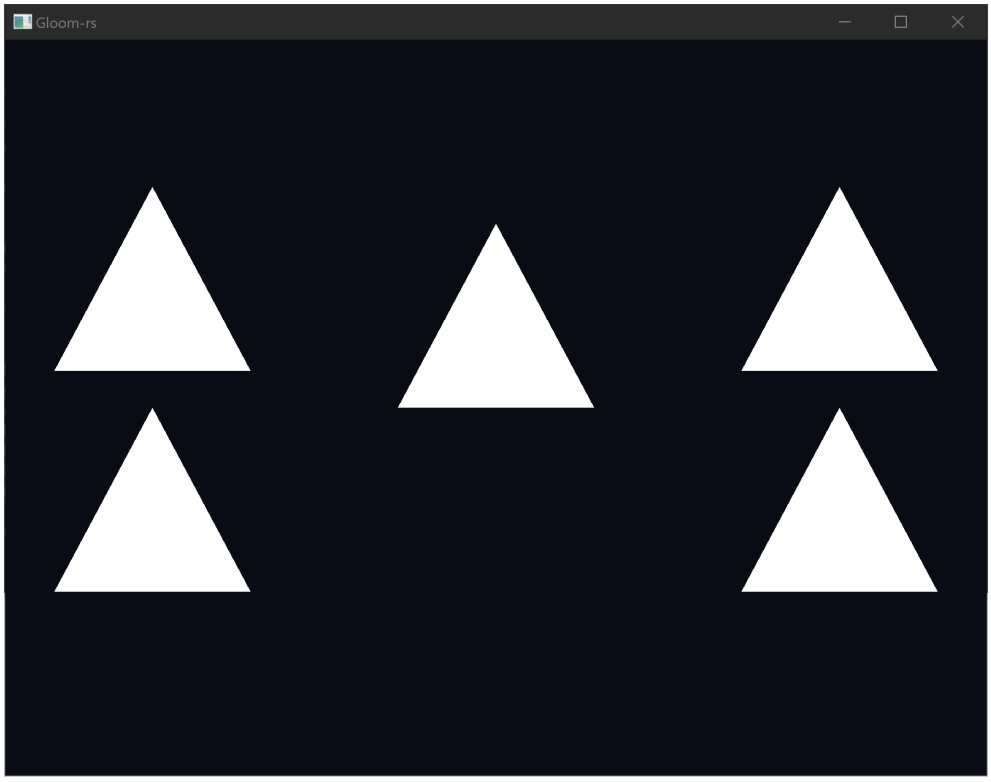
TDT4195 Exercise 1

Martin Skatvedt

September 7, 2022

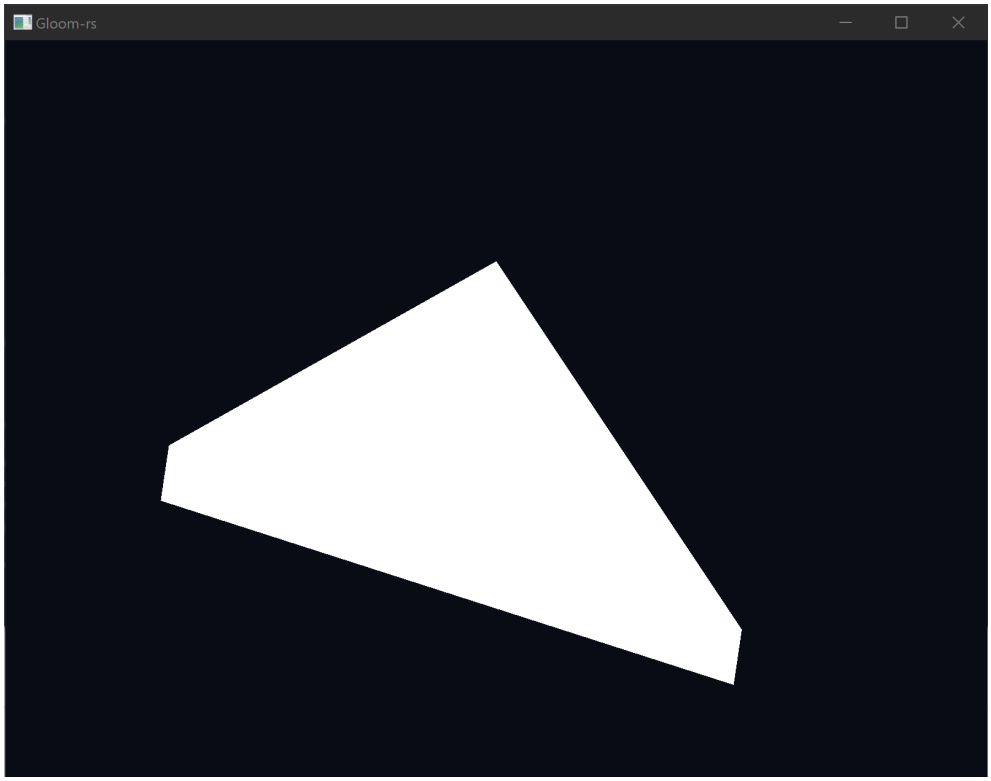
1 Task 1

1.1 Subtask 1C



