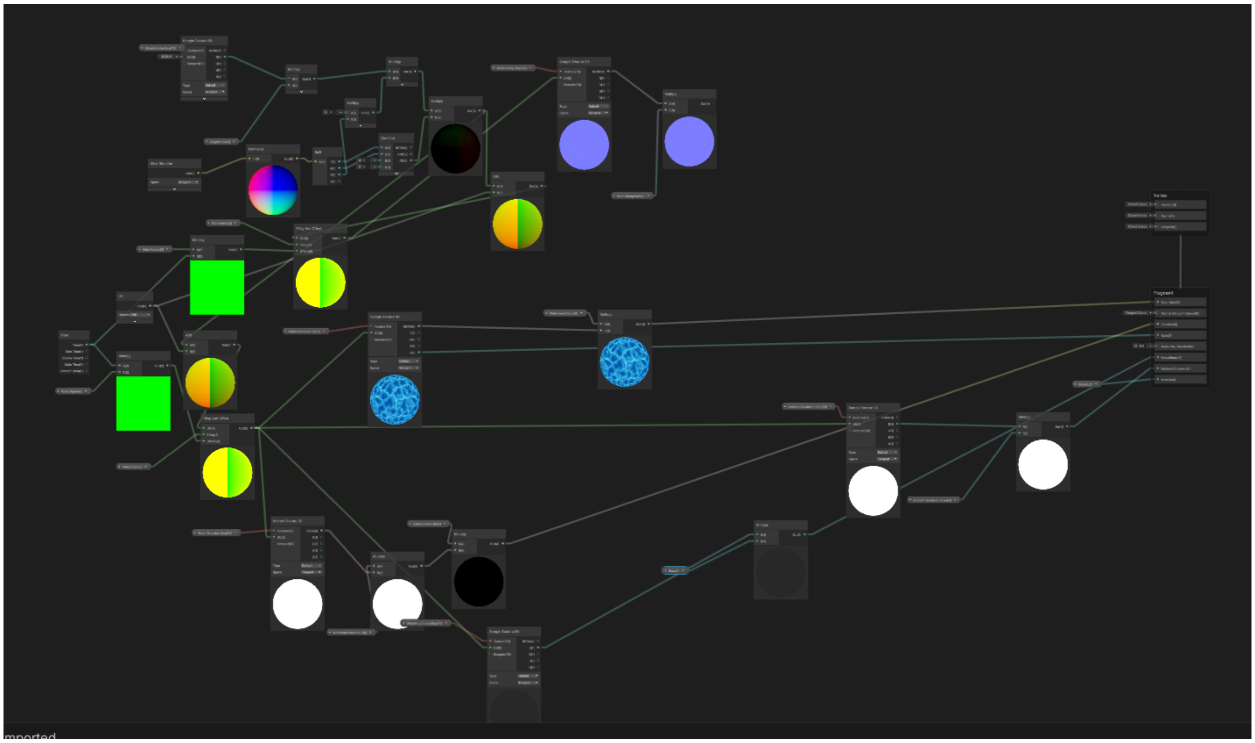
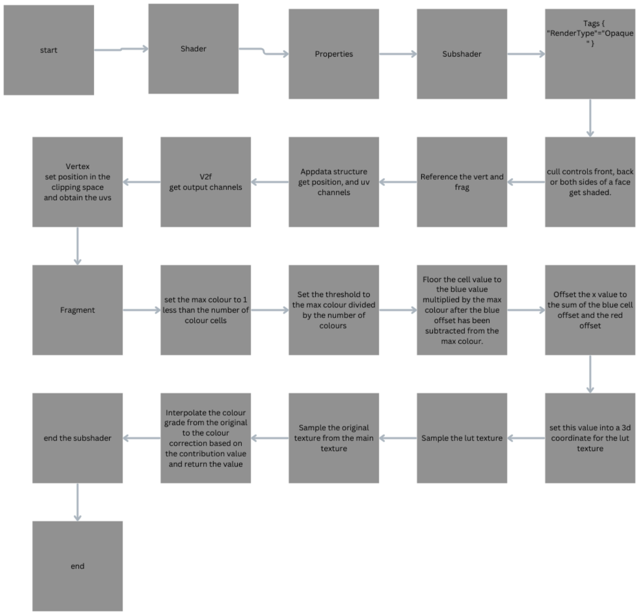
Jamie Mason 100819142

There were 4 shaders that were created the illumination shader, the color correction, the hologram, and the water shader.

