**Advanced Graphics Research Project Report**

For this research project I decided to research a couple of rendering techniques and use them to generate water effects. The specific techniques I looked into were animated transparent textures using texture transformations with blending techniques and reflections using the stencil buffer.

**Animated Water Textures**

Initially I implemented the animated texture by translating a single diffuse texture across a plane mesh by passing a matrix representing the translation to the vertex shader and transforming the texture coordinates by this matrix. This resulted in a bland and unrealistic representation of moving water. Water in reality does not have any texture instead the light that is reflected from the surrounding scene and the light that we can see through the transparency of the water is the colour that is perceived by a viewer. In order to generate the movement of water I needed to create waves, more importantly I needed to translate these waves. First I generated a wave by adjusting the height of the vertices based on time in the vertex shader using the wave formula as follows:

Where is the amplitude, is the direction vector, is the frequency (), is the time constant and is the phase-constant ().

Using this formula it was easy to generate a simple single wave. A summation of the same formula with different parameters provides a more complex effect with multiple waves interfering with one another. Such an effect is necessary to generate a realistic wave simulation; however an ocean typically has thousands of waves of different amplitudes, wave lengths and speeds, thus making it impractical to effectively generate an ocean effect at real-time frame rates. In order to subvert this problem I found it much more computationally practical to use normal maps to generate wave effects. Using the normal maps to manipulate the lighting calculation in the pixel shader instead of performing geometry manipulation in the vertex shader is vastly cheaper and provides much more detail to the waves. Running just one normal map across the water does provide a wave effect, but it’s too obvious to the user that a texture is moving across a plane as the player can too easily focus on it. I felt it lacked an animated feel, thus initially to rectify this issue I thought to use a series of the normal maps swapping them frame by frame as you would with a typical animated image. However, this would require approximately a hundred or so images to create a smooth animation and probably an artist at hand to generate them. Therefore I decided to take an approach similar to the summation of waves above and use single normal map but transform it in four ways to produce four normals and combine them together.

**Water Reflections**

To do the reflection I first tried using the stencil buffer method. This involved rendering the entire scene normally (excluding the water, which is rendered only to the stencil buffer). Then rendering the reflected scene (excluding the water again), by multiplying all world transforms of the objects in the scene by a matrix that describes a reflection through the water plane, only the stencilled out region. Lastly, I rendered the water using alpha blending, by setting the colour blending states to use the alpha for the source colour blend factor and the inverse alpha for the destination colour blend factor, to have the reflection show up through the water. The result is a nice reflection on the water:



Figure 1 Stencil Buffer Reflections

However, with this approach I struck a few issues. The first of which is that the reflection entirely masks what’s below the water’s surface, thus there’s no way to see through the water, ultimately making the water lack realism near the coastal where you’d expect to see the beach/sand through the water. The second issue comes in regards to the fact that the back face culling has to flipped when rendering the reflected scene. In some places at certain camera angles the back faces show through because they are facing the camera, but are not facing the reflected camera.

