TDT4195 Assignment 1

Olof Ljunggren

September 8, 2023

1 Visual computing fundamentals TDT4195 Assignment 1

1.1 Task 1

Drawing your first triangle [2.5 points]

1.1.1 a: Implement function create_vao

Done

1.1.2 b: Load, link and activate shaders

Done

1.1.3 c: Define and instantiate a VAO and draw triangles

Done

1.2 Task 2

Geometry and Theory [1.5 point]

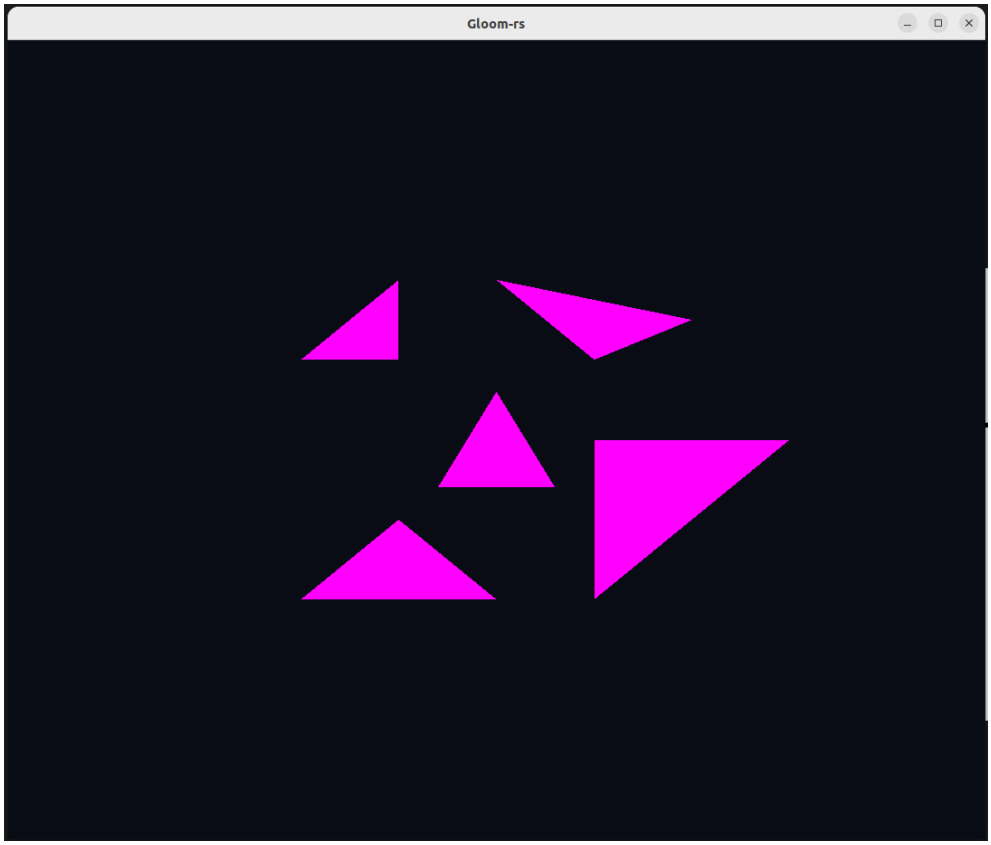
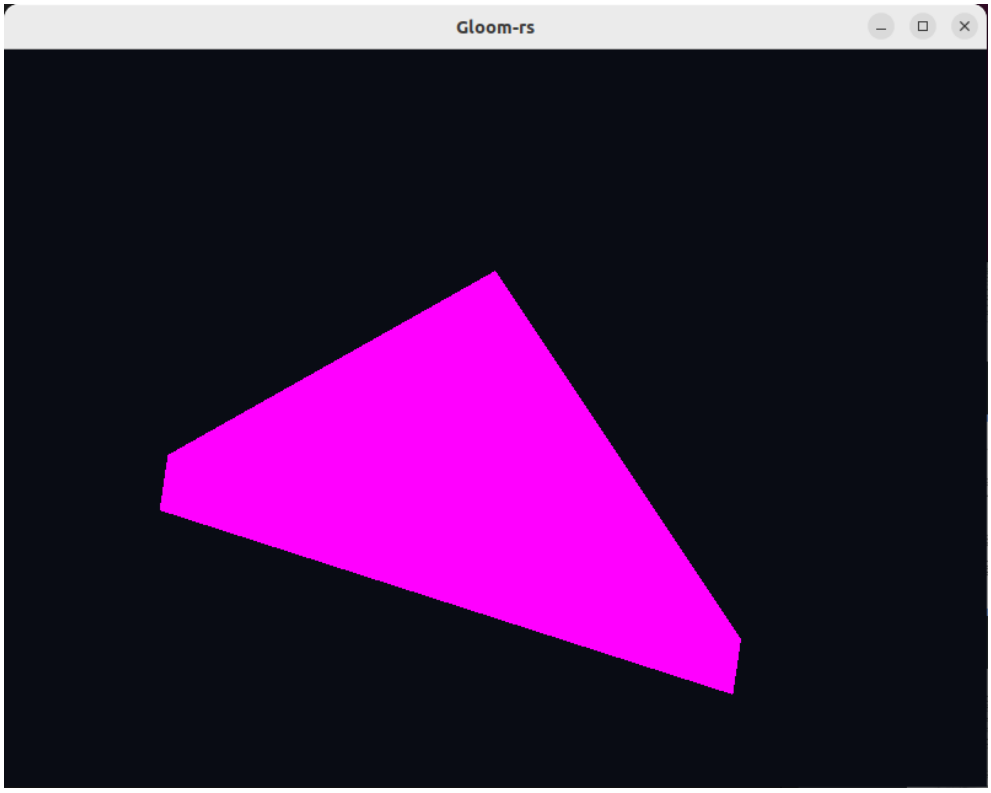


