Post-Processing and Glow Effects

Cole Faust
UC Santa Cruz
cfaust@ucsc.edu

Talon Baker
UC Santa Cruz
taebaker@ucsc.edu

Andy Long
UC Santa Cruz
andlong@ucsc.edu

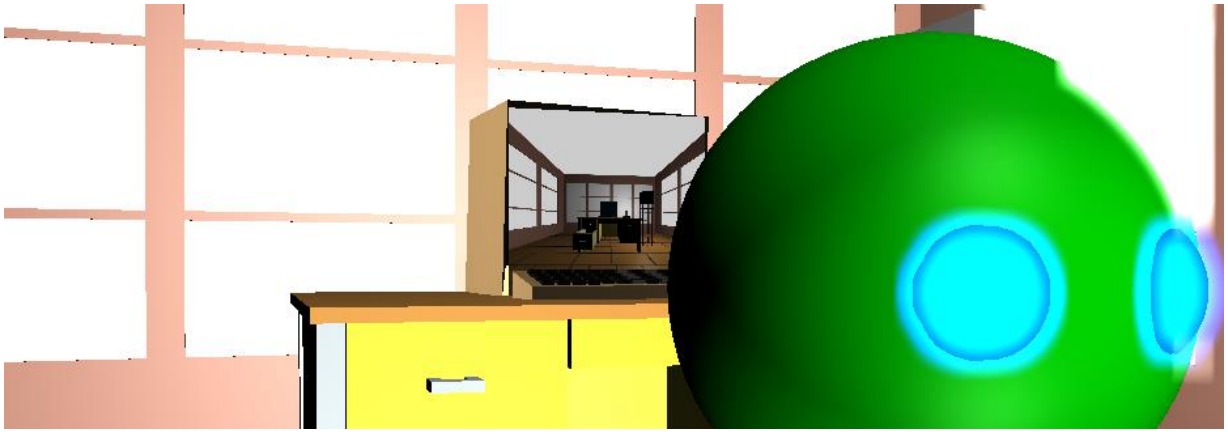

Figure 1: *Scene with Render-to-texture on the computer monitor.*

ABSTRACT

We chose to make our scene a compilation of images and effects so that most of us could work independently but still come together in the end to build a final, complete scene.

1 INTRODUCTION

Our group is Talon, Cole, and Andy. We chose to make our scene a compilation of images and effects so that most of us could work independently but still come together in the end to build a final, complete scene.

2 EXPERIMENTAL AND COMPUTATIONAL DETAILS

2.1 Pixelation Effect - Talon

Whether for artistic purposes or for some kind of censorship, is the process of breaking up an image into large, individual pixels that overlap and obscure the image.

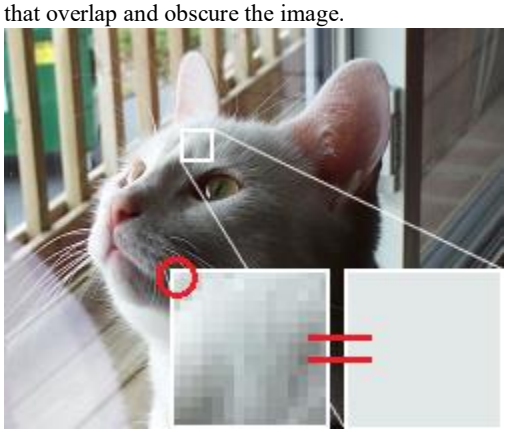

Figure 2. Cat Pixelation Example

The process of pixelation is very straight forward:

1. An image is passed into the Fragment Shader program of a known width and height.
2. The image is broken up into blocks of a predetermined size (these blocks will form the image after the process is complete)
3. For each block, sample one of the pixels - this is now the color for the entire block. See (Figure 2 for an example of this process).
4. When complete, much of the image data will be lost because much of the image will be broken up into samples of its own color. There's no going back! A good example of this can be seen in Figure 3.

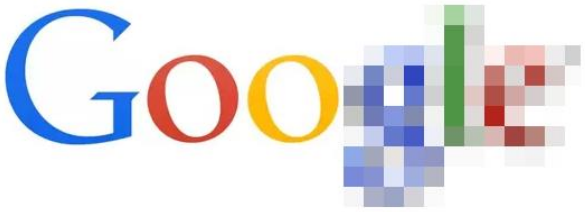

Figure 3. Half Pixelated Google Logo

2.2 Glow Effect - Cole

The glow effect on the eyes of the "alien head" in the scene is basically a blur effect, but only applied to certain portions of the image. The algorithm is as follows:

1. Render the scene normally, without the glow, to a texture and save it for later.
2. Render just the parts of the scene that glow to a buffer. The parts that don't glow should still occlude the parts

that do, so to accomplish this I temporarily changed all materials that weren't the glowing material in the scene to black.

