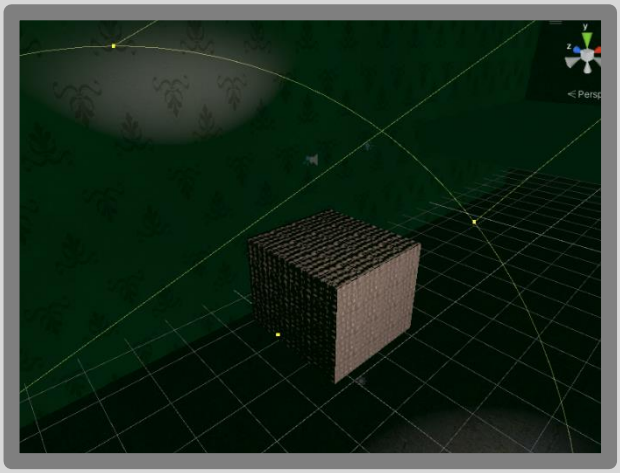
Portfolio

Blinn-Phong lighting model.

Implemented a basic Blinn-Phong lighting shader in unity unlit shaders. Including textures, normal maps and custom ambient light. Normal map achieved through a tangent to world conversion.



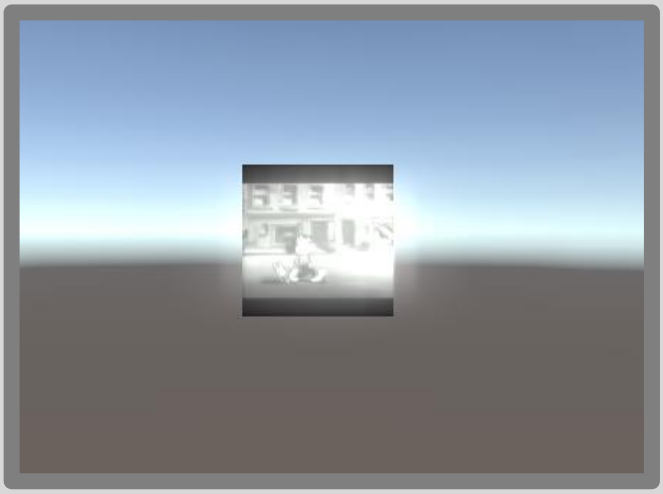
Low resolution pixilated effect.

Created a post processing effect that emulates the pixelated look of an older console. The intensity of the pixelation can be adjusted with a parameter.
It use GPU down sampling with a point sampler. This stretches out individual pixels to be bigger instead of taking an average of surrounding pixels like most sampling methods.

TV bloom effect.

Another post processing effect. Similar use of GPU down sampling with a box blur instead of point sampling to create blur. Uses parameters to control power and spread of glow.

Originally it used a manual box blur over full sized texture, but this method is more optimized.



Proximity based vertex shrinker.

Vertex displacement shader which shrinks the geometry based on how close the player is. Parameterized sensitivity.

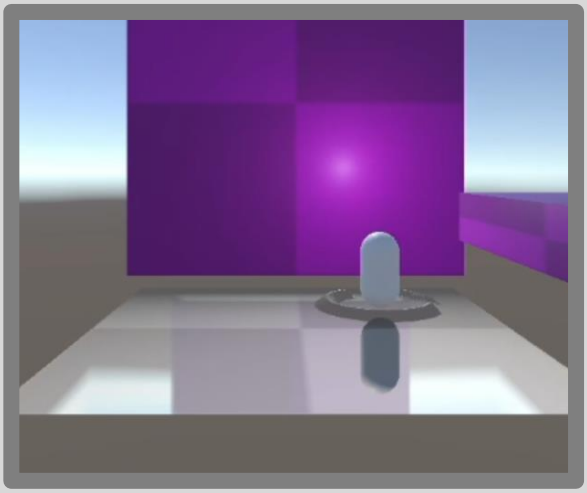
(Current code on GitHub uses 2 if statements this has been replace by lerp(lerp(-1,t, t > -1), 1, t>1) and lerp(lerp(-1, anotherthing, anotherthing > -1), 1, anotherthing >1). Although this might already be done by compiler)

[Work in Progress and Experimental]
Real time water with reflection and ripples.

Instead of using the traditional 2d method of flipping the image or the 3d one of generating a cube map every frame like RDR2.

I combined them by having an initial cube map then reflecting ONLY the player using a command buffer.
This would allow me to apply a unique material to this reflection to make the real player and the reflection different.

Future plans include sum of sines noise. Replacing vertex displacement with a normal map for ripples. Making reflection more accurate to real position.
Fix ripple overlapping bug.



Flocking simulation real time normal map.

Using unity jobs up to 32 entities do a flocking simulation having the boids track the players movement if close and if far they will scatter. Returning a position and rotation.

From this a height map is produced into a render texture using a compute shader. From here it is converted into a normal map using another compute shader. (These could be combined into one)

From here that normal map is sent to the Blinn Phong shader to produce the effect of something crawling inside the wall.

Future plans: Instead of generating a normal map send the height map in as a parameter increasing the strength of the normal map in the individual pixel.

This allows me to have both this effect as well as use another person’s normal map.