***Shaders***

*Toon & Water*

Date: 08.09.2021

Name: Luca Ruiters

Student nr: 500796991

GitHub: [ru1t3rl/GPE](https://github.com/ru1t3rl/GPE)

Table of Contents

[Intro 3](#_Toc82605804)

[Toon Shader 3](#_Toc82605805)

[Diffuse 3](#_Toc82605806)

[Specular Highlights 3](#_Toc82605807)

[Water Shader 4](#_Toc82605808)

[Color 4](#_Toc82605809)

[Flow 4](#_Toc82605810)

[Reflection & Refraction 4](#_Toc82605811)

[Post Processing Outline 5](#_Toc82605812)

[Conclusion 5](#_Toc82605813)

[References 6](#_Toc82605814)

Notes and sketches can be found in the *Files* chapter

# Intro

I really like the effect shaders can have on games. Previous semester, during the VR minor I started experimenting with Shader Graph and enjoyed it a lot. This experience will help with writing normal shaders. Which I have never done before.

# Toon Shader

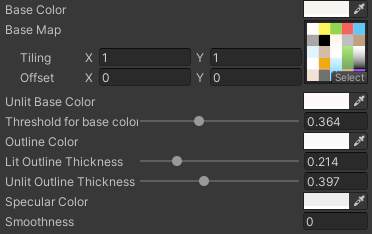
Before I started, I made of list of properties of a toon shader. For example, you need to be able to change the shadow color, the number of shadow “rings” and a custom specular. Eventually, I came across this wiki page which had a very explanation on how to make this effect (Wikibooks, 2020).

Figure 1 - Toon Shading in action (With support for multiple light source)

## Diffuse

At the beginning of the shader, I first calculate the light direction based on the “Main” directional light’s position. The light direction will later be used for specular highlights and shadows.

Figure 2 - Toon Shader Inspector

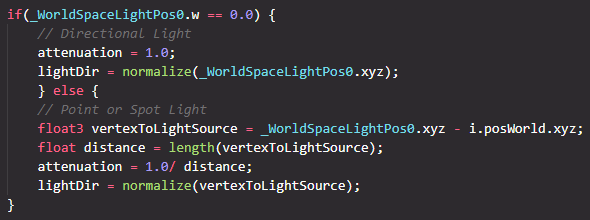


Figure 3 - Calculate the light direction and attenuation base on the "Main" light's position

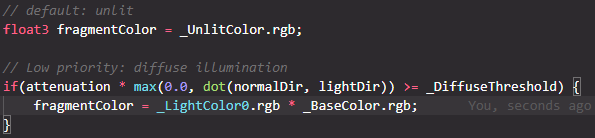
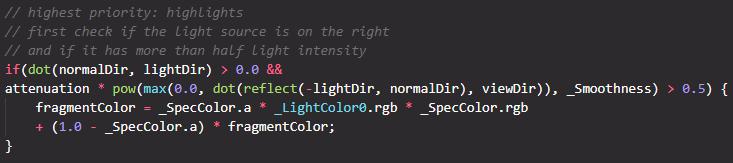
To get the correct diffuse color, I multiply the “Main” light’s color with the base color.

Figure 4 - Calculate the fragments color based on the light color and base color

## Specular Highlights

Calculate the specular color base on the light color and out specular tint. The alpha of the specular tint will determine the intensity of the specularity. In a later version of the shader, I have added a variable called “\_LightColorInfluence”. This variable will change the influence the light color has on all the calculations. The shader also contains an outline option, most of the time I use this as an extra contrast instead of an outline. To support multiple light sources, I transferred the shader code to a *.cginc* file and include that in both the base and add pass.

# Water Shader

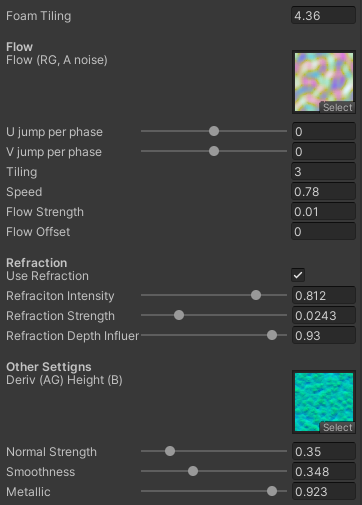
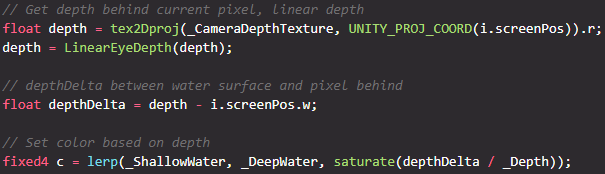
The water shader consists out of multiple parts, a color/gradient, reflection, foam, flow and refraction part. I have used a couple of sources to reach the result. I will link them below and throughout the text.

