

## BASE SCENE

My assigned scene to recreate was Street Fighter 2, a game where two characters face off in a battle. Since my main goal was to focus on implementing the shaders, I did a very simplistic recreation of the game using Unity primitives and models that I already had from previous classes(I used this Buff Peter Griffin model in a few shader demonstrations, and thought that the model fit the theme of Street Fighter well enough), as well as a bonus custom made UI element that I created quickly in Krita.



## DECAL

My first implementation was the Decal shader. I decided to apply it to the clothing part of each model, as I felt that adding a Decal shader to them would make them both benefit from having a more unique texture to their clothes. Customizing the Decal shader with different textures to make a unique one for each of them helps differentiate the two characters as well, allowing the player to be able to determine which one is theirs. Retep, the player's enemy, was given a red tint to signify that he is evil. I used a shader script to sample the textures and blend them together, and used a toggle to determine which textures would be blended and which not. I also used the Decal shader to add a texture to the ground, but didn't toggle the blended texture setting to keep it standard.

For my additions to the shader, I decided to add a third texture for the purpose of tinting all of the models collectively. I noticed that after implementing the shader, the textures were a little desaturated for my liking, so I added a third texture to the script that had the colour "black" to it in order to saturate the colours of the textures and make them stand out a bit more.

```
Properties
{
    _MainTex ("Main Texture", 2D) = "white" {} // Main texture
    _DecalTex ("Decal Texture", 2D) = "white" {} // Decal texture
    _TexThree ("Texture Three", 2D) = "black" {} //Third extra texture
    [Toggle] _ShowDecal ("Show Decal?", Float) = 0 // Toggle to show/hide decal
}

// Fragment shader: blend decal with main texture based on toggle
half4 frag(Varyings IN) : SV_Target
{
    // Sample the main texture
    half4 mainTexColor = SAMPLE_TEXTURE2D(_MainTex, sampler_MainTex, IN.uv);

    // Sample the decal texture
    half4 decalTexColor = SAMPLE_TEXTURE2D(_DecalTex, sampler_DecalTex, IN.uv);

    //Sample the third texture
    half4 thirdTexColor = SAMPLE_TEXTURE2D(_TexThree, sampler_TexThree, IN.uv);

    // If _ShowDecal is 1, blend the decal with the main texture; otherwise, use the main texture only
    half4 finalColor = (_ShowDecal == 1) ? mainTexColor + decalTexColor + thirdTexColor : mainTexColor;

    return finalColor;
}
```



## RIM LIGHTING

My next implementation was a Rim Lighting shader, for the energy ball between the two opponents. I initially wanted to implement a modified water shader to simulate the energy ball moving back and forth between Peter and Retep, but I realized that I needed a shader from the first half of the course so I changed my mind. I implemented the Rim Lighting

shader by taking the base and rim colours, adding them together, and then multiplying them by the rimLighting factor(the power of rimPower and rimFactor, the combination of the view direction and normal).

My addition to this shader was to add a second rim colour in order to better replicate the energy ball in the reference image. At the end of the script, I took the previous finalColour and added it to the second rim colour(multiplied again by rimLighting) in order to create a neat gradient effect on the energy ball, and then I played around with colours and the rimPower slider to create this effect that looks cooler.

```
Properties
{
    _BaseColor ("Base Color", Color) = (1, 1, 1, 1)
    _RimColor ("Rim Color", Color) = (0, 0.5, 0.5, 1)
    _RimPower ("Rim Power", Range(0.5, 8.0)) = 3.0

    _RimColor2 ("Rim Color 2", Color) = (0, 0.5, 0.5, 1)
}

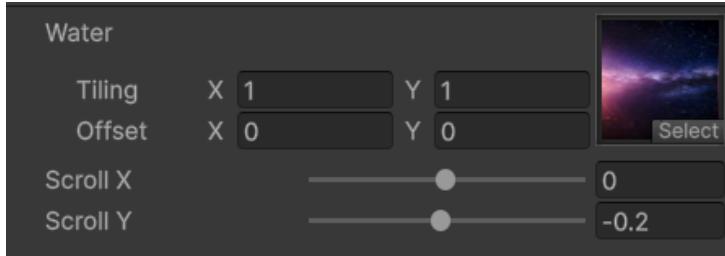
half4 frag(Varyings IN) : SV_Target
{
    half3 normalWS = normalize(IN.normalWS);
    half3 viewDirWS = normalize(IN.viewDirWS);
    half rimFactor = 1.0 - saturate(dot(viewDirWS, normalWS));
    half rimLighting = pow(rimFactor, _RimPower);
    half3 semiFinalColor = _BaseColor.rgb + _RimColor.rgb * rimLighting;
    half3 finalColor = semiFinalColor + _RimColor2.rgb * rimLighting;
    return half4(finalColor, _BaseColor.a);
}
```