2 Task 2

2.1 Subtask 2A



Q: What is the name of this phenomenon?

A: This phenomomenon is called clipping.

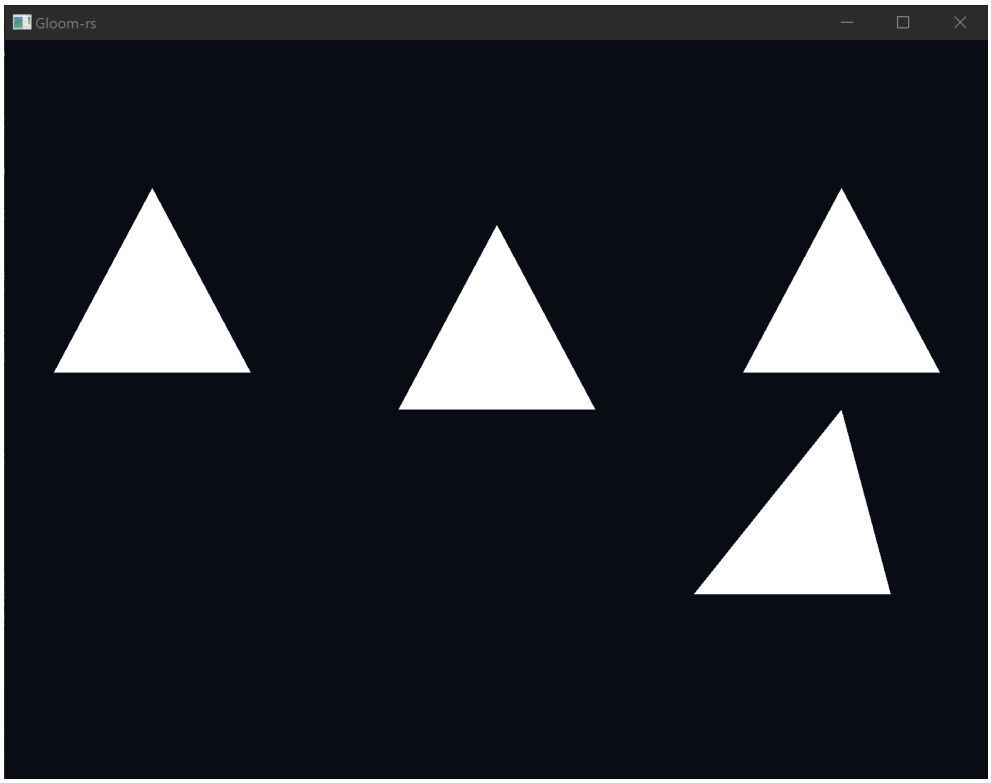
Q: When does it occur?

