Introduction

Creating a more interactive environment was the goal of the Tech team coming into our final semester working on the game. I was task with finding ways to make the environment more responsive to the player actions and one of the areas I chose to make interactable was the sand landscape more specifically the material that is used on the landscape. I researched and found that there were two methods of achieving what I set out to accomplish Render Targets or Runtime Virtual Textures (RVT). The render targets method had one glaring drawback, lighting is not supported when using them as lighting is a huge part of the game I went with the latter. Below I delve into the architecture of the interactive landscape system as well as how I expose different parts of the system to let other disciplines tweak the results of interaction.

Architecture Overview

Before I started implementing, I had to ask myself what actions from the player would warrant a response from the landscape they were only two, swimming close to the sea floor and using the Aquavac on it. Once I came up with the interaction cases I then looked at their corresponding responses the player swimming close to the sea floor would cause the sand on the sea floor to slowly part leaving behind a trail. The Aquavac sucking on the landscape would cause a non-uniform circular shape to be created and the longer the player sucks the landscape the bigger the depression. In this document I go over the design of the first interaction case.

Landscape Deformation System

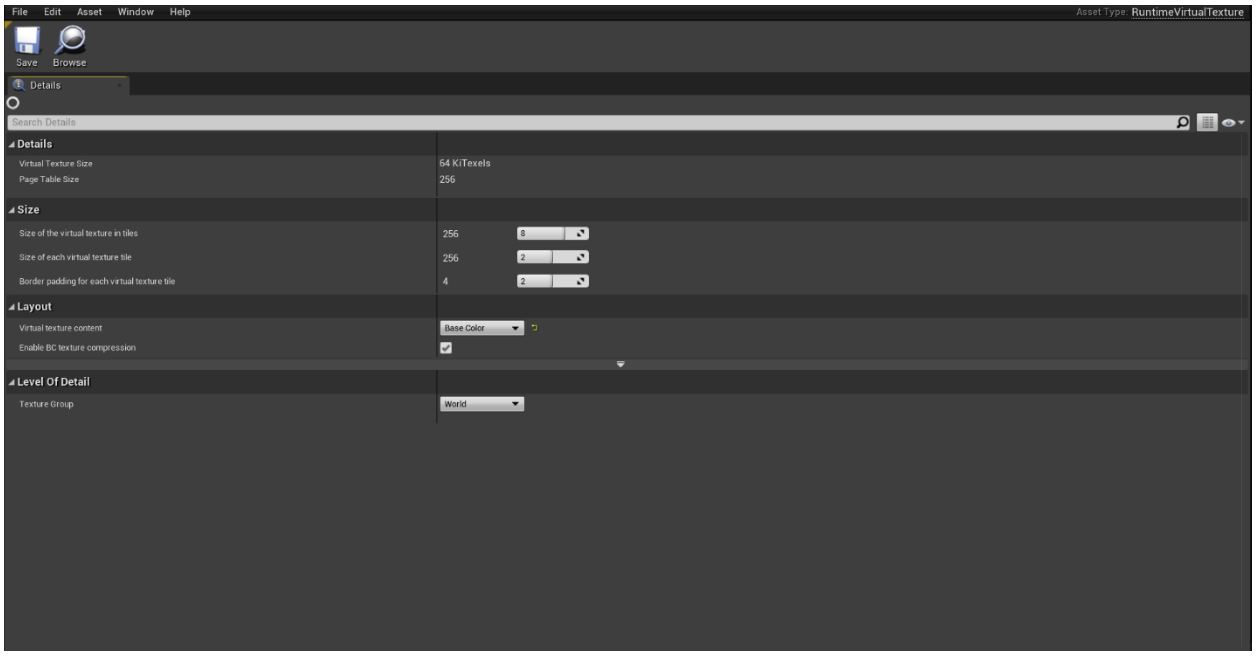
The system has four main parts:

1. Runtime Virtual Texture Object
2. Runtime Virtual Texture Volume
3. Landscape Material
4. Material Defining how to deform the landscape

Runtime Virtual Texture Object

The Runtime Virtual Texture object is a built in Unreal 4 data container (See Figure 1) that goes onto the Runtime Virtual Texture Volume and other materials that need to know about the RVT. This holds data explaining how a virtual texture tile should be drawn in a Runtime Virtual Texture volume.