3. Blur the buffer of just the glowing parts of the scene. For this, you can just use a standard box blur, but I opted for a gaussian blur to try and make it a little more natural looking. One important detail however is that if the blur's kernel sums to 1, it is likely only going to produce a very faint blur. I ended up using a kernel that summed to 5. Another variation on a traditional blur is to do it in two passes for performance. On the first pass, you simply blur along one axis and on the second you blur along the other. This produces the same result, but requires only 2N samples per pixel as opposed to N^2.
4. Finally, add the blurred image from step 3 on top of the normal render from step 1.

Figure 4. Phases of Glow Effect

2.3 Bezier Surface - Cole

A Bezier Surface was used to transform the render and make it look like it was swirling around. This was a fairly simple effect to accomplish; all I did was take the UV coordinates of the final render, pass them through a Bezier surface equation, and then used the resulting surface position as the new UV coordinates. A 2D Bezier surface can be modeled by the following equation:

$$\mathbf{p}(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} B_i^n(u)\, B_j^m(v)\mathbf{k}_{i,j}$$

Where *B* is a Bernstein polynomial:

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}$$

And **k** are the control points, and n and m are the number of control points along that UV axis. I chose 9 control points. 8 of them are the corners and centers of the edges of a square from (0,0) to (1,1). The final control point is animated along a circular path inscribed in that square. This essentially grabs the middle of the render and drags it around in a circle to create that swirl effect.

One issue I had with the implementation of the effect was the nCr requiring a factorial to compute. As far as I'm aware, theirs is no factorial function in glsl, and I can't directly compute it because it would require an unbounded loop. Fortunately we only need to compute a factorial as high as max(n, m) = 2 in this case. So I just used 3 if statements to compute the factorial of 0, 1, and 2.

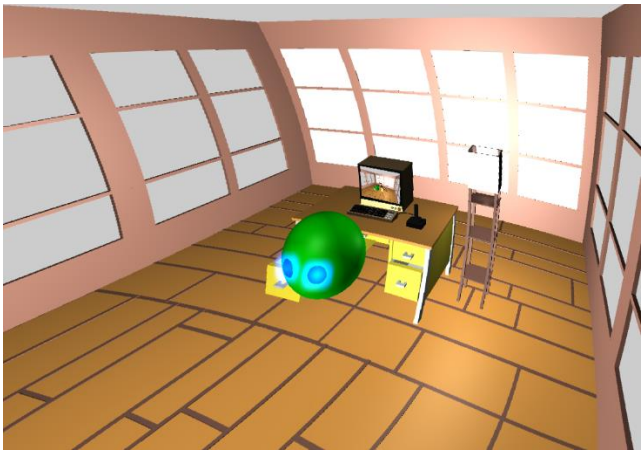

Figure 5. Demonstration of Bezier Effect

2.4 Chromatic Aberration Effect - Andy

The chromatic aberration effect kind of simulates a TV that had its RGB channels desynced. In the shader, the image will have its red, green, and blue (RGB) channels shifted away from one another. The process is quite simple:

1. A texture is passed into the shader.
2. An offset amount between each color channel is chosen.
3. The texture is broken into individual red, green, and blue channels.
4. The green channel gets no offset to give the viewer a frame of reference.
5. The red channel has the offset added to it and the blue channel has the offset subtracted from it so that both channels move in opposite directions.
6. Finally, add each color channel into the gl_FragColor.
7. Optionally, multiply the offset amount with a looping number set, like time in the scene, to have the red and blue channels jiggle around.
8. Take the looping number set and use it with cos() and sin() to cause the red and blue channels to spin around the green channel while jiggling.

This shader was created from a Shadertoy version. The Shadertoy used a texture() function to shift its RGB channels. For three.js that had to be changed to the texture2D() function which simplified the parameters passed in. The UV coordinates of the texture in three.js were generated from the vertex shader which simplified the fragment shader.

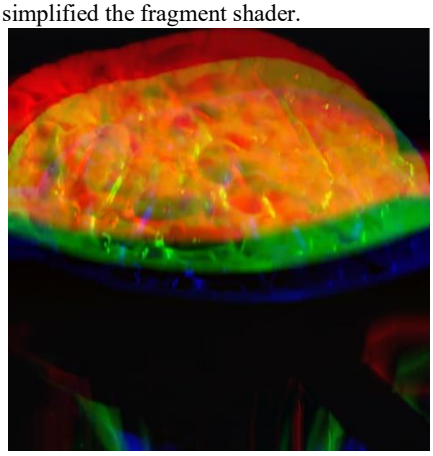

Figure 6. Demonstration of the chromatic aberration.

3 CONCLUSIONS

We ended up creating a scene with a variety of post-processing effects included. These effects were very easy to integrate together due to the independent nature of these effects. Many don't rely on the geometry at all, and only work on a single image that is the result of a normal render. Another result of that is that these effects are very computationally cheap as well. Adding these effects to your projects can easily and cheaply increase their visual appeal.

REFERENCES

[1] Computer Object https://poly.google.com/view/bS8OHMaA6Wt
[2] Standing Lamp
[3] Office Desk
[4] Three Quarter Room
[5] Wikipedia page on Bezier Surfaces https://en.wikipedia.org/wiki/B%C3%A9zier_surface
[6] Nvidia's *GPU Gems* https://developer.nvidia.com/gpugems/GPUGems/