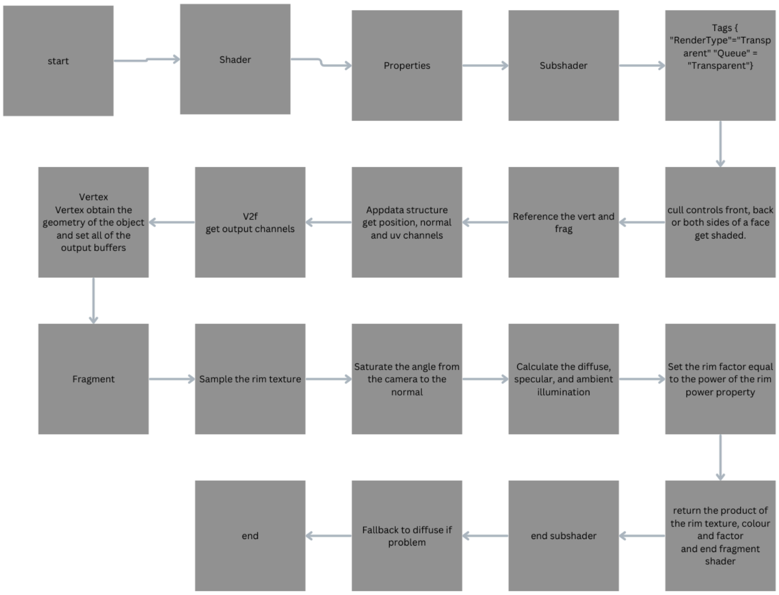
The water shader uses a scrolling effect which is controlled by a time node. This time node outputs the time into the tile and offset node which will control the uvs for the bump map and one that will control the scrolling uvs for the roughness, colour, emission and ambient occlusion maps. The parallax map controls the contrast and difference in bump height and main colour maps. This uv offset is sampled and used set as the tile and offset standard uv the output of this is then put into the rest of the maps which will allow them to scroll. The roughness map is outputted through the smoothness channel. Ambient occlusion is multiplied by its scaling value and outputted into the ambient occlusion channel. The emission is outputted in the emission channel. The metallic in the metallic. In the diagram below some of the outputs are white because some of the properties do not have a texture assigned to them because the water PBR did not have some of these kinds of texture maps.



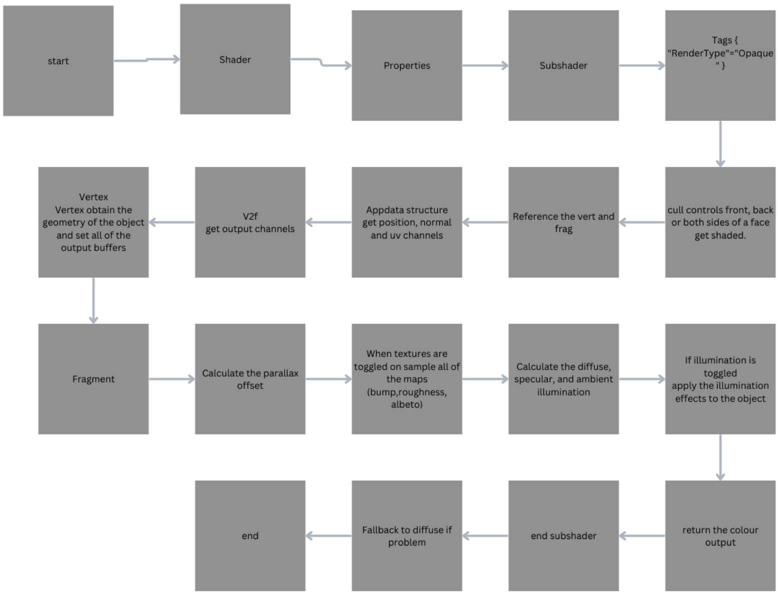
The Colour-grade shader works by sampling a point on the lut texel The offset on the x is set to the blue offset, which determines the cell the red offset determines where in the cell the offset x will be and the green offset determines the y offset in the colour correction. The colour correct is then interpolated from 0 to 1 which is set by the contribution property. This is then applied as a render image to the camera to create a colour correction. The lut was designed to be green-blue because I think it gives the game a cartoon and fun colour feel.