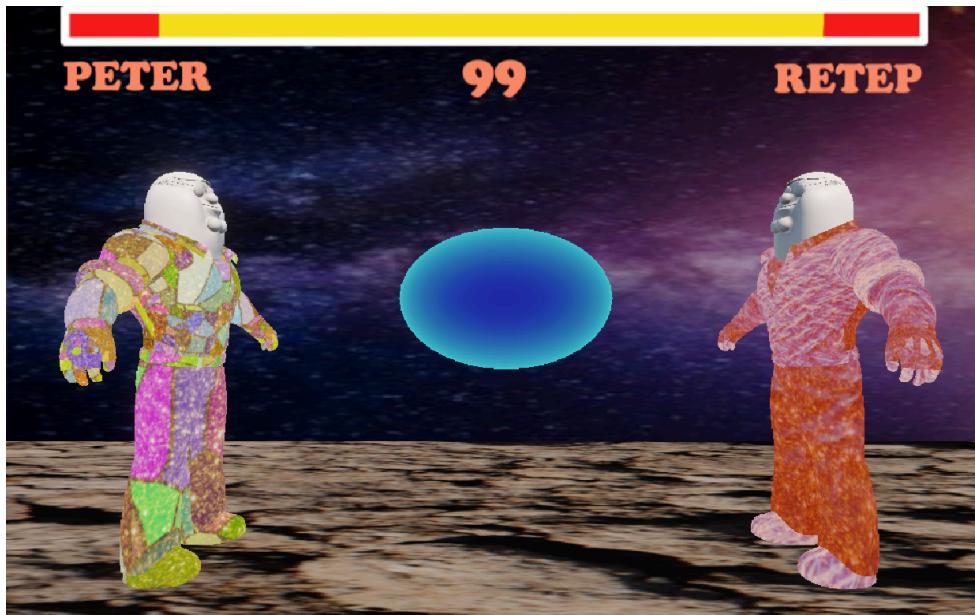
SKY SCROLL

I modified the Water Scroll shader given on Canvas to create a scrolling sky texture for the background of the scene. I wanted a cool and unique background that added some movement to the scene, and the scrolling shader was the perfect way to implement that. To do this, I took out the unneeded Foam Texture code from the shader script, as it would only add an unnecessary overlay of white that would make the shader look worse. Then, I locked the X-scroll factor to 0 so the sky was only scrolling vertically, and then I set the Y-scroll value to -0.2 to get the sky texture to scroll downwards.

```
Properties
{
    _MainTex ("Water", 2D) = "white" {}
    //_FoamTex ("Foam", 2D) = "white" {}
    _ScrollX ("Scroll X", Range(-5,5)) = 1
    _ScrollY ("Scroll Y", Range(-5,5)) = 1
}
```



