CS6360 Assignment 3

Lauren Lee Isaacs and Monesha Murdeshwar

Keywords: Bloom, blurring, foveated rendering, fixed foveated rendering, dynamic foveated rendering, depth-of-field, bokeh, kernel, and blit.

1. Bloom

The Bloom effect blurs pixels into one another in order to create a brighter looking image [1]. This is accomplished based on brightness and removes the clearly defined edges of objects making them appear to glow.

1.1 Baseline Bloom

Screens are able to display low dynamic range of light which is mapped onto red, green, and blue light hues from 0-1. Displays are limited in the amount of light they can produce, but can read high dynamic range light which surpasses the limit of 1 by **tone mapping** them on to low dynamic range light. Pixels are averaged together to make them appear blurred creating a new shape, but done through a progressive process, or several passes, in order for the image to resemble its original shape. The original shape is then averaged together in an additive way to the new, blurred shape in order to create the bloom effect that is unique and not uniform in blurriness (figure 1 **and 2**)

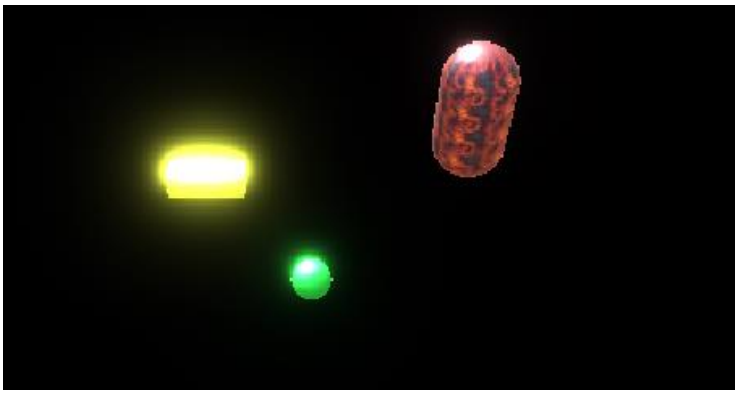

Figure 1: Baseline Bloom on basic texture objects

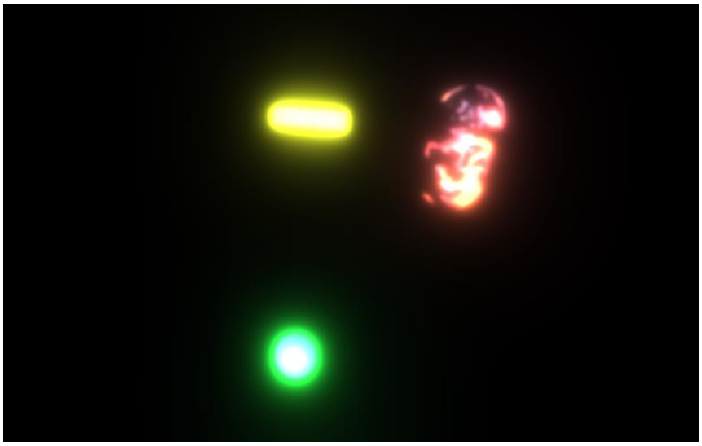

Figure 2: Baseline Bloom on lava texture object

Implementation:

For Our baseline bloom, we first set the scene with 2 capsules (one textured) and one sphere with a black background. The black background was chosen so that the effect of the bloom could be seen better. We created another texture performed progressive downsampling. We notice that it is produces blocky images

to fast as the number of iterations increase. To solve this, we use progressive upsampling. To improve our results even further we created a new custom shader called Bloom Shader. We then created a pass in the shader for the scene. Soft thresholding was performed and bloom intensity adjusted. The final result can be seen in figure 2.

1.2 Modifying Bloom

Implementation:
The modification for the bloom was made by decreasing the blur rate i.e sampling rate by 17 and adding a white tint on the bloom irrespective of the color of the gameobject. The result for the same is illustrated in figure 3.

