Assignment 2

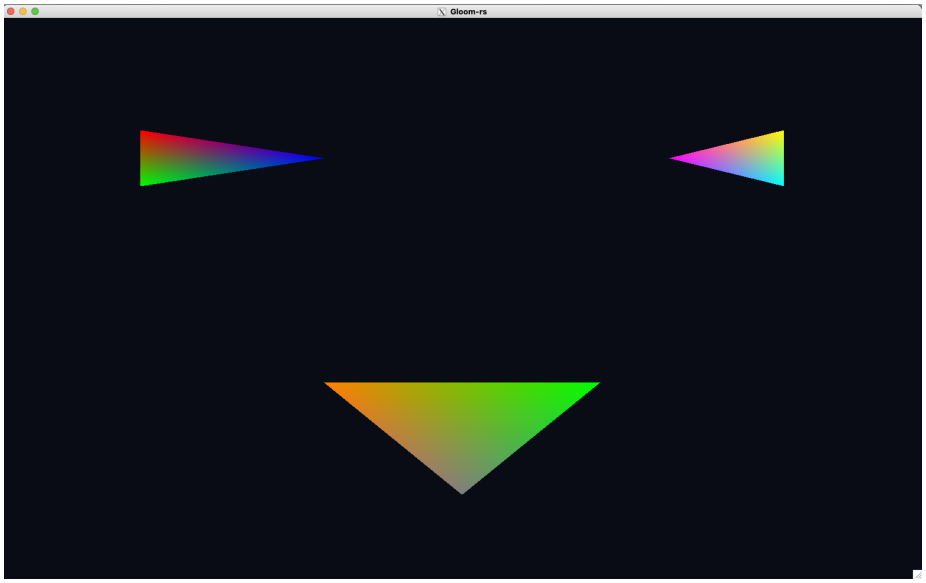
Henrik Horpedal

September 20, 2024

Task 1

b)

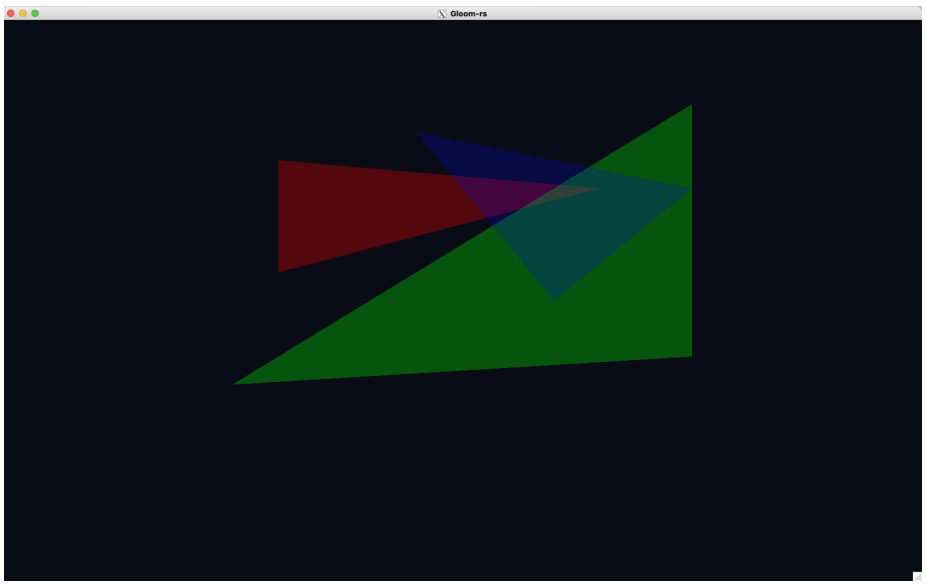
Three triangles all with different colored vertices:



OpenGL interpolates vertex attributes like colors. That way we get a smooth color gradient between vertices of different colors. By default it takes the depth into account, called smooth interpolation.

