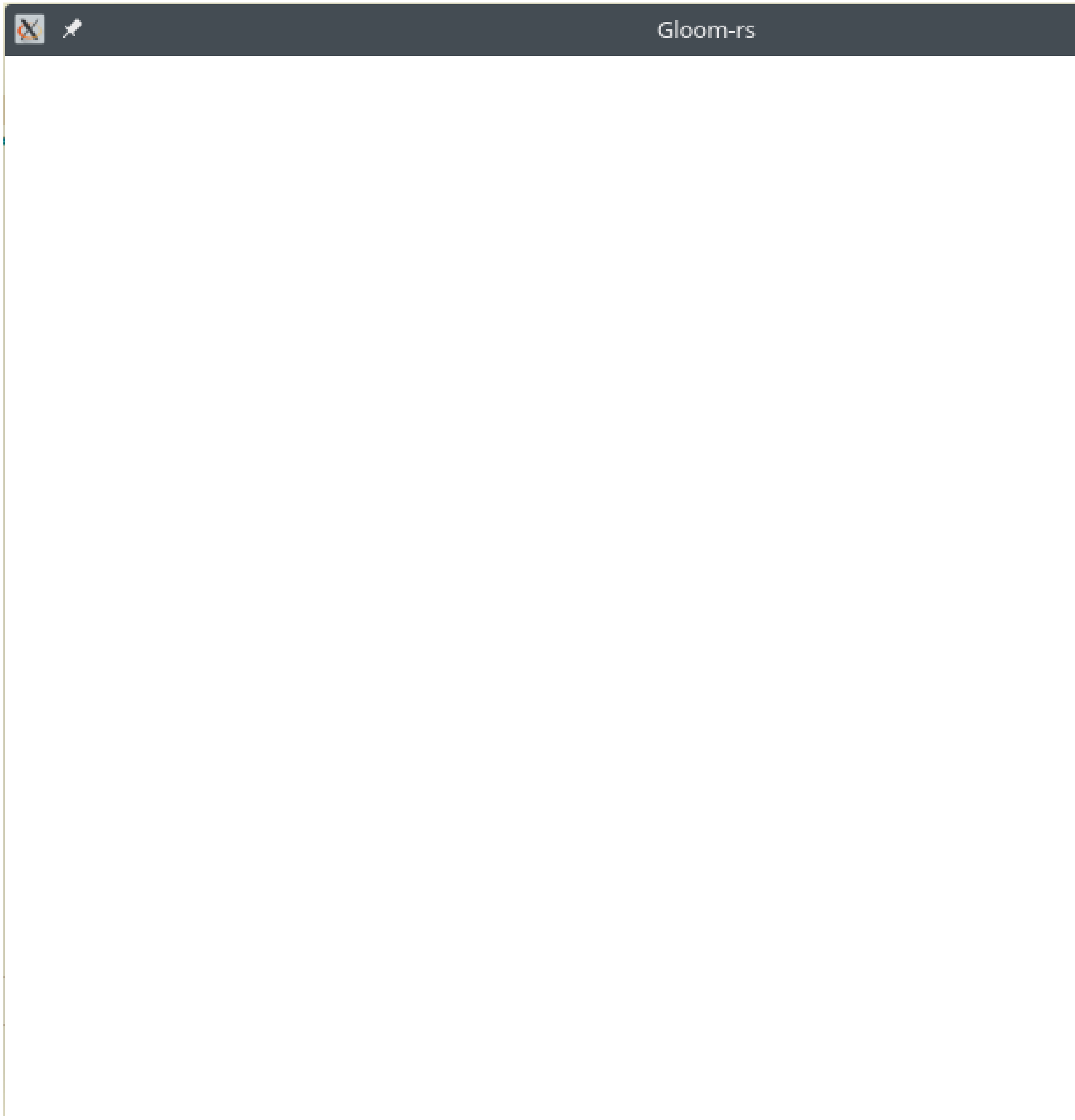
TDT4195 Assignment 1

Tobias Slettemoen Kongsvik

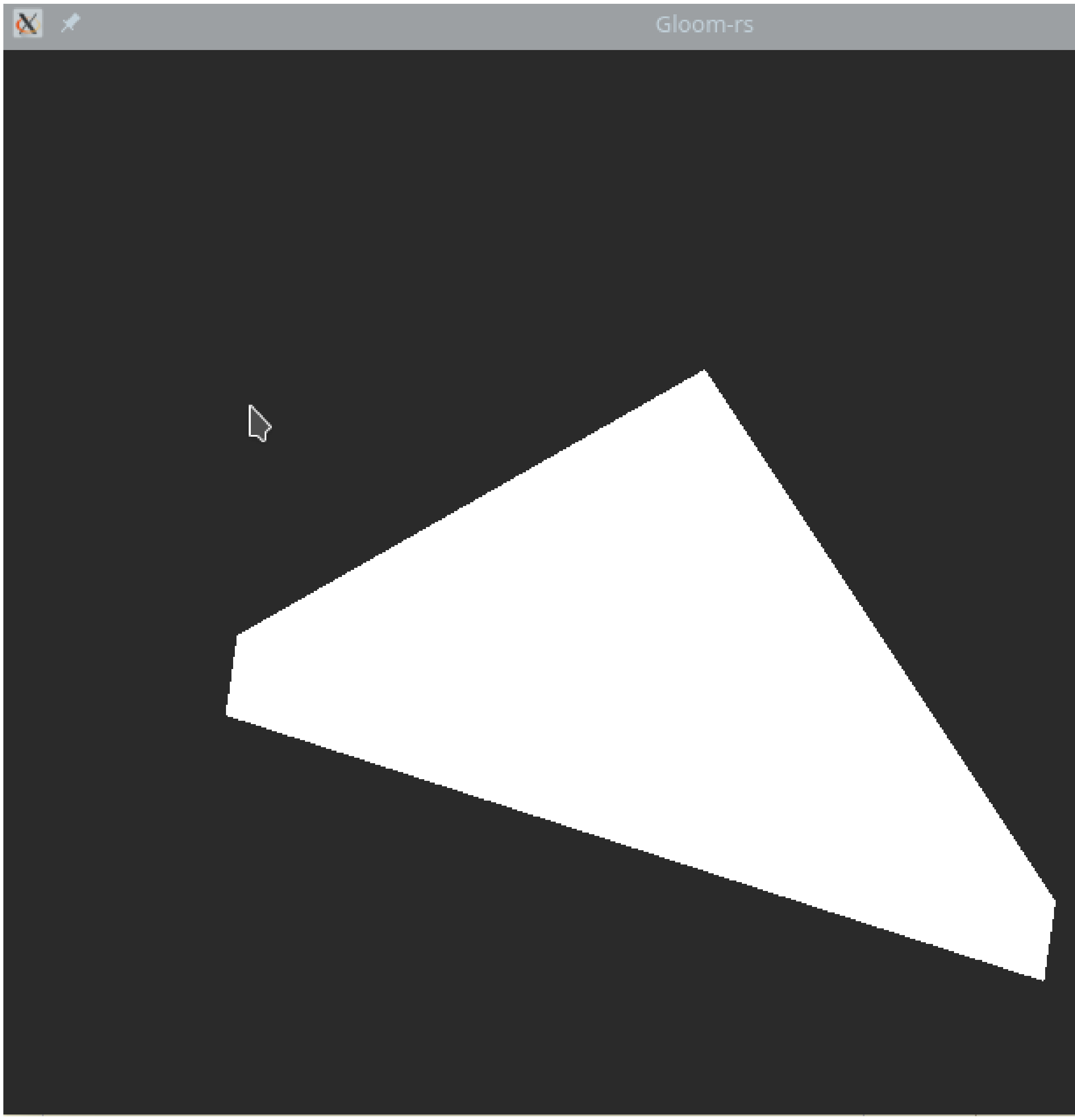
August 2020

1 Screenshot of 8 white triangles filling up the screen



2 Theory

2.1 a)



2.1.1 i)

This is called clipping.