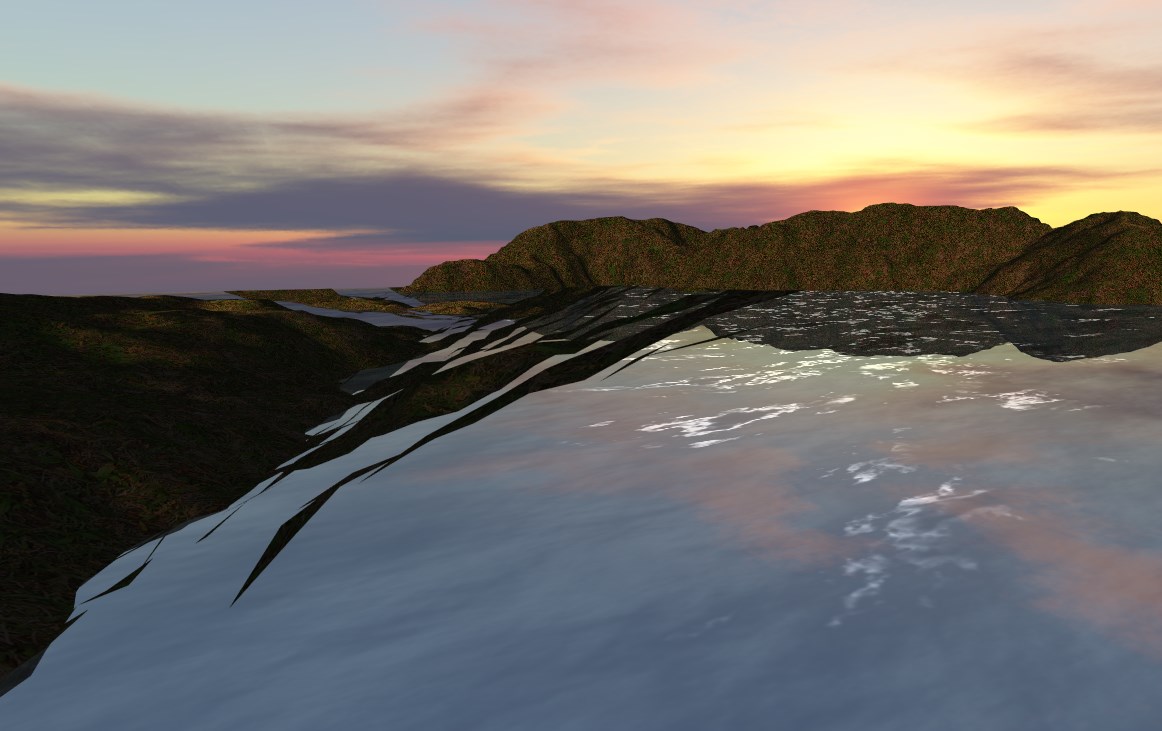


Figure 2 Back Faces

The third and final problem with this approach is that there’s no way to distort the reflection with regards to the waves. Without this distortion the waves and the reflection completely look unrelated, which detracts from the realism of the water.

To solve these issue I decided to render the reflection to a texture. This was mainly for the distortion effect as the other two issues are solvable through using the stencil buffer method, however they both have nicer and more effective solutions through the render to texture method.

**Water Refraction and Reflection**

To do the refraction I rendered the entire scene to a texture (refraction map) from the view of the camera, excluding the water.