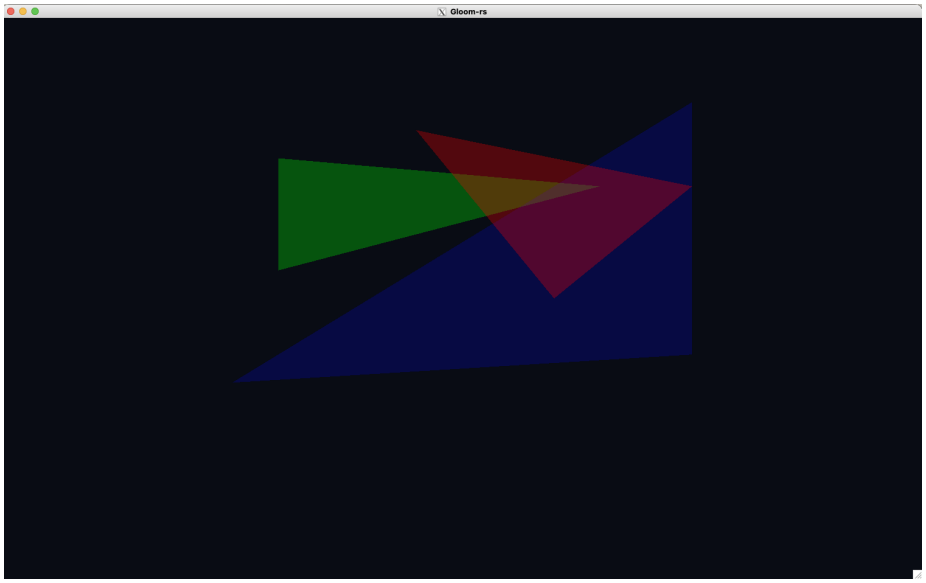
Task 2

a)



b)

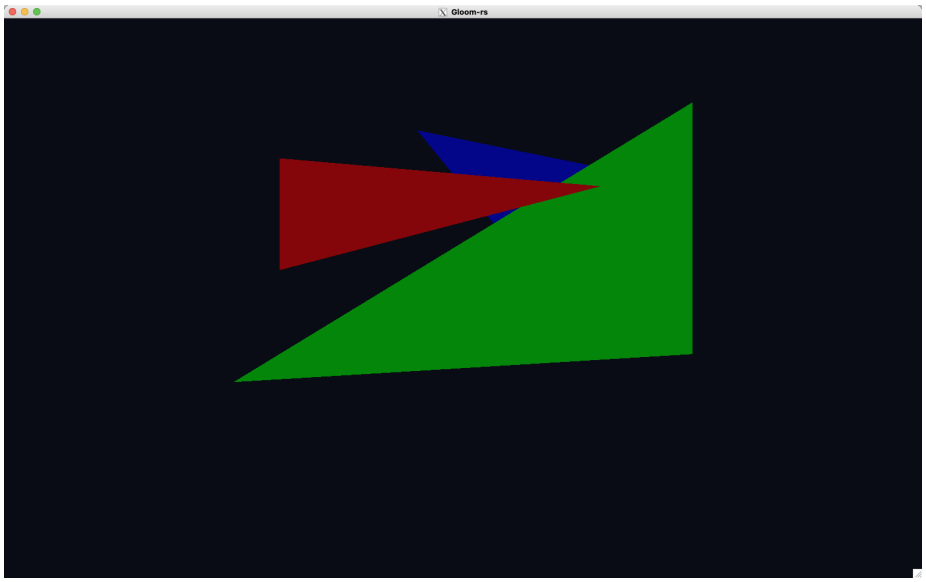
i)



You can see that the triangle closest dominates in the area where all three triangles overlap. My guess is that the blended colors are calculated as an average sequentially as triangles are loaded, causing the first two colors to contribute 25% while the last contributes by 50% to the final color. If we study the result from the previous task, there is actually an optical illusion. The blended color looks to be red, but if you zoom in on the only that color, it is clearly blue-ish!

ii)

As we can see, when changing the z-coordinates so that triangles are drawn front-to-back, the transparency effect is lost. It is because OpenGL blends the transparency color based on what is behind the triangle. So when the triangles are drawn front to back, OpenGL thinks that there is nothing behind the transparent color, and therefore there is nothing to show "through" the fragment. The depth-buffer is as we know, responsible for keeping track of which object is closest in each pixel.


