CS 5963 Assignment 3 Report

Tim Trimble

Austin Anderson

11/2019

**Part 1.1**

Bloom works by taking the image being rendered to the screen, de-rezzing it, and then re calculating it back up to a higher resolution to achieve a “blur” effect. Except, with a bloom effect, instead of only factoring in the position of the objects in order to “bleed” them into other pixels to achieve a blur, the bloom filter takes into account the brightness of the object, and bleeds bright objects into their surroundings with greater precedence than dull objects.

**Part 1.2**

The changes I made to the bloom filter were to allow the blurring (or the pixel “bleed”) to be adjusted manually to bleed more in a horizontal or vertical direction. The method I used to do this can be seen in BloomEffect.cs code. I thought this would be useful in case there was a particular movement or light source causing the bloom, the bloom would most likely be used to reflect a light that wasn’t pointing straight at the objects. This would provide bloom that I would find more represented of brightness in a real life situation. For example, the “bloom” effect would be more intense on an object horizontally if a flashlight were pointing at it from the side.   
Expansion on this variable and it’s usefulness could be done by having the horizontal and vertical values of the bloom bleed adjust dynamically by checking the position of the lights in relation to the object.

**Part 2.1**

The Depth of field code functions by passing a depth from camera parameter to the shader which the depth of field script holds. A blur is added by rendering the image to a smaller resolution and back up again and the blur is smoothed using a round blur average rather than a square one (the Bokeh radius). Finally, to keep the focal point and a certain radius around it in focus, the depth of the current pixel is Linearly Interpolated to determine if the image at that point should be the lower resolution, higher resolution, or an averaged resolution.

**Part 2.2**

The FFR effect is achieved very similarly to part 2.1 by bringing the image to a lower resolution and then back up. For determining which section should be rendered at which resolution, two min and max values for both the x and y axis are given to the shader. The shader then uses these boundaries to lerp the two resolutions based on the pixels current x or y coordinate.

**Part 2.2 Fixed Foveated Rendering++**

The different levels of FFR are achieved by bringing the image down to even lower resolutions, and making more passes for blending using the min and max x,y values. In effect, it is a slightly more built up version of FFR-Low. To simplify debugging, each level of FFR was split to its own shader so that each pass could be more easily managed.