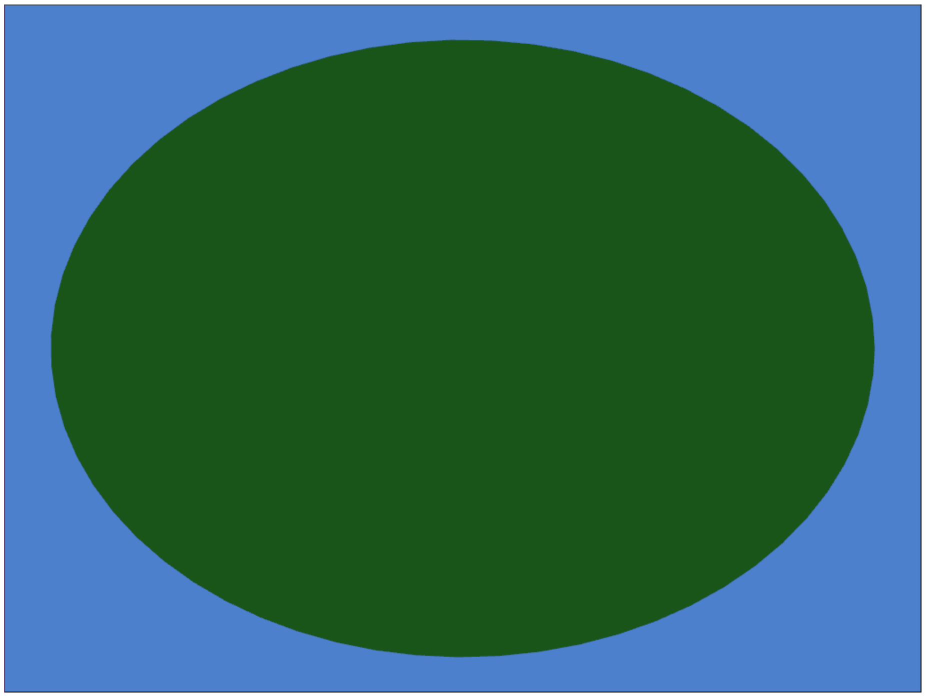


Relevant fragment shader code:

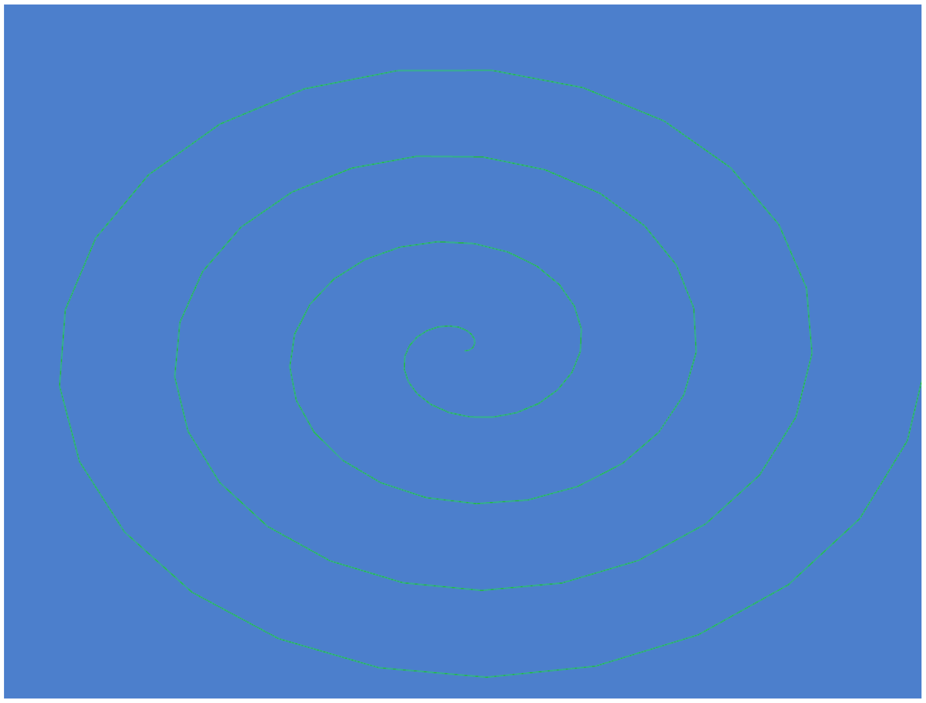
```
float b = float((int(gl_FragCoord.x) % 32 < 16 && int(gl_FragCoord.y)
% 32 < 16) || (int(gl_FragCoord.x) % 32 >= 16 && int(gl_FragCoord.y)
% 32 >= 16));
```

```
color = vec4(b, b, b, 1.0);
```

b



c



d

This is done for both b and c, but is a bit hard to demonstrate in a pdf.

e