TDT4195 assignment report 2

Nils Reed

September 2021

1b

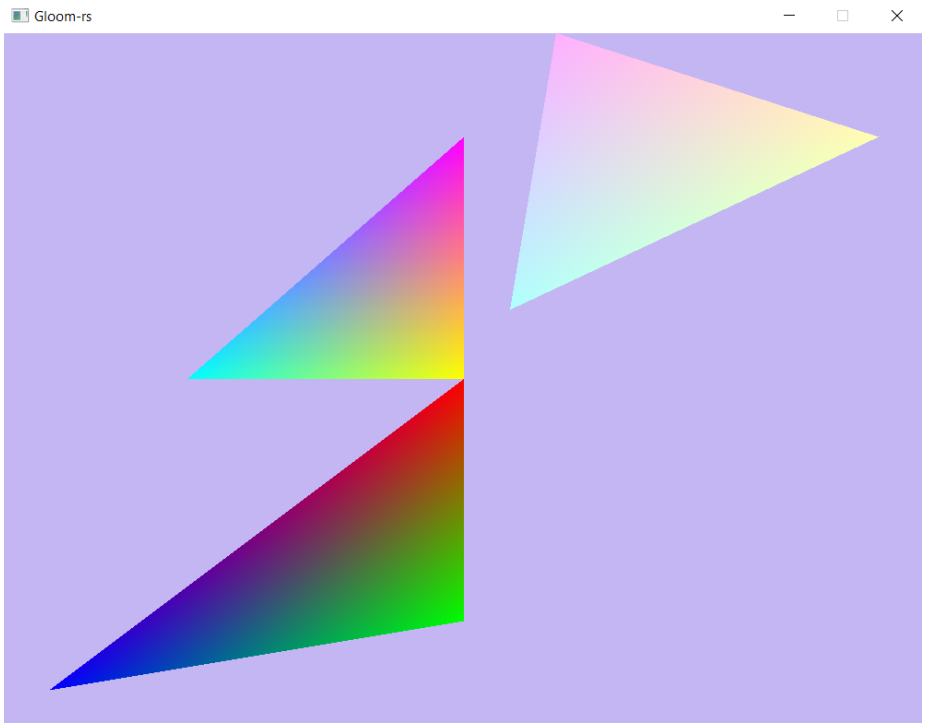

Figure 1: Three colourful triangles :)

2a

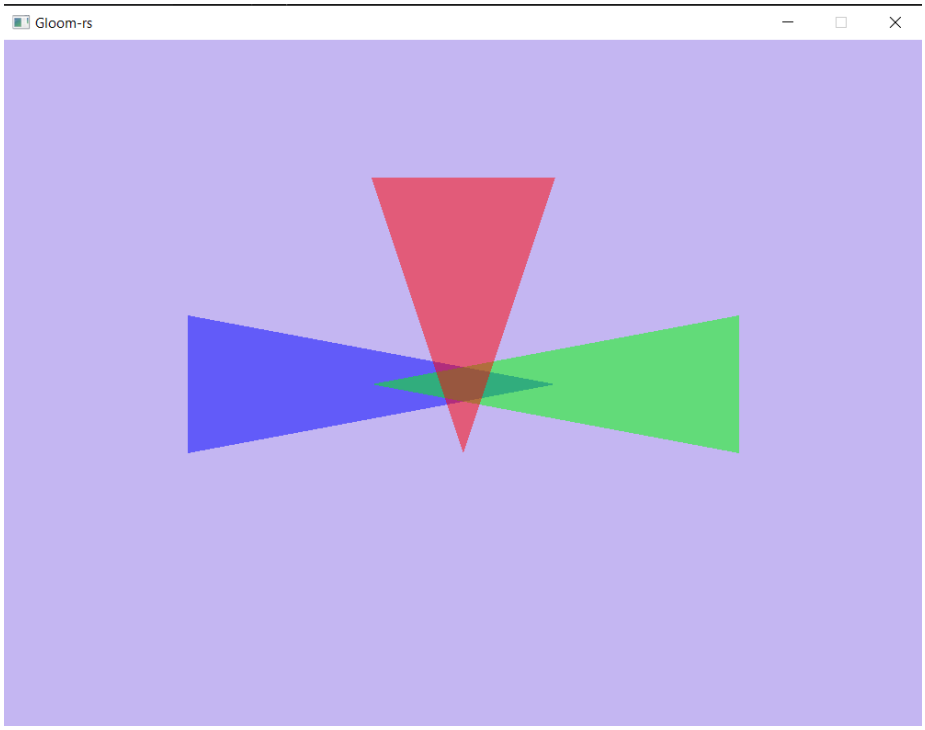

Figure 2: Three overlapping, partially transparent, monochrome triangles

2b

i)