Figure 1. The default RVT settings were used



Details

Virtual Texture Size	64 KiTexels
Page Table Size	256

Size

Size of the virtual texture in tiles	256	8
Size of each virtual texture tile	256	2
Border padding for each virtual texture tile	4	2

Layout

Virtual texture content	Base Color
Enable BC texture compression	✓

Level Of Detail

Texture Group	World

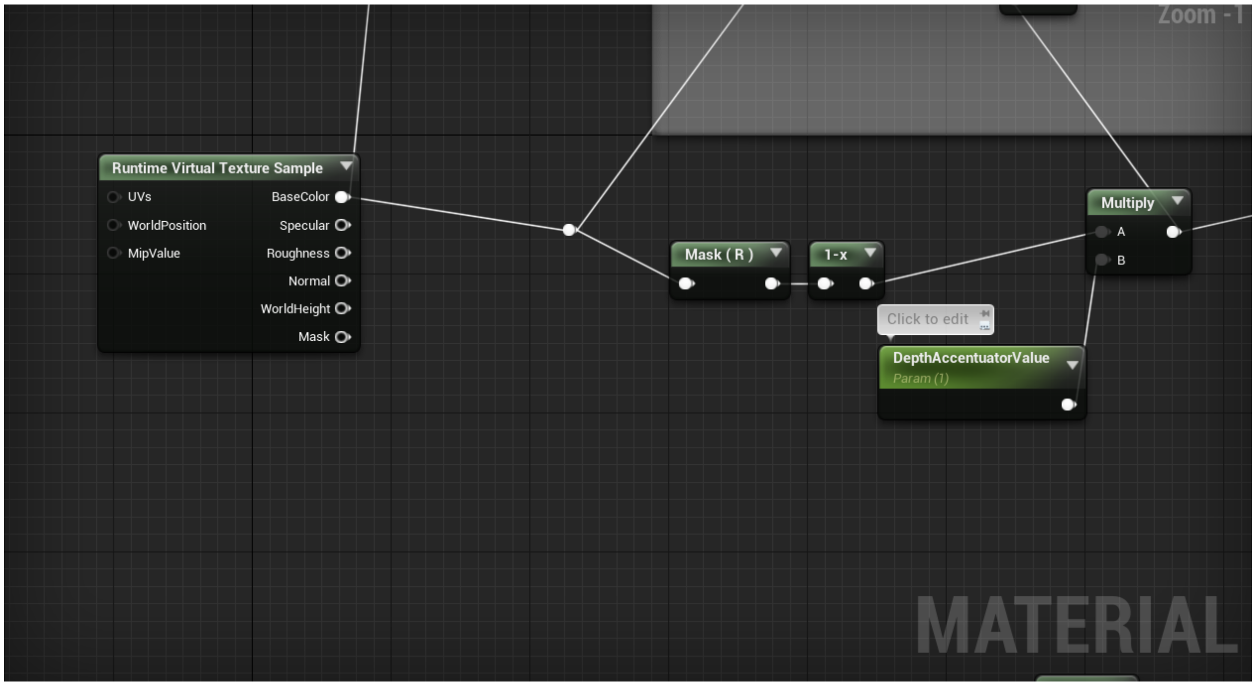
Runtime Virtual Texture Volume

The RVT volume is a special actor in Unreal that encompasses the landscape you want to interact with. This actor allows the material on the landscape to receive a runtime virtual texture as input by sending this information through a RVT sample node.