Gloom-rs

Task 3

a)

I used a uniform varible to slowly modify each of the elements as the task suggested. This was what i observed for each parameter:

a

This changed the scaling along the x-axis.

b

This is shearing on x-axis.

c

This controls translation along the x-axis.

d

This is shearing on y-axis.

e

This is scaling in the y-axis.

f

This is translation along the y-axis.

b)

A rotation could bee seen as a combined shear and scaling-transformation. Since we only did one at a time, there was no rotation.

Task 4

I used the recommended key-binds, but added the key R as a "reset to starting-point" so that you don't loose were you are.

Task 5

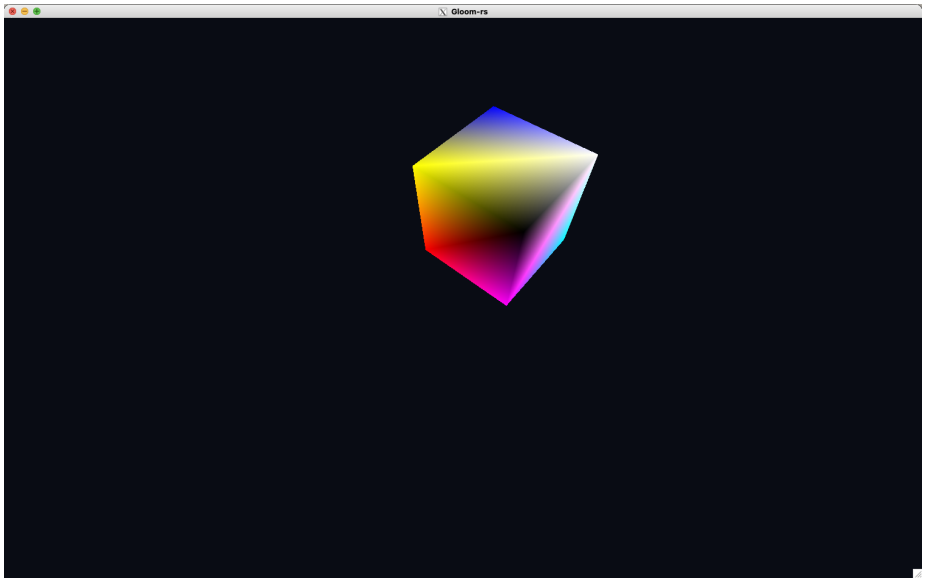
a)

Coded it.

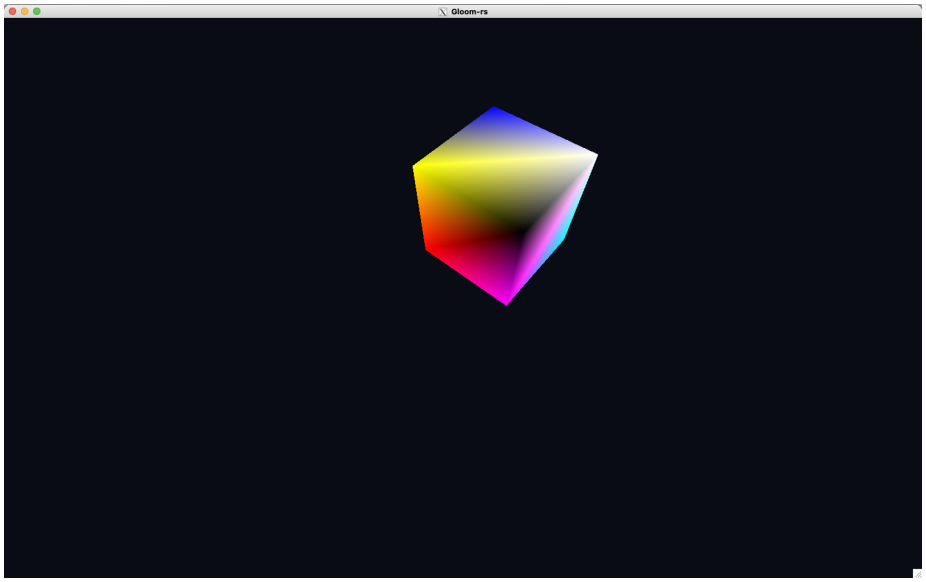
b)

I tried a lot of stuff to try to spot any difference when using 'noperspective', but I literally cannot spot any difference. My guess is that you need a more spesific scene to see a difference.

With smooth interpolation:



With 'noperspective':


Gloom-rs