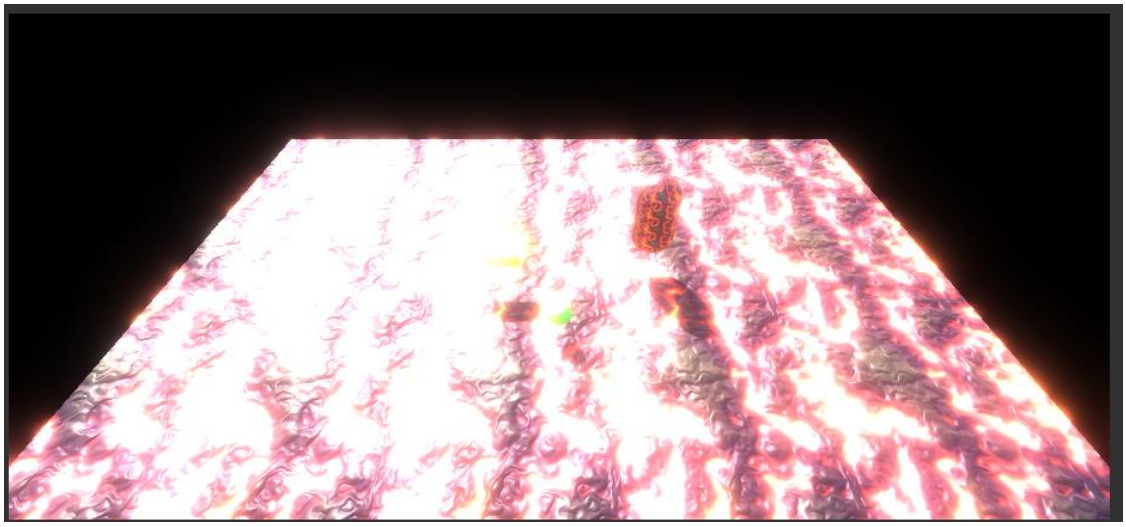

Figure 3: Modified Bloom

2 Foveated Rendering

Foveated rendering makes the image quality in the center of the virtual reality headset lens clear while blurring the peripherals in order to reduce the rendering workload of the hardware.

2.1 Depth-of-Field Rendering

Depth-of-field depends on how far away an object is in relation to the person who is looking at it; objects in the foreground look blurry along with those in the background being unfocused [2]. This is accomplished through creating a projection called the circle of confusion since light in Unity makes objects bright and sharp instead of blurry and unfocused. The circle of confusion directs light through a non-round circle like an aperture to put some objects in focus and others out of focus. This in combinated with a directed bokeh effect, textures, and shaders can create depth-of-field (figure 4). In this we used a similar shader to bloom. Instead of putting the texture for the entire scene we did source.height/2. We create 2 textures for this. One normal and the other the ½ blurred.