Landscape Material

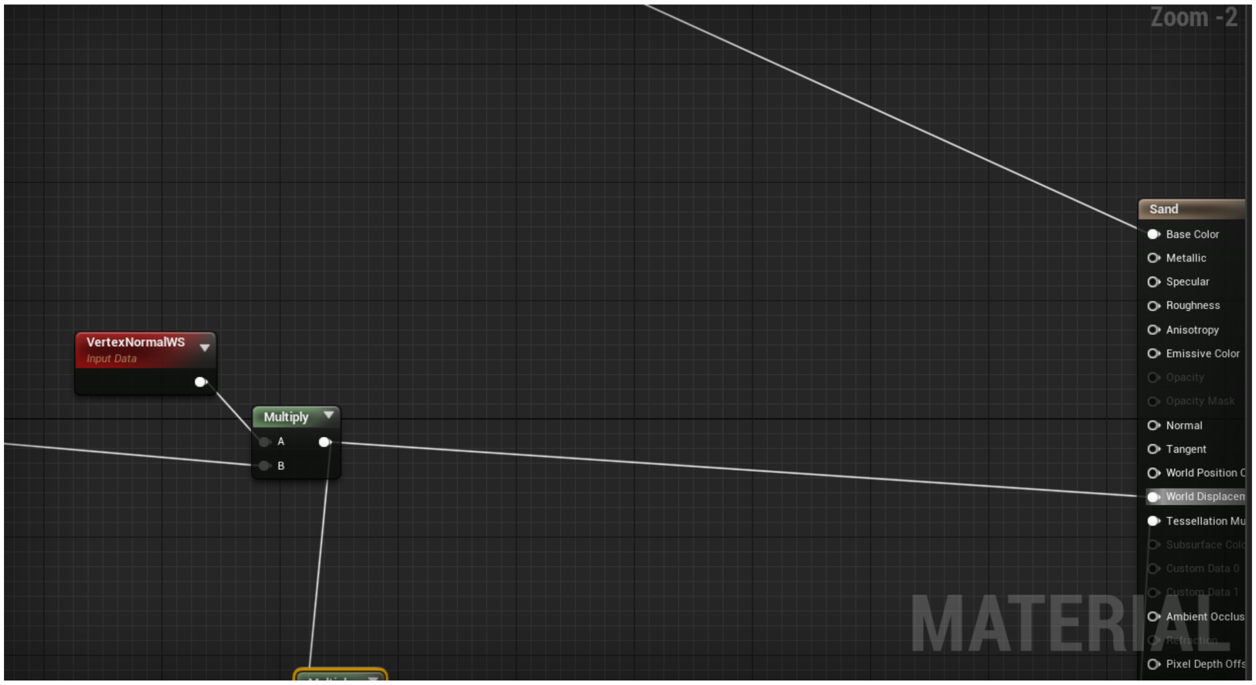
I used the already existing material on the landscape and modified it to receive the RVT information such as Base color and Opacity and I use this information to tell the landscape how it should deform.

Figure 2. RVT Input



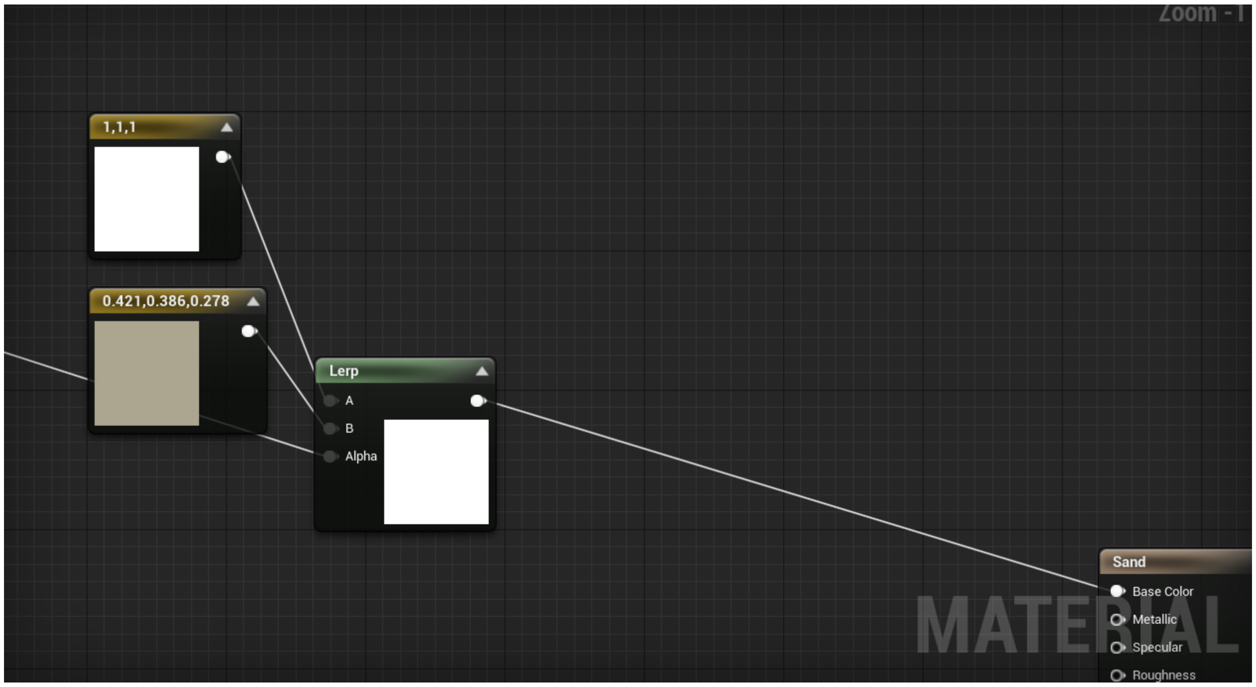
When the RVT gets spawned on the landscape the data from it is passed to this node (Runtime Virtual Texture Sample). I then take the red component of the texture’s BaseColor invert it so it can be used for depth. The value is then multiplied by a depth Accentuator value which is a scalar used to accentuate the depth of the trail. I expose this variable so artist and designers can tune the depth to their liking.