Figure 1: My first five triangles!

1.2.1 Draw a signle triangle and explain result



1. What is the name of this phenomenon? - This phenomenon is called clipping.
2. When does it occur?
   - It occurs when vertices is located outside the range of the clipping window (cube). Default cube (-1, -1, -1) to (1, 1, 1).
3. What is its purpose?
   - The primarily purpose is to remove lines, vertices and objects outside the view. This makes computations faster since you dont have to render unnecessary objects.

1.2.2 Draw triangle indicies in another order

1. What happens?
   - We can not see the triangles with swapped order.
2. Why does it happen?
   - Because OPENGL as default only displays triangles being oriented counter clock-wise, wtih normal pointing in the viewers direction.
3. What is the condition under which this effect occurs? Determine a rule.
   - The order of which the indexes is specified must correspond to counter clock-wise ordered vertices.

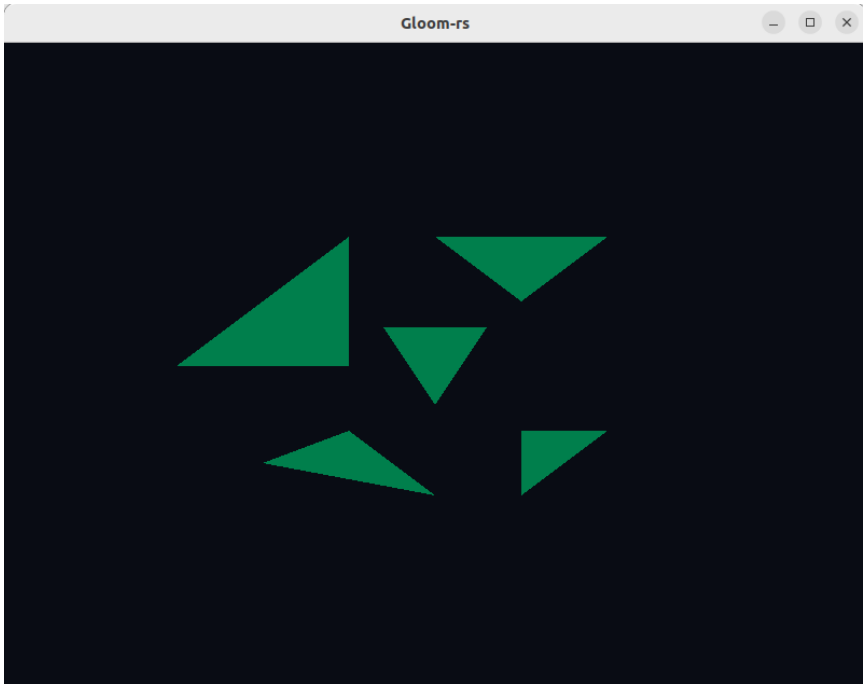
1.2.3 Knowledge questions

1. Why does the depth buffer need to be reset each frame?
   - Because each frame we want to draw objects in the correct order. Therefore we need to calculate the depth every frame.