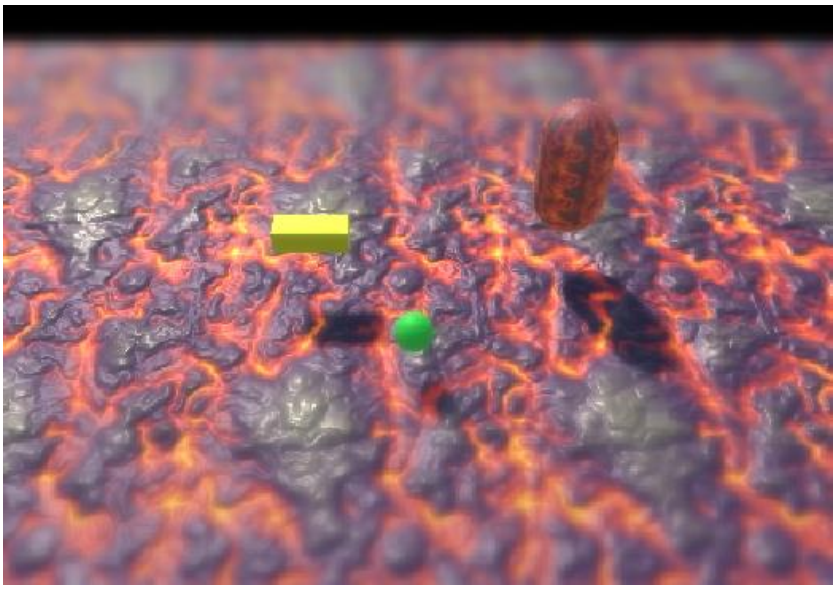

Figure 4: Depth of field Rendering

2.2 Low Fixed Foveated Rendering:

For low Fixed foveated rendering, we created 3 textures, one normal, one blur by ½ and other by ¼. We limited the regions to be assigned to each texture by the following code.

```
{

{
GPROGRAM
   #pragma vertex VertexProgram
   #pragma fragment FragmentProgram

   half4 FragmentProgram(Interpolators i) : SV_Target {
      if(i.pos.x < _MainTex_TexelSize.z/5 && i.pos.y < _MainTex_TexelSize.w / 4) return (tex2D(_SourceTex1, i.uv));
      else if (i.pos.x < _MainTex_TexelSize.z / 5 && i.pos.y > 4*_MainTex_TexelSize.w / 5) return (tex2D(_SourceTex1, i.uv));
      else if (i.pos.x > 4*_MainTex_TexelSize.z / 5 && i.pos.y < _MainTex_TexelSize.w / 4) return (tex2D(_SourceTex1, i.uv));
      else if (i.pos.x > 4*_MainTex_TexelSize.z / 5 && i.pos.y > 4 * _MainTex_TexelSize.w / 5) return (tex2D(_SourceTex1, i.uv));
      else if (i.pos.x > _MainTex_TexelSize.z / 5 && i.pos.x < 4*_MainTex_TexelSize.z / 5 && i.pos.y < _MainTex_TexelSize.w / 4) return (tex2D(_SourceTex, i.uv));
      else if (i.pos.x < _MainTex_TexelSize.z / 5 && i.pos.y > _MainTex_TexelSize.w / 4 && i.pos.y < 4*_MainTex_TexelSize.w / 5) return (tex2D(_SourceTex, i.uv));
      else if (i.pos.x > 4*_MainTex_TexelSize.z / 5 && i.pos.y > _MainTex_TexelSize.w / 4 && i.pos.y < 4 * _MainTex_TexelSize.w / 5) return (tex2D(_SourceTex, i.uv));
      else if (i.pos.x > _MainTex_TexelSize.z / 5 && i.pos.x < 2* _MainTex_TexelSize.z / 5 && i.pos.y > 4*_MainTex_TexelSize.w / 5) return (tex2D(_SourceTex, i.uv));
      else if (i.pos.x > 3*_MainTex_TexelSize.z / 5 && i.pos.x < 4 * _MainTex_TexelSize.z / 5 && i.pos.y > 4 * _MainTex_TexelSize.w / 5) return (tex2D(_SourceTex, i.uv));
      else return tex2D(_MainTex, i.uv);
   }
NDCG
```

Applying and testing, the results for the following can be observed in figure 5. The bluring cannot be correctly observed due to the black background.

Figure 5: Low Fixed Foveated Rendering

3. Thinking Outside the Parameters

The Bloom effected and foveated rendering could be used in multiple ways, including to create empathy.

3.1 Non-Centered Foveated Rendering

Adapted foveated rendering to where the pixels and the clear images were not in the center of the eye, but rather moved to the bottom, top, or one of the sides could simulate the experience of what it is like to have low vision or experience a stroke. This disorientation could cause the user to experience a world in which images and objects are not clear and what it is like to navigate a world that is not as accessible in the visual realm. Conversely, foveated rendering could be adapted and individualized for someone with low vision such as macular degeneration or glaucoma. The foveated rendering could be rendered in the areas of the screen where the user is able to see, therefore not missing any pieces of information in the scene.

References

[1] J. Flick, Bloom Blurring Light Catlike Coding Unity Tutorials Advanced Rendering, 2017. Accessed on: Nov. 21, 2019. [Online].

[2] J. Flick, Depth of Field Bending Light Catlike Coding Unity Tutorials Advanced Rendering, 2017. Accessed on: Nov. 21, 2019. [Online].