start
Shader
Properties
Subshader
Tags { "RenderType"="Opaque" }
cull controls front, back or both sides of a face get shaded.
Reference the vert and frag
Appdata structure get position, and uv channels
V2f get output channels
Vertex set position in the clipping space and obtain the uvs
Fragment
set the max colour to 1 less than the number of colour cells
Set the threshold to the max colour divided by the number of colours
Floor the cell value to the blue value multiplied by the max colour after the blue offset has been subtracted from the max colour.
Offset the x value to the sum of the blue cell offset and the red offset
set this value into a 3d coordinate for the lut texture
Sample the lut texture
Sample the original texture from the main texture
Interpolate the colour grade from the original to the colour correction based on the contribution value and return the value
end the subshader
end

The Hologram shader works by sampling three properties colour, intensity, rim texture, and a toggle property. The colour property modifies the colour of the rim lighting. The intensity modifies the strength of the rim lighting and the toggle just controls whether there will or will not be rim lighting. Since there are no boolean values in shader properties I set it to a float value and if the value is greater than or equal to 0.5 rim lighting is toggled on else no rim lighting. In the vertex shader the shader sources the object geometry such as the vertex transforms, normals, and uvs. Then, the position transforms in clipping space, the normals in world space and the uvs are just sampled to the uvs. I also calculate the view position of the camera. In the fragment shader, I first sample the rim texture using the obtained unwrapped object uv. Then I calculate the rim factor by obtaining the angle from the camera to the object's normals. To then be able to amplify this rim factor it is set to the power of the rim power property. A rim power of 0 fills the whole thing. I thought this made it easier if I wanted the effect to cover the whole thing because anything to the power of 0 = 1. Finally, the colour output that is returned is the product of the rim colour the rim factor and the rim texture.


start
Shader
Properties
Subshader
Tags { "RenderType"="Transparent" "Queue" = "Transparent"}
cull controls front, back or both sides of a face get shaded.
Reference the vert and frag
Appdata structure get position, normal and uv channels
V2f get output channels
Vertex Vertex obtain the geometry of the object and set all of the output buffers
Fragment
Sample the rim texture
Saturate the angle from the camera to the normal
Calculate the diffuse, specular, and ambient illumination
Set the rim factor equal to the power of the rim power property
return the product of the rim texture, colour and factor and end fragment shader
end subshader
Fallback to diffuse if problem
end

The maps and illumination effect work similarly to the water effect the difference also accounts for diffuse, specular and ambient lighting in the shader. It samples all the geometry of the object then it uses that information to apply all of the illumination, textures and colour to the object in the scene.


start
Shader
Properties
Subshader
Tags { "RenderType"="Opaque" }
cull controls front, back or both sides of a face get shaded.
Reference the vert and frag
Appdata structure get position, normal and uv channels
V2f get output channels
Vertex Vertex obtain the geometry of the object and set all of the output buffers
Fragment
Calculate the parallax offset
When textures are toggled on sample all of the maps (bump,roughness, albeto)
Calculate the diffuse, specular, and ambient illumination
If illumination is toggled apply the illumination effects to the object
return the colour output
end subshader
Fallback to diffuse if problem
end