A: It occurs whenever we try to render something that does not fit onto the screen. As in this example the triangle has z coordinates, which makes it rotate into the z plane, and makes the edges disapear.

Q: What is its purpose?

A: Its purpose is improving performance. By not drawing sections which are not displayed we can save memory.

2.2 Subtask 2B



Q: What happens?

A: The triangle (down left) disapears when swapping the indices of a single triangle.

Q: Why does it happen?

A: This happens because the triangle gets drawn clockwise. This makes the triangles frontside be on the other side, and it does not get drawn.

Q: What is the condition under which this effect occurs? Determine a rule

A: This effect is called *back face culling*. And it occurs when a shapes backside is shown inside of the front. This makes the rendering process faster, due to a smaller number of vertecies has to be drawn. The rule is, when a triangle is drawn counter clockwise its gets drawn, however if it is drawn clockwise it will not be drawn.

2.3 Subtask 2C

Q: Why does the depth buffer need to be reset each frame?

A: The depth buffer holds information about the depth of each pixel in the scene. This means that each pixel is compared to the depth buffer. But if you clear the scene, and dont clear the depth buffer. Each pixel will be compared to the depth of pixels in another scene.(https://stackoverflow.com/questions/19469194/why-do-we-have-to-clear-depth-buffer-in-opengl-during-rendering)

Q: In which situation can the Fragment Shader be executed multiple times for the same pixel? (Assume we do not use multisampling.)

A: Fragments can end up behind or infront of each other. Since fragment shaders are run once for every single fragment, a fragment shaders can be run multiple times on the same pixel. (Blokland, 2018)

Q: What are the two most commonly used types of Shaders? What are the responsibilities of each of them?

A: The two most common shaders are vertex shaders and fragment shaders. Vertex shaders runs once each time a single vertex is drawn. In the vertex shader we can transform the vertex, such as translating, scaling and rotating. Fragment shaders are run once for every single fragment. Fragment shaders are responsible for determining the color of each fragment.(Blokland, 2018)

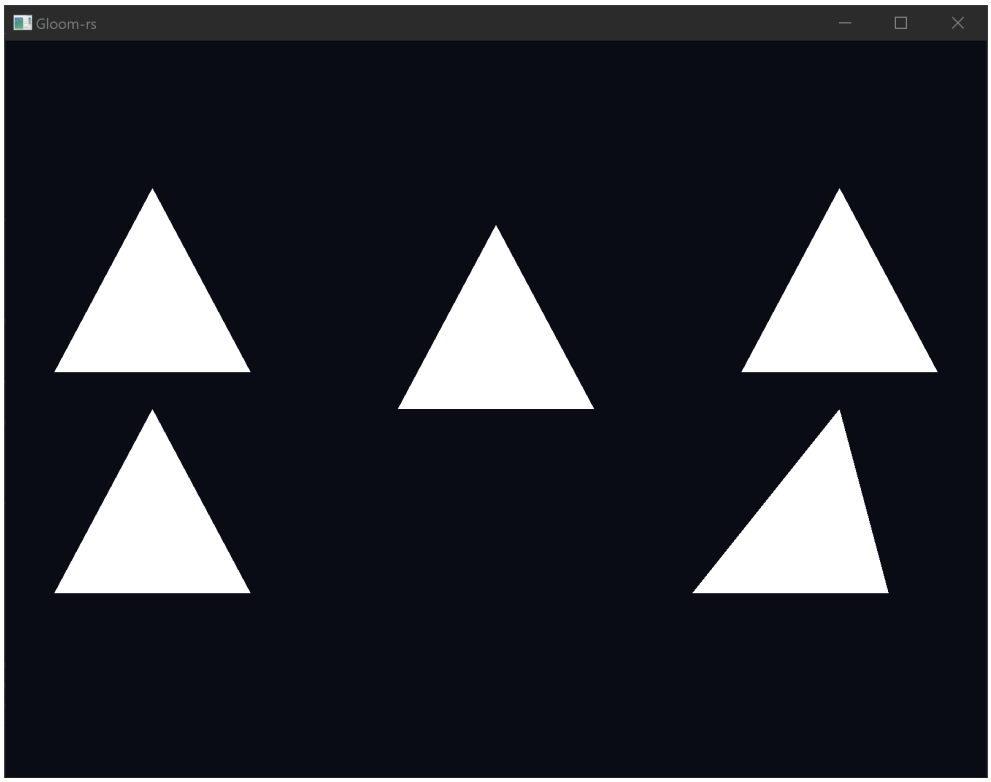
Q: Why is it common to use an index buffer to specify which vertices should be connected into triangles, as opposed to relying on the order in which the vertices are specified in the vertex buffer(s)?