Figure 3. Translating Red Component to Depth



Once I have the inverted red component of the texture, I then multiply it to the vertex normal on the landscape which defines the direction of the deformation. Since I inverted the red component, this direction is downward. This data then goes into the World Displacement node on the material which tells Unreal how much to deform the vertices on the landscape. To make sure the vertices on the landscape become deformed I had to enable the tessellation on the material and define a max tessellation.

Figure 4. How the color of the trail is determined

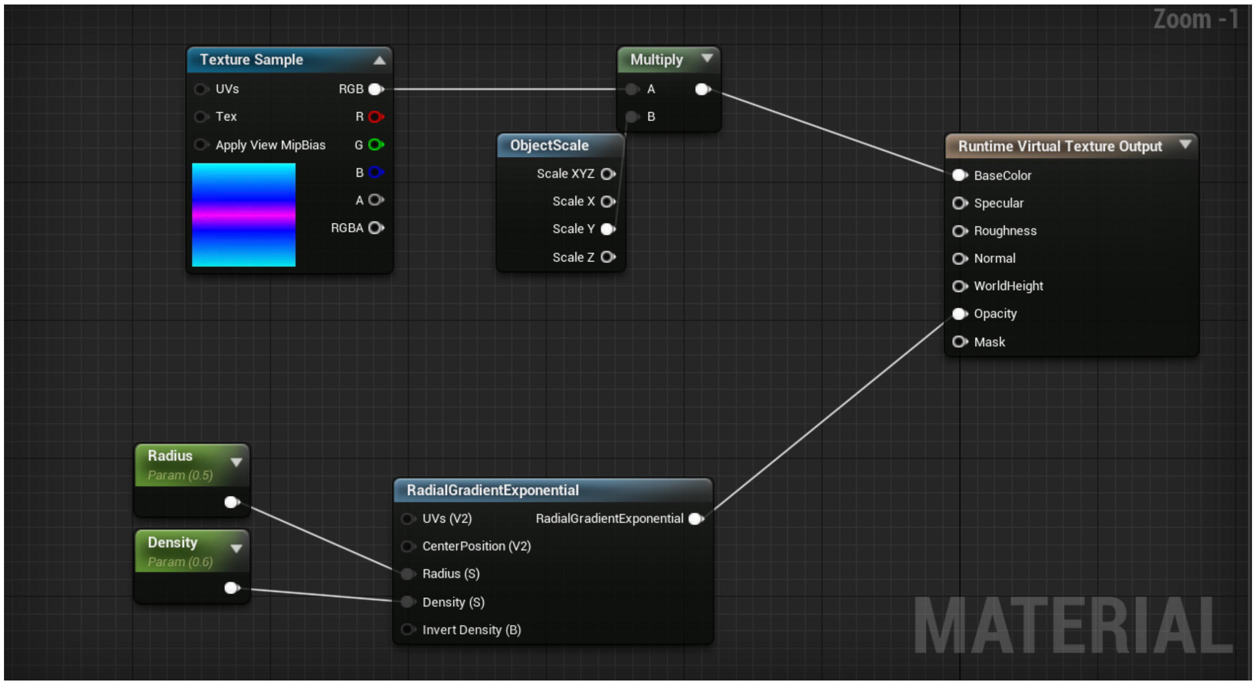


These two colors are interpolated between depending on the distance the player is from the vertices of the landscape. The further away a vertex is the darker it appears the closer it is the more it blends with the color of the undisturbed sand. I expose these two colors so artist can make sure the colors of the sand come out just right.