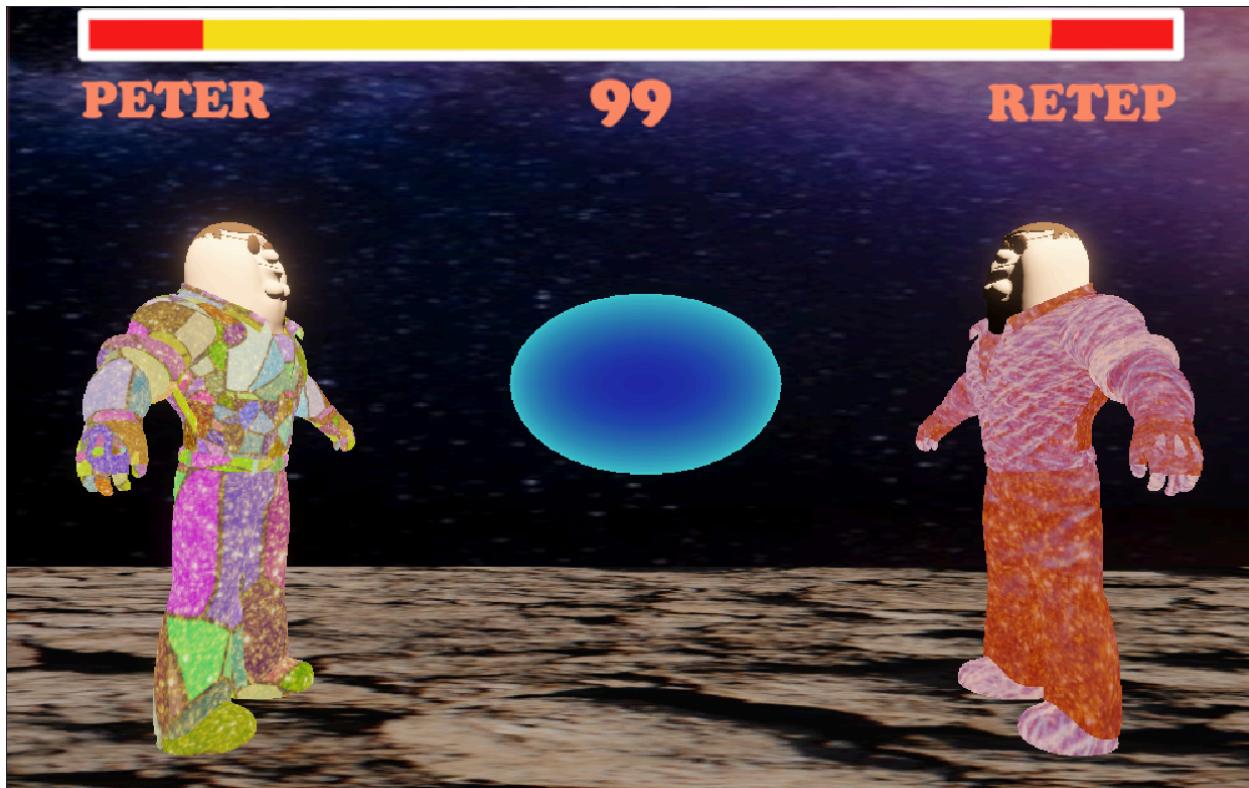
(Please view the project itself to see the sky scrolling properly, I do not have time to make a GIF, sorry!)



## SHADOWS

Finally, I implemented a simple lighting shadow to better highlight the good vs evil nature of the two characters. I wanted to keep the skin and hair parts of the model textureless in order to prevent the scene from looking too busy, so keeping the shadows simple felt best. I wanted to use the lighting to demonstrate which character was good and which was evil, as with the way the models were rotated it allowed me to do so with ease. Peter's face is shadowless, to signify his pure and noble heart, while Retep's face is cast in shadow to showcase how he has dark thoughts and is evil. For my adjustment, I noticed that the shadow shader made the faces of Peter and Retep feel flat, so I went in and added a x3 multiplier to the diffuseLight factor in order to make their faces glow a bit more. It adds an extra light source to the scene, as well as draws your attention to their faces so you can notice the environmental storytelling with them.

```
// Lambertian lighting calculation
half NdotL = max(0.0, dot(i.worldNormal, lightDirWS));
half3 diffuseLight = NdotL * mainLight.color * 3;
```



I ran out of time to do the shadows shader diagram, unfortunately! Sorry about that.

## ISSUES WHEN BUILDING GAME



I'm unsure what causes this, as the shaders all work fine when running in unity. Please view the game in unity in order to properly judge the shaders.