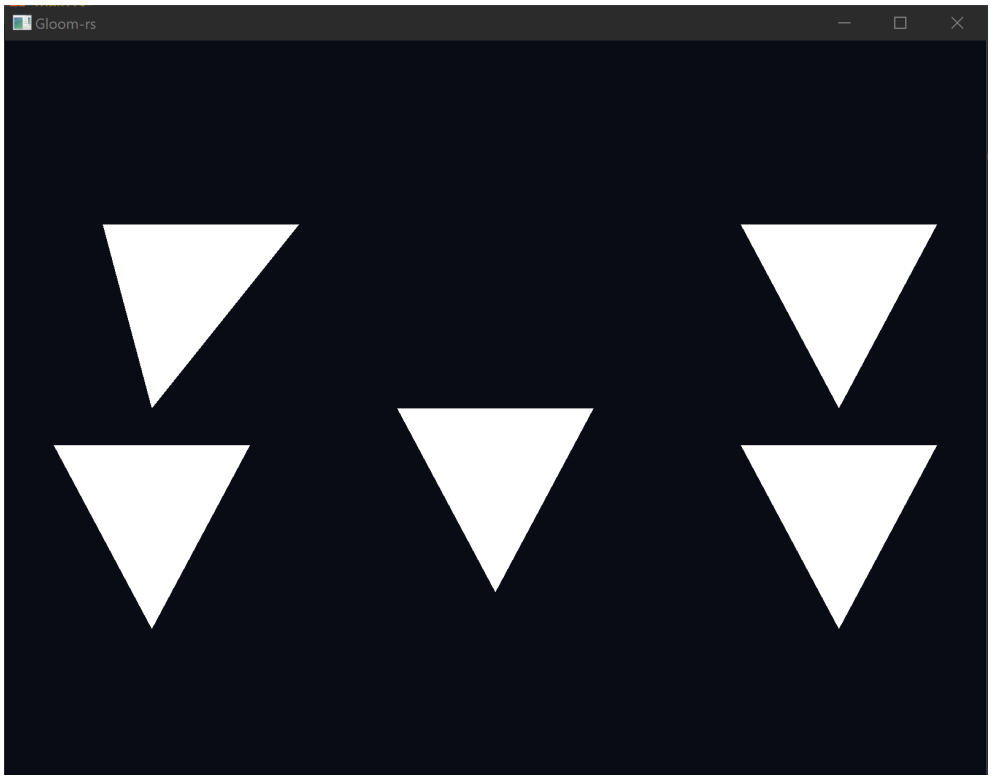
A: Often shapes are connected to the same point, so to avoid drawing the same vertex multiple times, we can use its index. This could also same alot of memory.

Q: While the last input of gl::VertexAttribPointer() is a pointer, we usually pass in a null pointer. Describe a situation in which you would pass a non-zero value into this function.