<u>Material defining how to deform the landscape</u>

This material houses a texture and uses a RVT output node to send information to the landscape material about the chosen texture. This information is then received in the landscape material, which is the material that decides how the landscape should deformed based on the given information.

Figure 6. Material that runs it all



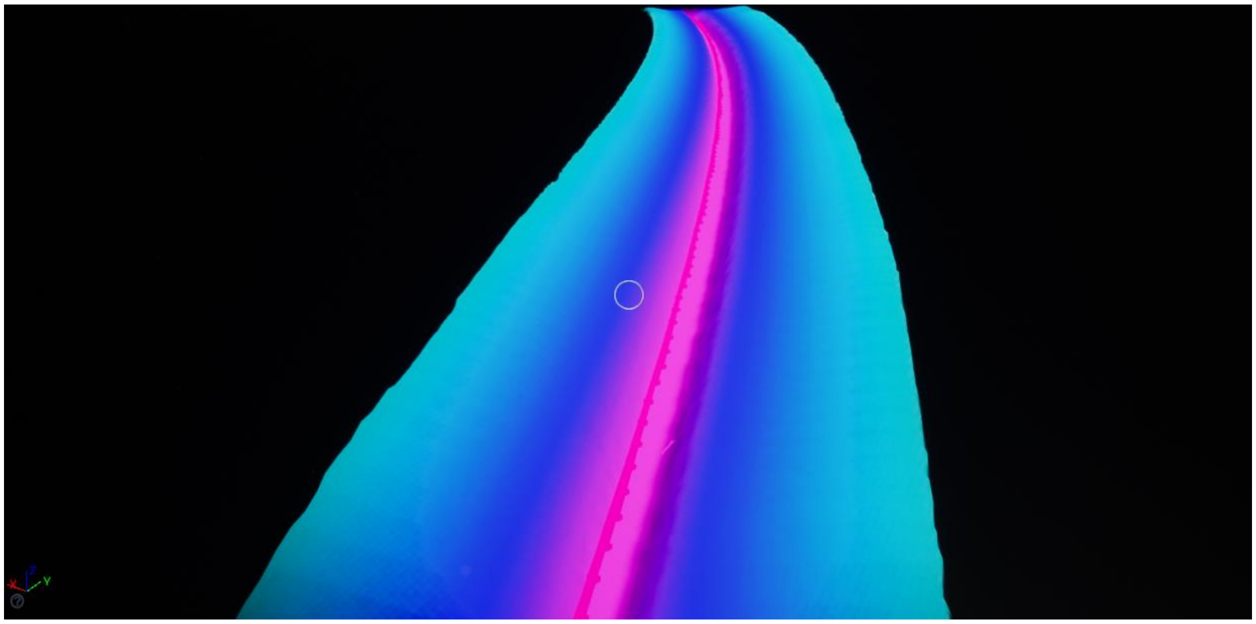
The texture sample node you see in this photo is how I defined the trail to look. I take the color of the texture and scale it up depending on the scale of the object it is placed on (in my case a plane). This is so that the deformation of the trail stays relatively uniform to the object’s scale. The radius & density values dictate the center of the texture and how sharp the transitions of colors in the texture should be respectively. The texture sample, radius & density properities have all been parameterize for other disciplines to fine tune the look of the trail.

I took this material and placed it on a plane and made it invisible in game so I could outsource the scaling of the trail to a separate object. This separate object uses an Unreal timeline to make the trail grow and shrink over time which animates the trail. This also made it easier for other disciplines to go in and change the behaviour of the trail just by adjusting a float graph.

Outcome

In the end the results achieved with this system looked phenomenal and was also done without adding any additional work to the artists' plate. Once this system was implemented it was trivial to expand upon this for the Aquavac-sand interaction.

