Advanced Shaders

Karan Sharma, Adam Harris & Albert Gonzales

Grand Canyon University

05 December 2021

Advanced Shaders

In this two-part project (Projects 9 and 10), you will learn how to write shaders to produce several different mapping effects, in preparation for writing more advanced shader code. In addition, you will implement camera movement and object inspection controls.

# Theoretical Background

For the following project the theory used is Shaders. Computer graphics, especially 3D graphics, use many kinds of information to construct the visuals meshes, textures, lights and so on. That information passes to the graphics hardware, which then processes the image and displays it onto the screen. Rendering is what programmers call the process of generating an image, and Shaders are short programs that render graphics data. That is, a Shader is a program that takes meshes, textures etc. as the input and generates an image as the output.

## Main Idea

The main idea behind the project is using shaders to render the illusion of a current of air. The project follows a barber blow drying a head of hair and the camera zooms into the head of hair to show it flowing with the air provided.

### Desired Effect.

The desired effect is to the blades of hair to move in a motion, so it looks as if it is blowing due to the blow dryer.

#### Scene Layout

The scene for the following project involves a kid comes into the barber show to get a haircut. Once the haircut is done it is now time to blow dry the hair during the blow dry session the hair moves in accordance with the air from the blow dryer.

**Basic Shader**

The shader used for the following project is the wind blowing shader. The following shader works by moving the mesh ground. Moving the mesh causes all prefabs to appear to be flowing in the wind creating the illusion of wind blowing in the wind.

##### **Implementation**

To successfully implement the shaders to the project there were several methods used:

**C# Code**

Generate Terrain: To generate the terrain we took the initial size of the terrain which in our current case was set to 64. Using a double for-loop we were able to add a height map per pixel and set the color of our terrain to a greyscale. If the values, I and j were 0 it means the creation of the primary pixels of the mesh have not happen yet and the program would complete that task first. Next, once the primary portions of the mesh have been created all other parts of the terrain are created.

Generate Field: To generate the field would mean to add the “blade of grass” prefab to every patch of terrain on the mesh. Currently the number of patches per row assigned are 30 and the amount of grass to be added to each patch of terrain is 50. With these initials we calculate the size of each patch (Which would be uniform for each patch), the start position for the creation of the first patch which will update as we move further and a renderer that generates the grass mat mesh to create the grass on each patch of the terrain.

**Wind Shader**

For the wind shader we applied 3 wind functions in the x and y direction. The x and y directions have the sin and cos function applied to them respectively. There is also a central wind function with initial values provided. In the following code we also have variables to control oscillation strength, sinskewceoff, left bound wind and right bound wind. We also split the wind into four quadrants which blows away from its center of origin 45 degree towards its respective directions.

**Camera Code**

There are two parts to the camera code. The first being the camera starts at position 90.4, 71.5 and -23 in space. The position of the camera is then updated along the z axis as the value gets closer to -6.7. Once position z becomes greater than -6.7 the if condition is enabled and using the scene manger the “Close Up” scene is loaded and gets played where we can see the mesh and prefab flowing with the hairdryer. Clicking on the shift the user can speed up the zoom into the hairdryer, using the “spacebar” key the user can restart the code from the start zoom into the hair dryer scene again and clicking the “esc” key the user can escape out of the program.

**Mathematical Concepts**

The mathematical concepts used for the following project were using sin and cosine functions to create wind oscillation. We used vector geometry concepts to dictate the flow of wind and texture geometry to draw the ground mesh required.

Refrences

<https://gist.github.com/McFunkypants/5a9dad582461cb8d9de3>

<https://medium.com/chenjd-xyz/using-the-geometry-shader-in-unity-to-generate-countless-of-grass-on-gpu-4ca6d78b3de6>

<https://halisavakis.com/my-take-on-shaders-geometry-shaders/>

<https://open.gl/geometry>