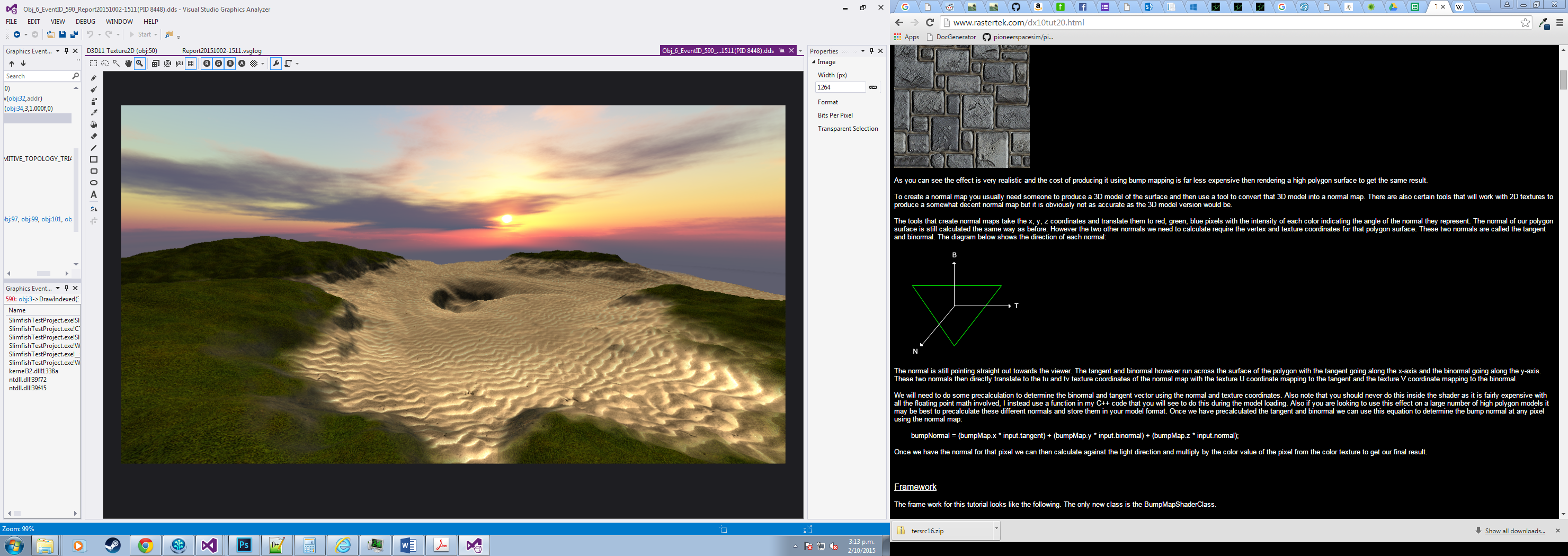


Figure 3 Refraction Map

Then I rendered the water by giving it the refraction map as a texture resource view in the pixel shader. I used the screen space coordinates of the pixels of the water to sample from the refraction map, as well as pushing theses coordinates around by the normals of the water sampled from the normal map to provide the distortion effect. The reflection is done in a very similar way, except a special view transform is used when creating the reflection map that reflects the camera about the water plane. Additionally a clipping plane is used to clip all the geometry below the water when rendering to the reflection map to avoid seeing this geometry in front of the reflected geometry we actually want to see. This special view matrix is then used in the water vertex shader to generate a reflected texture coordinate on the water to use when sampling from the reflection map, by transforming the water vertex by the reflected view transform and standard projection matrix.