Figure 6-7. Showing how the trail looks with the texture colors applied



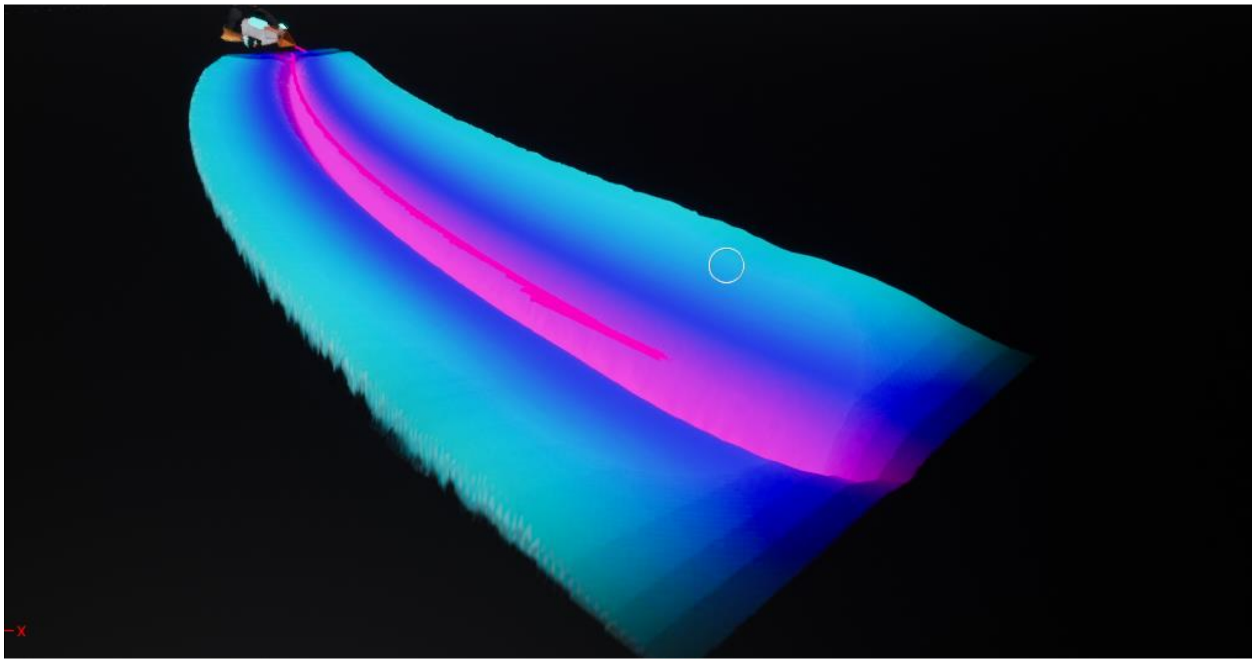


Figure 8. Shows the trail in game

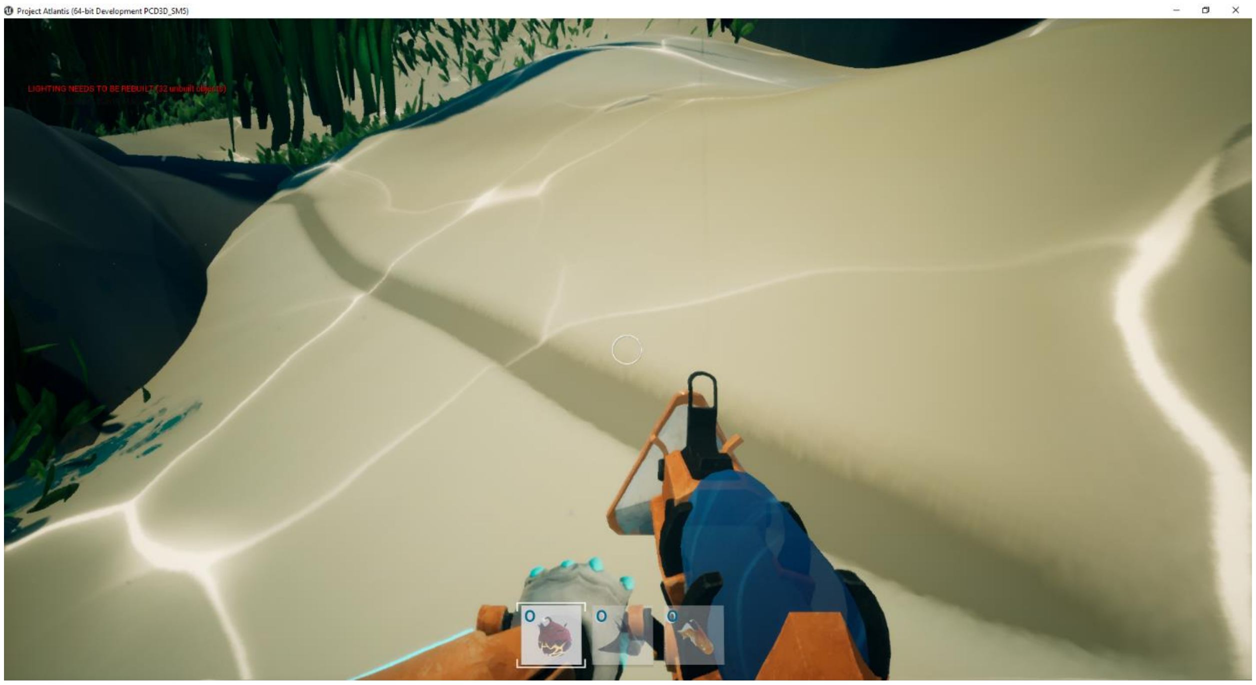