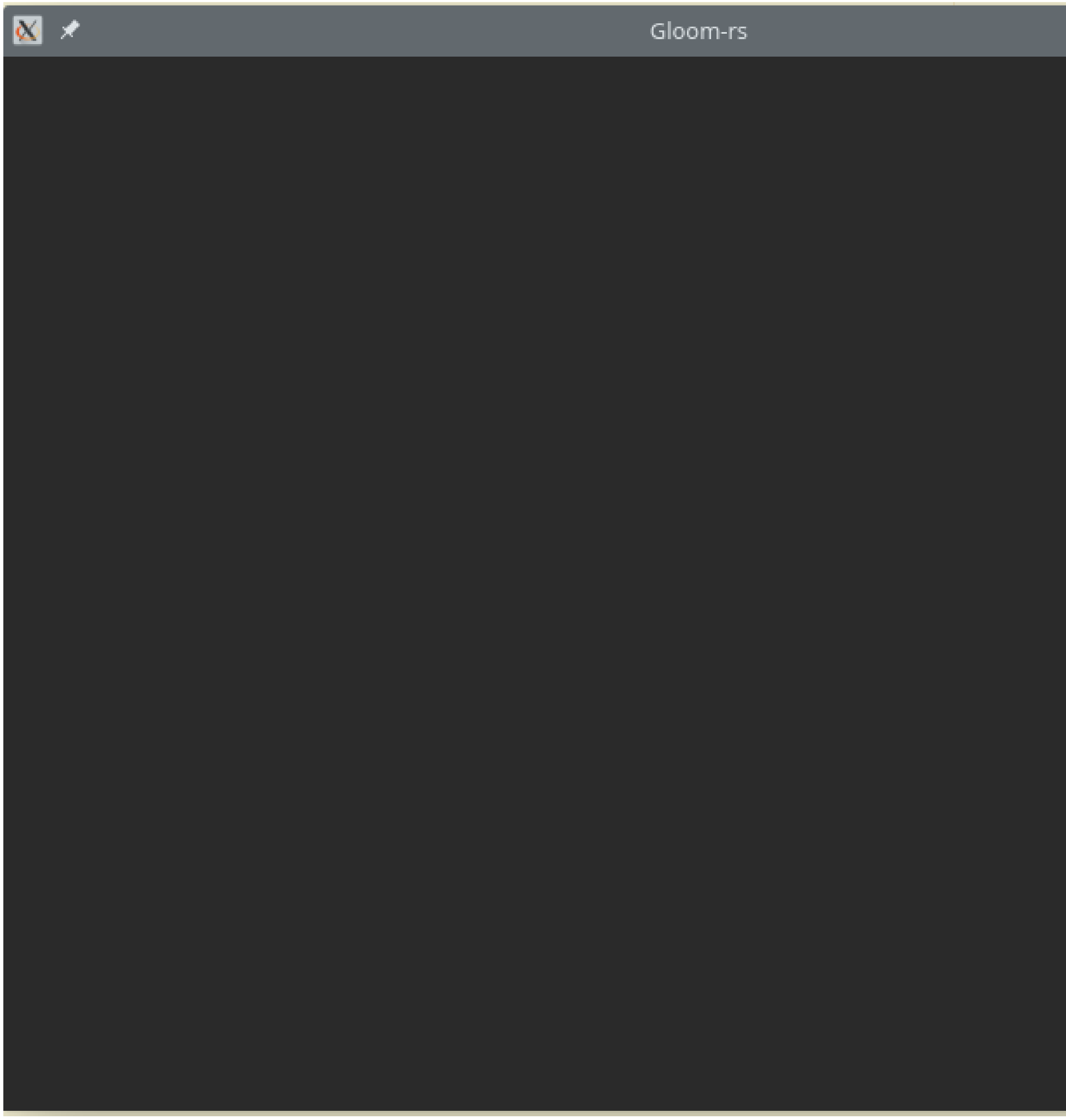
2.1.2 ii)

This occurs when you try to render something outside of your clipping volume.

2.1.3 iii)

The purpose of clipping is to not waste performance rendering things we can't even see.

2.2 b)



2.2.1 i)

The triangle dissapears.

2.2.2 ii)

It happens because of culling. Because the order of the vertices is changes we are now looking at the "back face" of the triangleand there is no point in wasting performance rendering what is usually inside the mesh or model we are rendering.

2.2.3 iii)

OpenGL by default interprets a triangle with vertices in clock-wise order as the backface, though this can be changed.

2.3 c)

2.3.1 i)

The depth buffer needs to be reset every frame so that we don't accidentally don't render something because we think it's behind something else.

2.3.2 ii)

When doing multipass rendering, for computing lightin or post processing effects the same pixel might get several passes with the fragment shader.

2.3.3 iii)

The two most common types of shader is the Fragment shader and the Vertex shader. The vertex shader processes a list of vertices. The fragment shader process a set of fragments and give each fragment a color and a depth.