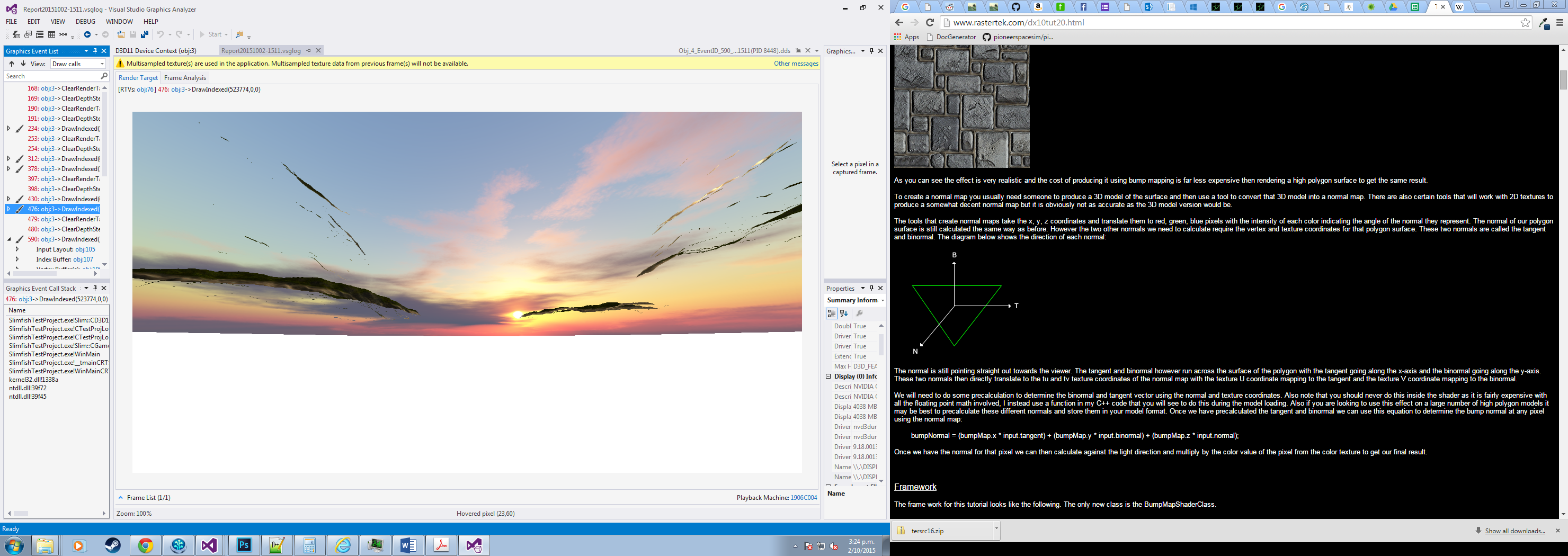


Figure 4 Reflection Map

These map are then combined within the water shader using a fresnel term to blend between them. The fresnel term is found using the fresnel equation, which determines how much light gets reflected and how much light is refracted.

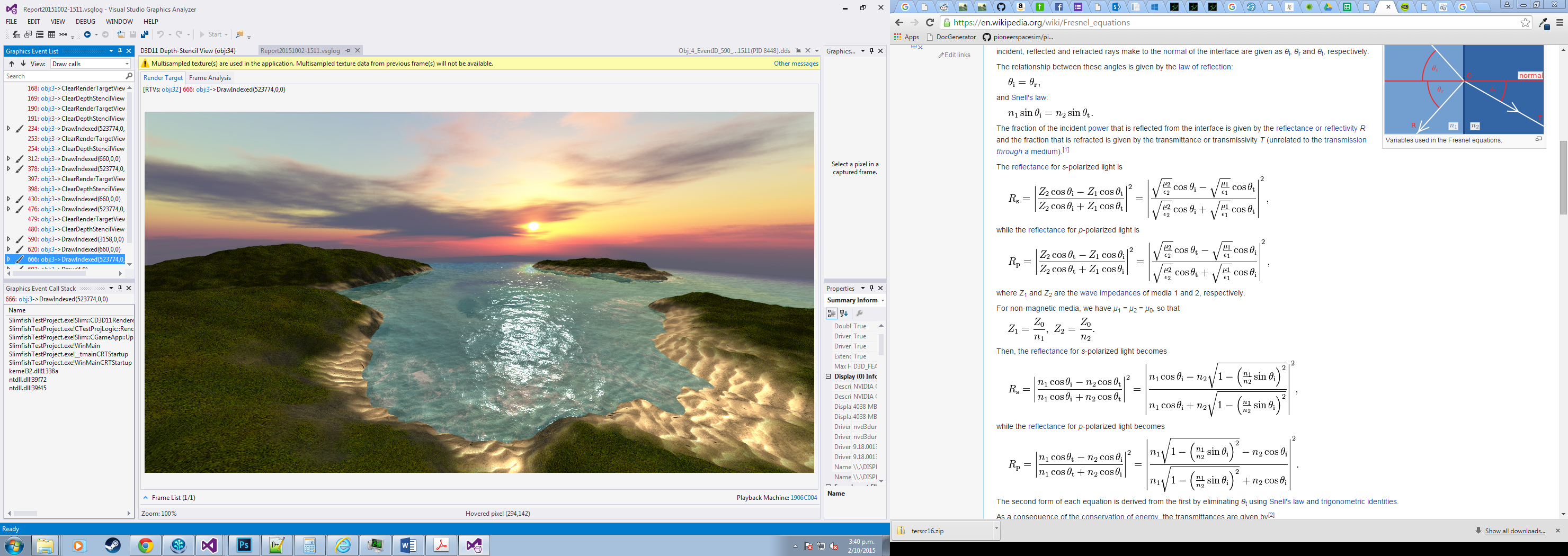


Figure 5 Final Combined Image