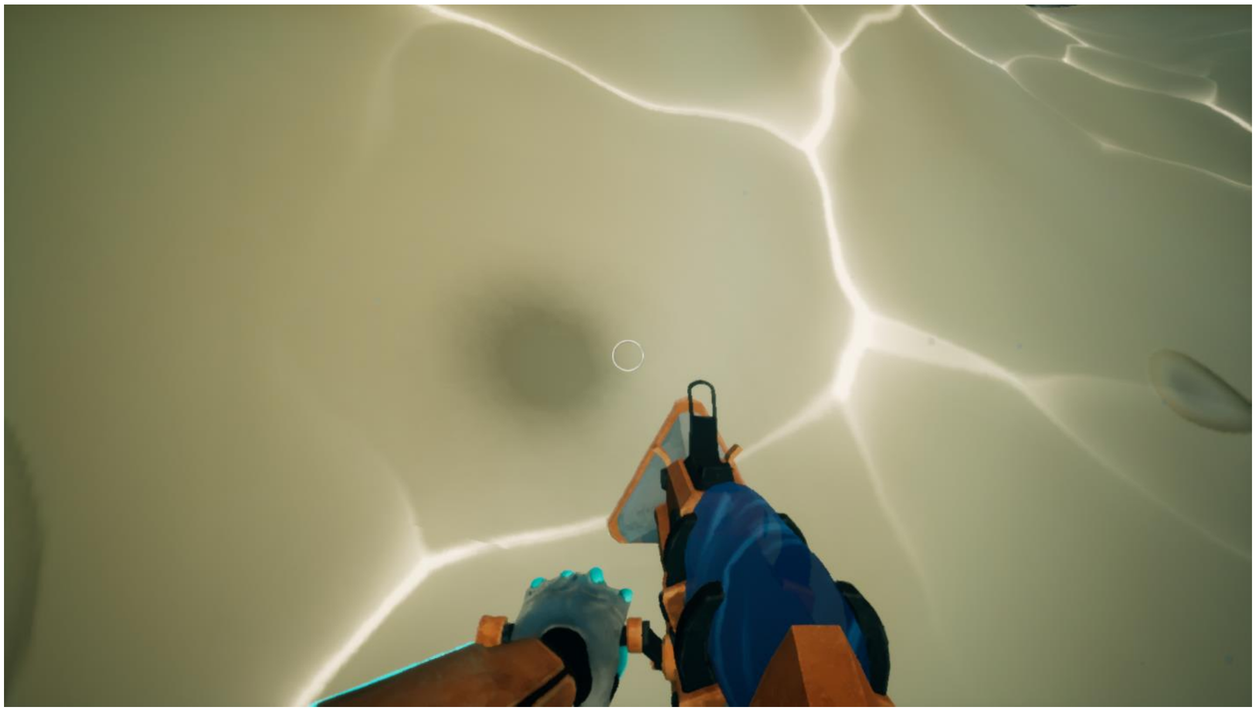

Project Atlantis (64-bit Development PCD3D_SM5)
LIGHTING NEEDS TO BE REBUILT (32 unbuilt objects)

Figure 9. Shows the deformity made by the Aqauvac



Done By: Deante James