2. In which situation can the Fragment Shader be executed multiple times for the same pixel?
    - The fragment shader can be executed multiple times for the same pixel when a later processed fragment is projected on the same pixel but have another depth.
3. What are the two most commonly used types of Shaders? What are the responsibilities of each of them?
    - They are called the fragment shader and vertex shader. The vertex shader is doing the transformations for every vertex. The fragment shader is calculating the final color of the pixel corresponding to a fragment.
4. 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)?
    - The index buffer is used because it simplifies the process of using the same vertices multiple times. Therefore it is more efficient. Furthermore bodies get "water tight".
5. 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.
    - If there is multiple entries in the buffer it is possible to define a offset. For instance to texture data. This makes it possible for OPENGL to find the wanted data.

1.2.4 Modify the shaders

1. Mirror/flip the whole scene both horizontally and vertically at the same time and 2. Change the colour of the drawn triangle(s) to a different colour.
    - In order to make flip the axises I constructed a scaling matrix and scaled x and y values with -1. Then i multiplied the matrix with the position coordinates in the vertex shader. To change colour, I just changed the values in the fragment shader. I figured the colour vec4 being output was (R, G, B, Opacity).



1.3 Task 3

Optional Bonus Challenges [up to 0.2 bonus points]

1.3.1 Checkerboard

In order to draw a checkerboard, I did some maths in the fragment shader using floor function and gl_FragCoord.

1.3.2 Circle

To draw a circle I began with creating vertices in a circle and then connect these into triangles. This one was made in the "main.rs". After that I also made a smaller green circle using gl_FragCoord and calculating distance from center in the fragment shader.

1.3.3 Sine

Finally I made two sine functions using y=sin(x) and some abs-logic, all coded in the fragment shader.

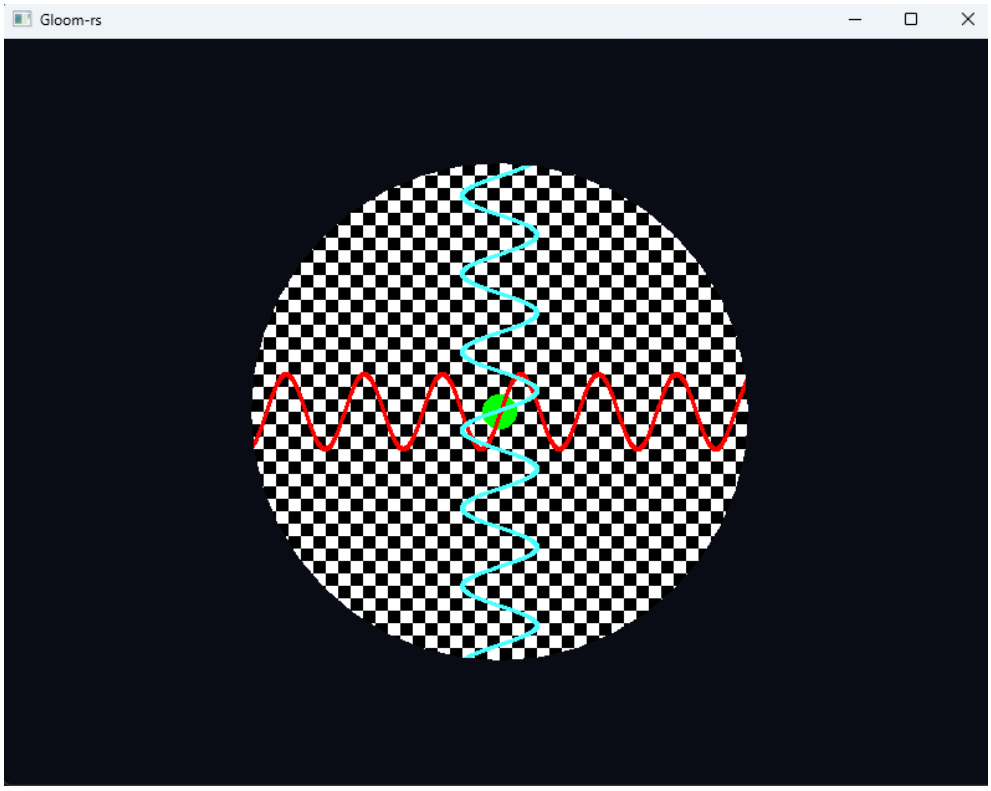


Figure 2: Bonus tasks!