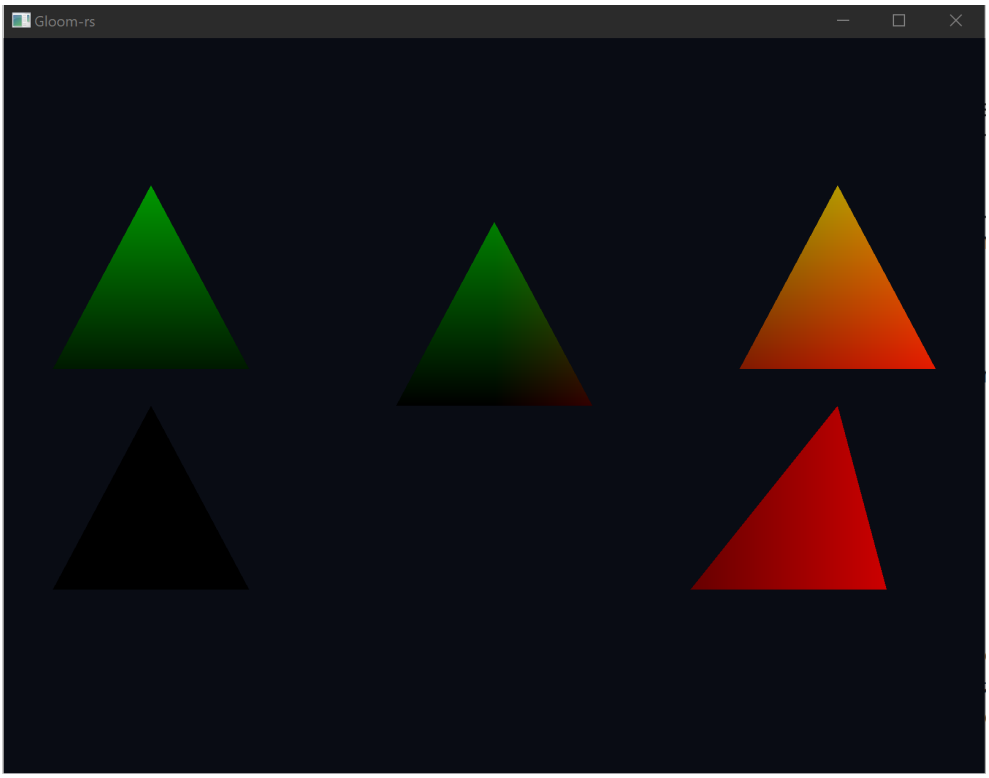
A: If your input contains different types of information, such as color or texture data you can have several Vertex Attribute Pointers. For example one for position, one for color and one for texture. The pointer should therefore point to the first occurance of each type of data.

2.4 Subtask 2D





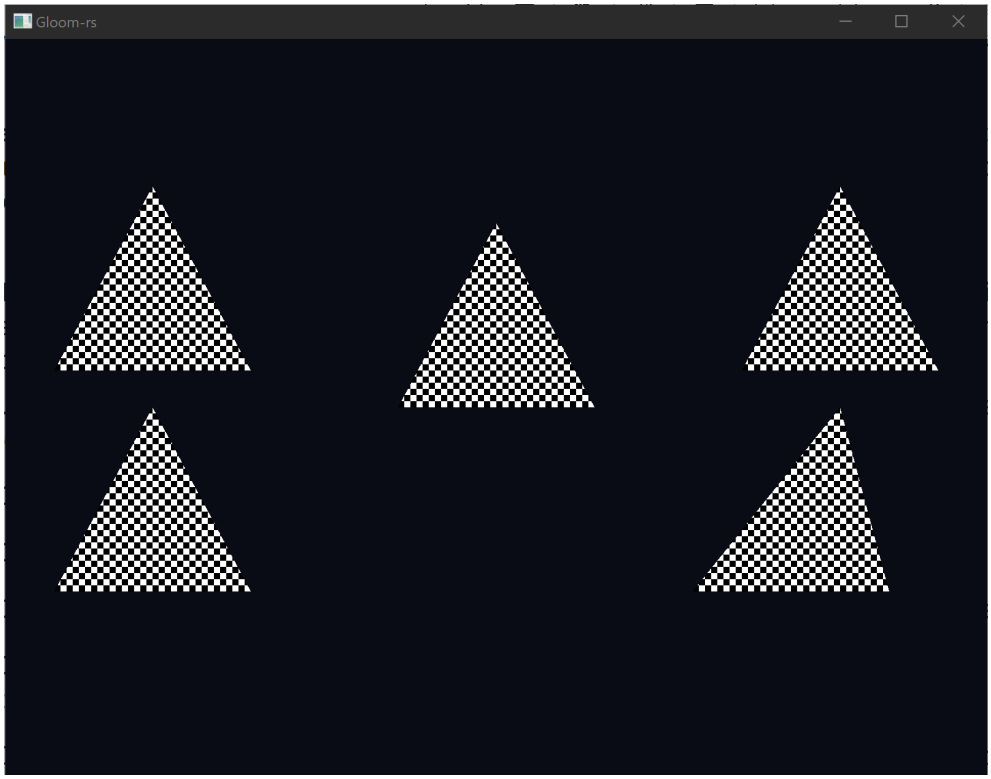
To flip the triangles vertically and horizontally I changed the vertex shader. I used a transformation on the original position. I multiplied the original position by a new vector *vec3(-1, -1, 1)* such that the x and y cordinates where flipped.