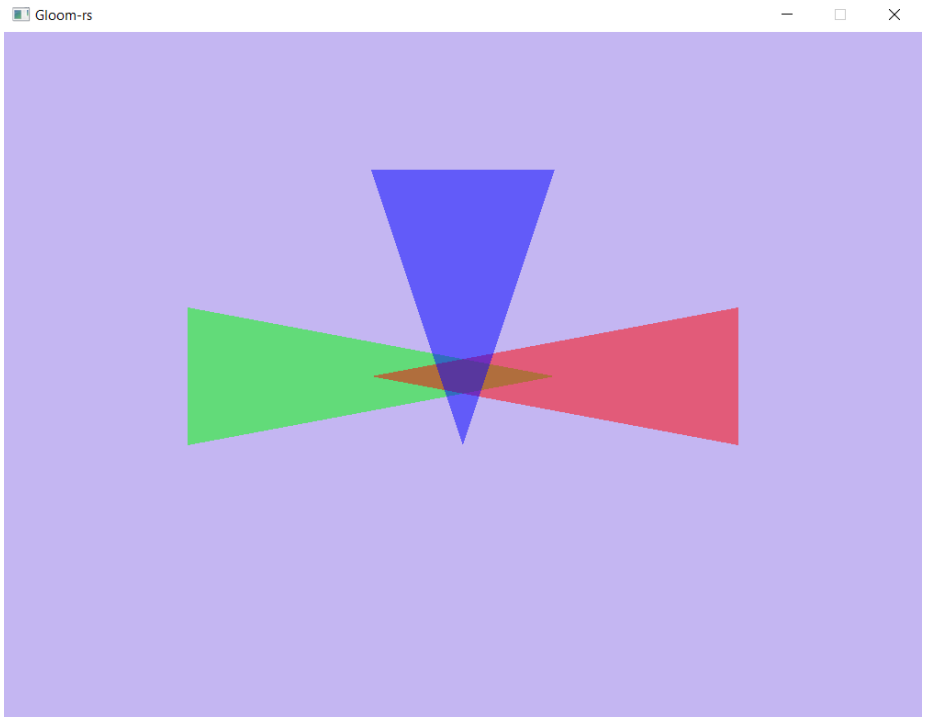

Figure 3: Colours swapped around

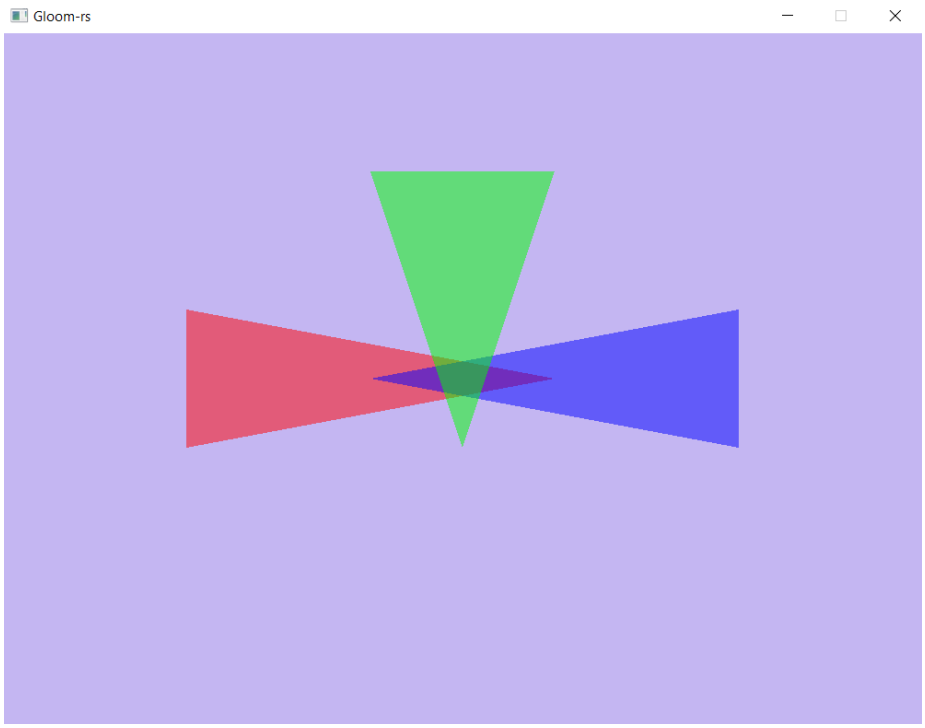

Figure 4: Colours swapped around again

The first figure (2) has a red-brown colour in the intersection, the second figure (3) has a blue-purple colour in the intersection and the third figure (4) has a green-purple colour in the intersection.