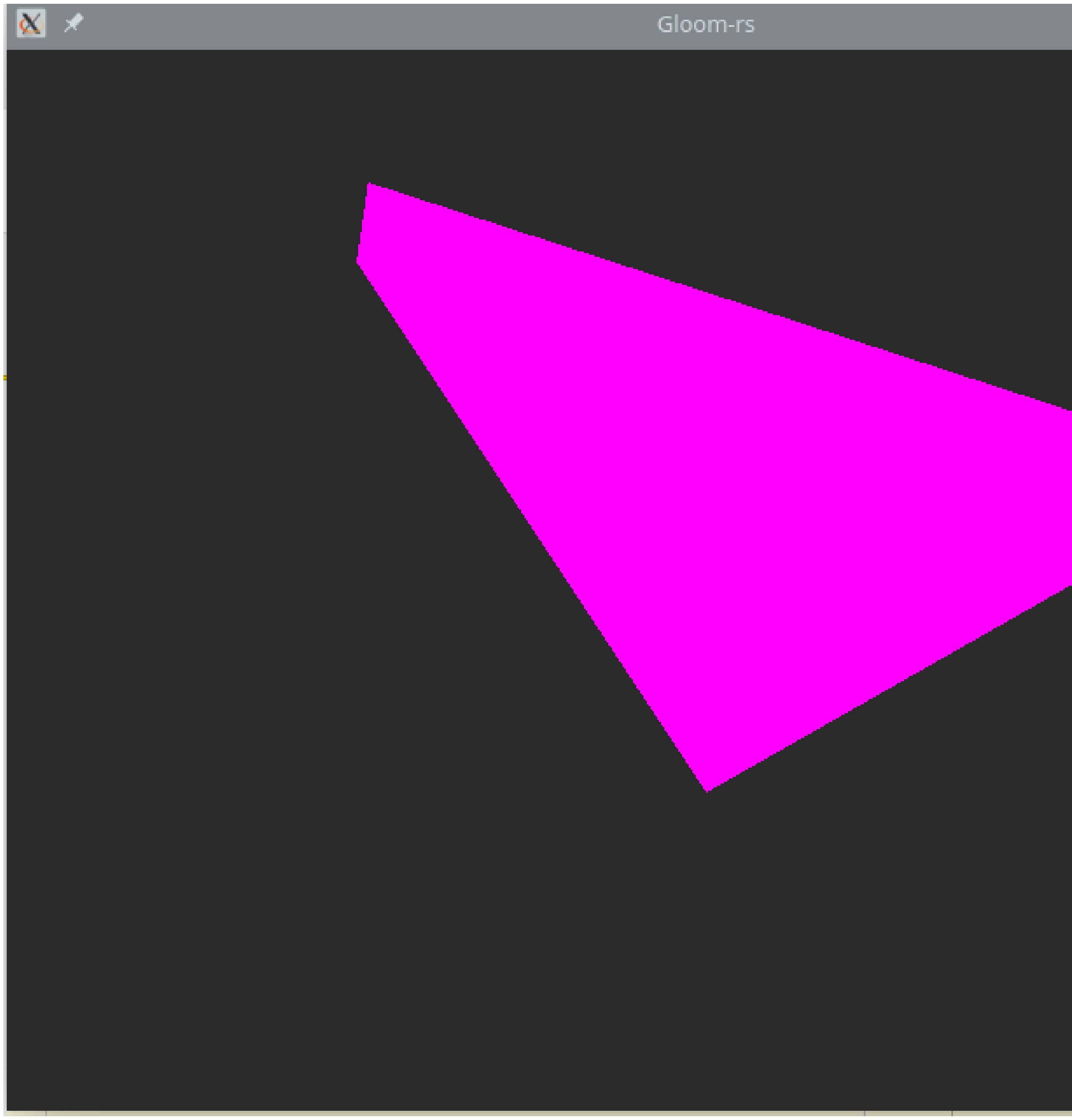
2.3.4 iv)

It's common to use index buffers because a lot of triangles in a triangle mesh will often share vertices. So we can cut down on memory usage by not duplicating the vertex data for every shared vertex.

2.3.5 (v)

You would pass a non-zero value to the last param of glVertexAttibPointer() when we have multiple types of entries in our buffer, for example positions and texCoords.

2.4 d)



2.4.1 i)

I achieved the effect by negating the x and y values of the the position vector.

2.4.2 ii)

I achieved the effect by modifying the color vector in the fragment shader.