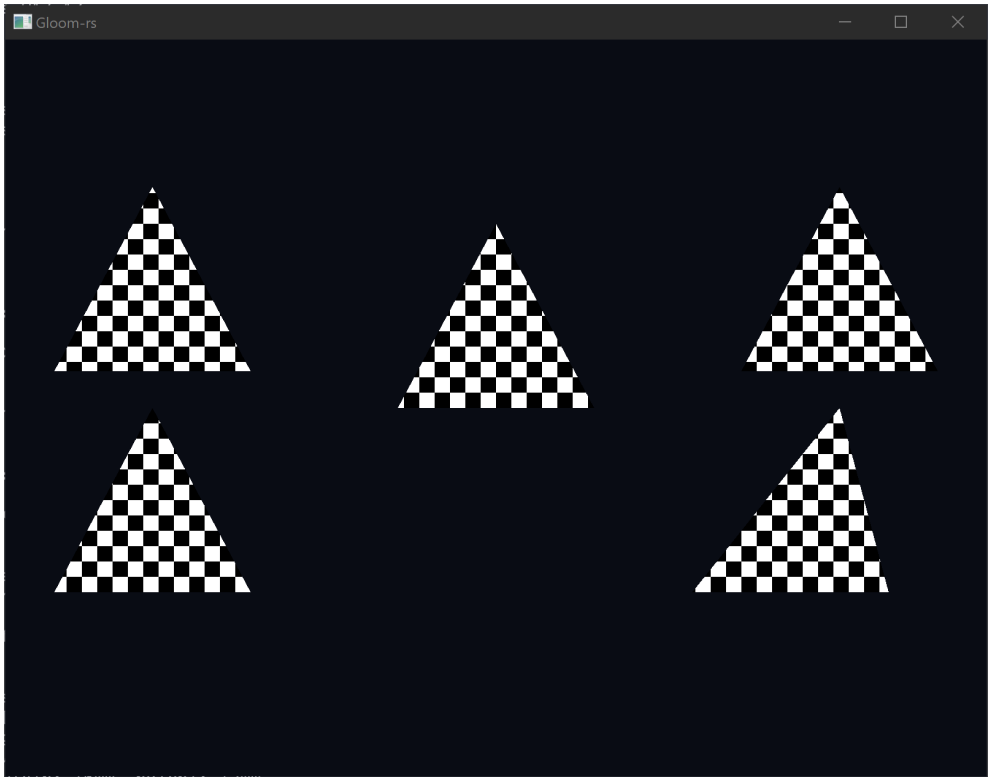


To change the color of the fragments I used the position of each vertex to change the color. I forwarded the position from the vertex shader into the fragment shader, and put that as the color. This way I achieved a almost rainbow effect.

3 Task 3

3.1 Subtask 3A




Gloom-rs