The effect seems to be that the colour at the front is the one most strongly represented in the blend, then the one in the middle, and we see very little of the colour all the way at the back. This makes sense when looking at the formula 1

Changing the triangle colours caused this to happen by placing the colours behind one another in a different order, thus yielding another blend according to the formula described in the task:

$$C_{new} = C_{src} * \alpha_{src} + C_{dest} * (1 - \alpha_{src}) \tag{1}$$

(With C: Colour, src: Source, dest: destination).

ii)

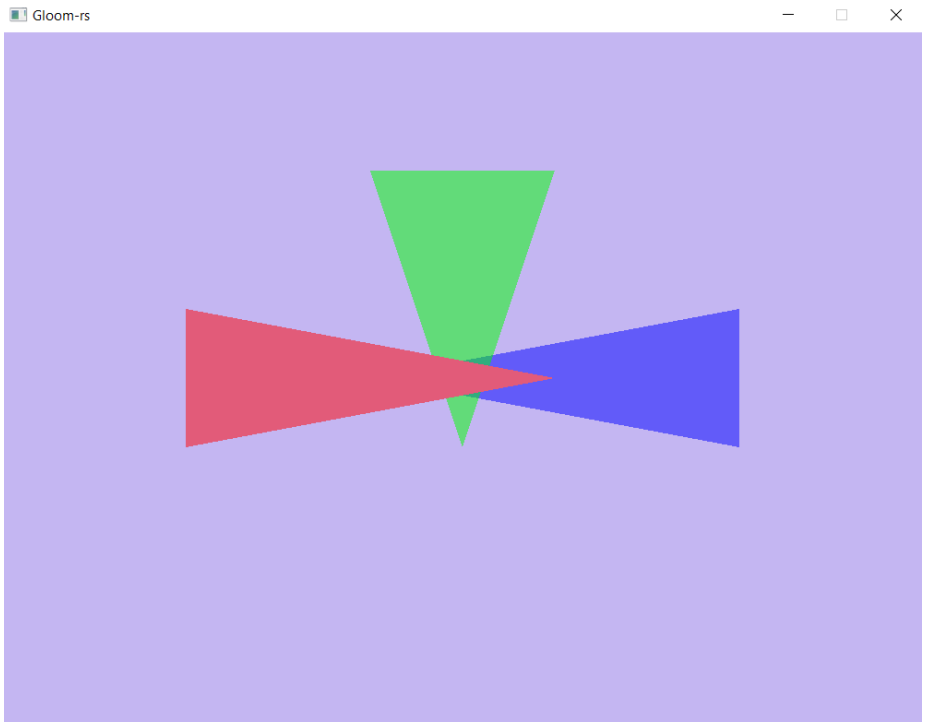

Figure 5: Order of drawing swapped around

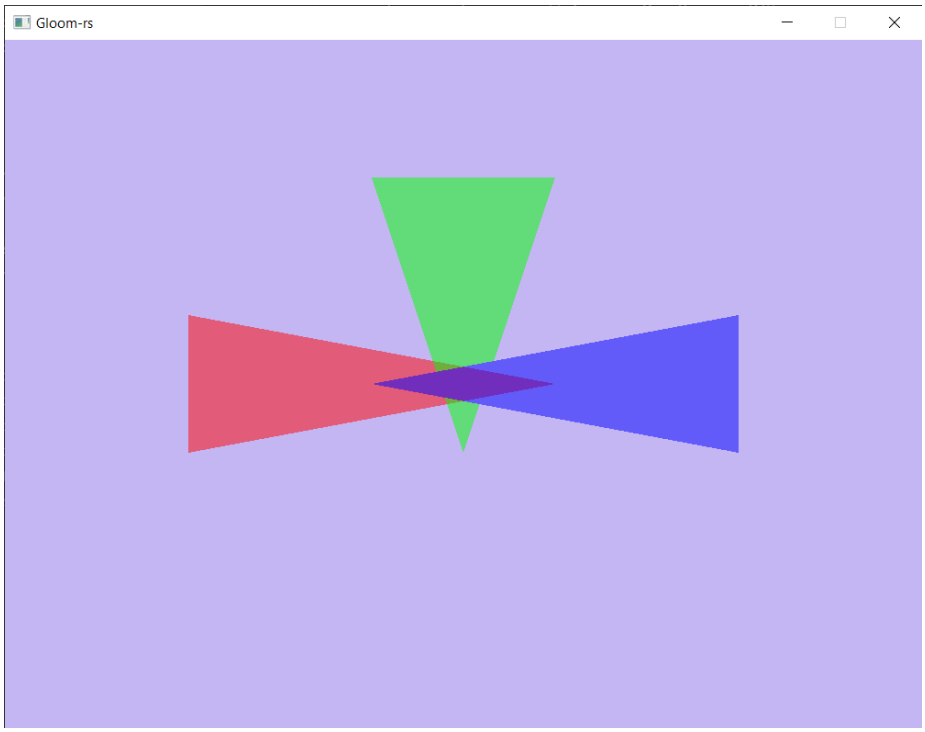

Figure 6: Order of drawing swapped around again

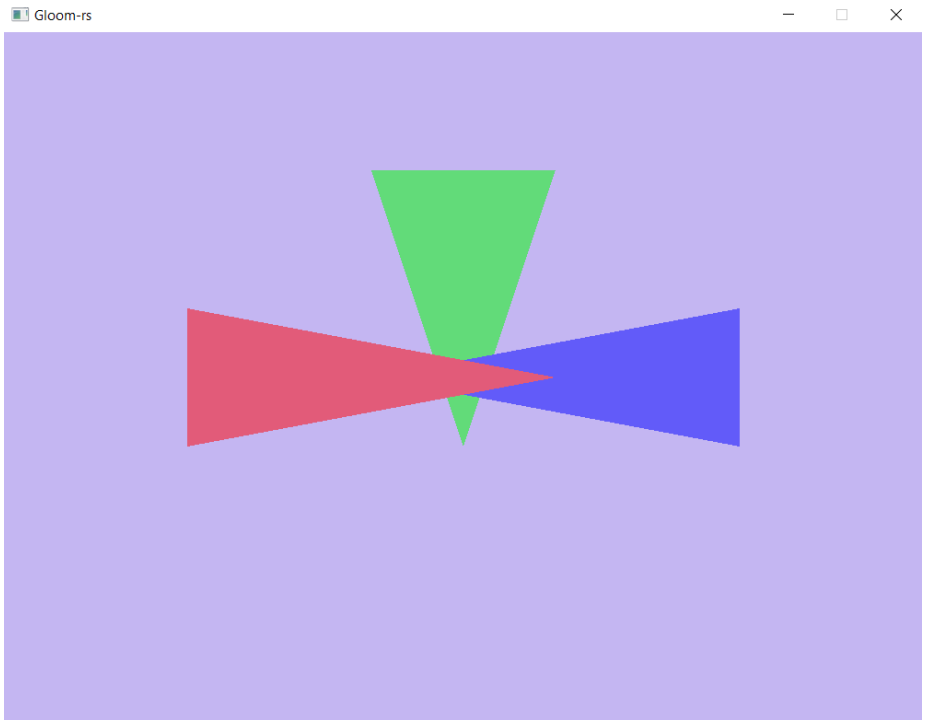

Figure 7: Order of drawing swapped around again again

In fig 5 the red triangle is drawn first, then the blue and then the green. This is visible since the red is all the way at the front but is not blended into anything

In fig 6 the red is drawn, then the blue and then the green. This is visible because the green is blended into the red but not in the intersection between all three triangles.

In fig 7 the red is drawn first, then the blue, then the green. This is evident because none of the colours are blended with each other,

The depth buffer is the cause here, and this is evident from equation 1. When a triangle is drawn, OpenGL checks if there are any triangles behind it, but it only checks the ones already drawn, the ones registered in the depth buffer. This means that triangles that aren't drawn yet, but lie behind it won't get blended into the intersection area, and when they are drawn, they're just ignored in that area since there is already a triangle in front there.

3b

Modifying a stretches and squeezes the figures along the x axis. This operation must therefore be a scaling along the x-axis.

Modifying b also stretches the figure, but not along any static axis (axis not independent of value). The figures are stretched outward and inward. Also, the points furthest from the origin in the y direction are stretched the most in relation to their x-coorinate. This must therefore be a shearing operation along the x-axis.

Modifying c causes the figures to slide to the right and to the left. This is obviously translation on the x-axis.

Modifying d also stretches the figure, but not along any static axis (axis not independent of value). The figures are stretched upward and downward. Also, the points furthest from the origin in the x direction are stretched the most in relation to their y-coorinate. This must therefore be a shearing operation along the y-axis.

Modifying e stretches and squeezes the figures along the y axis. This operation must therefore be a scaling along the y-axis..

Modifying c causes the figures to slide up and down. This is obviously translation on the y-axis.

3c

I Can be certain that none of the transformations were observations because they either moved all points of the figure uniformly in one direction, or they morphed the figure (i.e. angles are not preserved). A rotation preserves angles and will move points differently based on their position.