Figure 5 - Water shader inspector

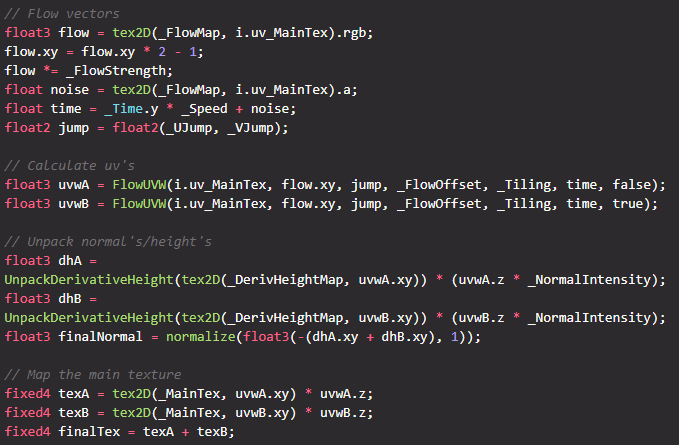
## Color

I have chosen to use two separate colors, one for deep water and the other for shallow water. I will lerp between those colors using a depth value. To get this depth value, I use the CameraDepthTexture and use it to calculate the distance between the depth and the current vertex position.



## Flow

Figure 6 - Calculate UV's based on a flow vector and time



## Reflection & Refraction

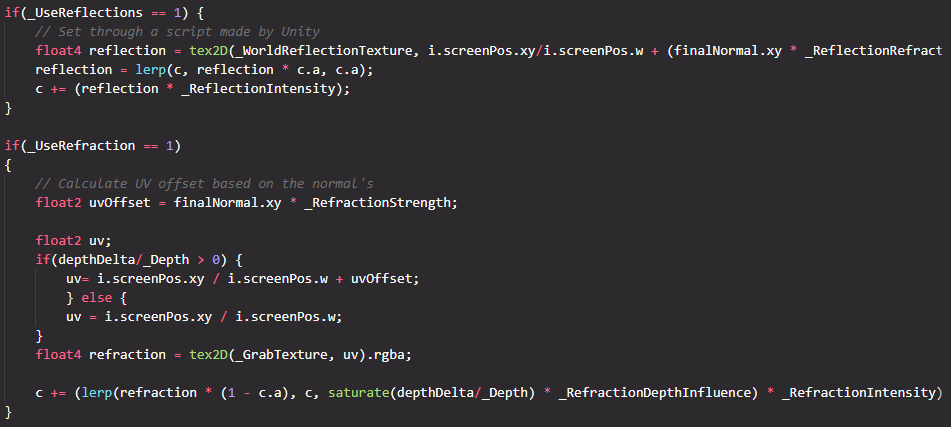
The reflection and refraction are set using the “finalNormal” offset. Doing so will make sure the refraction moves equal to the ripples/waves.

Figure 7 - Reflection and Refraction

Figure 8 - Linked the displaced UV to the main texture and normal map

# Post Processing Outline

Since I made the toon shader, I figured it would be nice to have some form of an outline effect. To do this I decided to go with an image effect (3x3 kernel). This was partially based on an example Daniël showed us.

To get started, I opened the Wikipedia page about the image kernels and used that in combination with a tutorial to make the image effect (Ilett, 2019).

Using this effect will online show the edges of your scene. I wrote this simple formula to get the color back.



# Conclusion

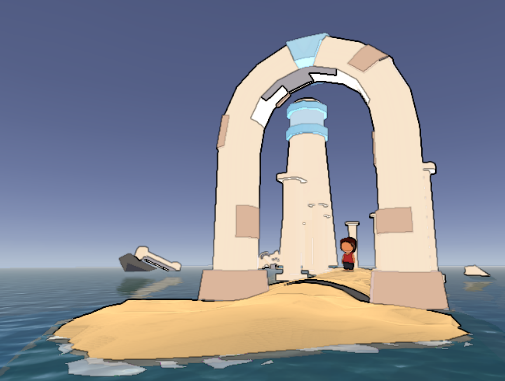
I first started the water shader as a surface shader. This mainly had to do with my method for reflections doing weird stuff in a vertex shader. But during development and research I found another method to do reflections which worked better and in a converted version of the shader (surface to vertex).

Figure 9 - Black Outline

Figure 10 - White Outline

Figure 11 - Sobel kernel transformation

# References

A. (2017, August 14). *Unity Refractive Shader*. Creative Tinkering. <http://tinkering.ee/unity/asset-unity-refractive-shader/>

Alisavakis, H. (2020). *My take on shaders: Stylized water shader – Harry Alisavakis*. H. Alisavakis. <https://halisavakis.com/my-take-on-shaders-stylized-water-shader/>

*Cg Programming/Unity/Toon Shading - Wikibooks, open books for an open world*. (2020, April 16). Wikibooks. <https://en.wikibooks.org/wiki/Cg_Programming/Unity/Toon_Shading>

Flick, J. (2018a, May 30). *Texture Distortion*. Cat Like Coding. <https://catlikecoding.com/unity/tutorials/flow/texture-distortion/>

Flick, J. (2018b, August 30). *Looking Through Water*. Cat Like Coding. <https://catlikecoding.com/unity/tutorials/flow/looking-through-water/>

*How do I calculate UV space from world space in the fragment shader?* (2016, August 29). Game Development Stack Exchange. <https://gamedev.stackexchange.com/questions/129139/how-do-i-calculate-uv-space-from-world-space-in-the-fragment-shader>

*How to calculate screen space texcoords for surface shader? - Unity Answers*. (2017, September 3). Unity Form. <https://answers.unity.com/questions/1402029/how-to-calculate-screen-space-texcoords-for-surfac.html>

Ilett, D. (2019, May 11). *Image Effects | Part 4 - Edgy Talk*. Daniel Ilett: Games | Shaders | Tutorials. <https://danielilett.com/2019-05-11-tut1-4-smo-edge-detect/>