Graphics Project

Landon Kump - jlk3735

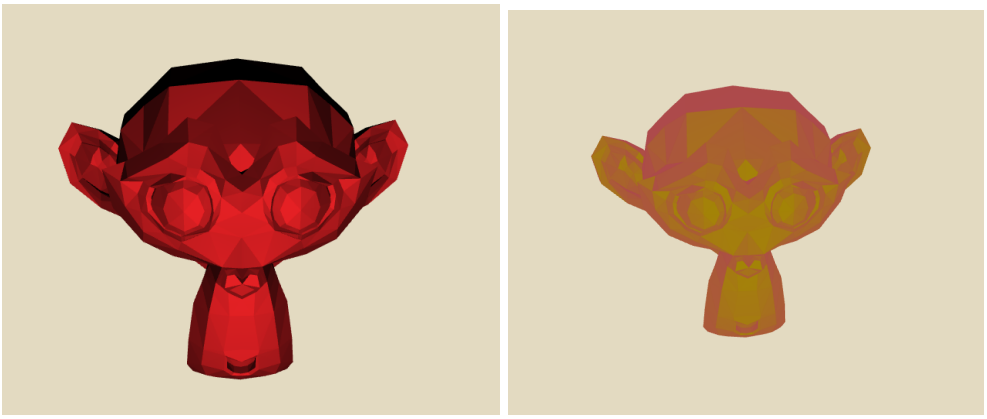
Project Goal

For this project, I have implemented a NPR painterly shader in Opengl using c++, GLFW, and GLEW using the Opengl tutorials project as a base to extend from. The goal of the project was to get the appearance of a painted scene in real-time, incorporating artistic techniques such as warm-cool shading and variance in color of brush strokes. I also had the goal of defining the form of the model using the brush strokes.

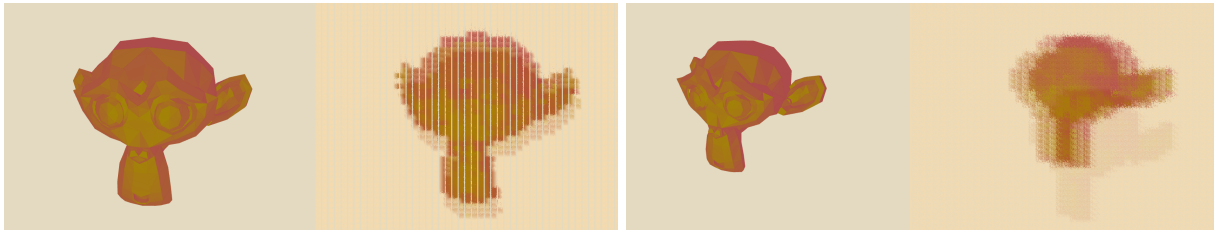
Process Summary

For references, I used Paint By Numbers by Paul Haeberli to understand what would be important for getting an artistic appearance. I used the Goochs' non-photorealistic lighting model to get the warm-cool shading I was going for, with some modifications to get a slight specular highlight.

Before and after Gooch Illumination Model



To get the brush strokes, I would render the result from Gooch Illumination to a texture, which would then be sampled at the vertices of a grid of quads covering the screen. This would be done in the post-processing vertex shader, which passes the color to the fragment shader. This way, a more uniform gradient of color is shown across the brush stroke.



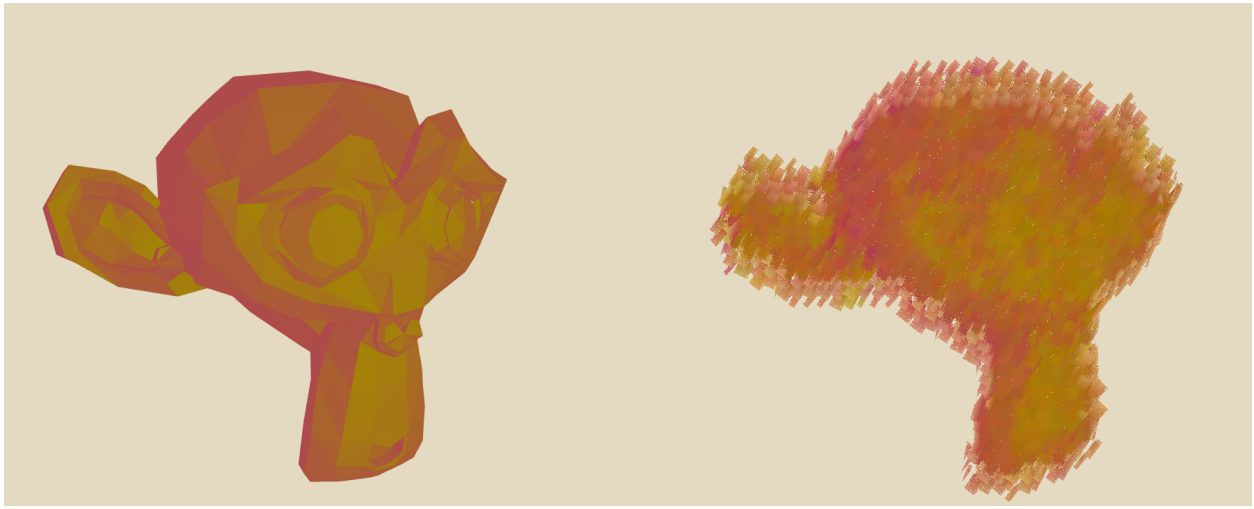
From the picture above, you can see that using this method created what appeared like a shadow underneath and around the painted object. This was particularly apparent with larger

brush sizes. This came from a majority of the vertices of the quad having the background color value, but one or two vertices would have the object color. To fix this, I discarded the vertices which had a color value close to the background color in the post-processing vertex shader. While not a perfect solution, since we must now know the background color for the shader, it does get rid of the shadow effect:

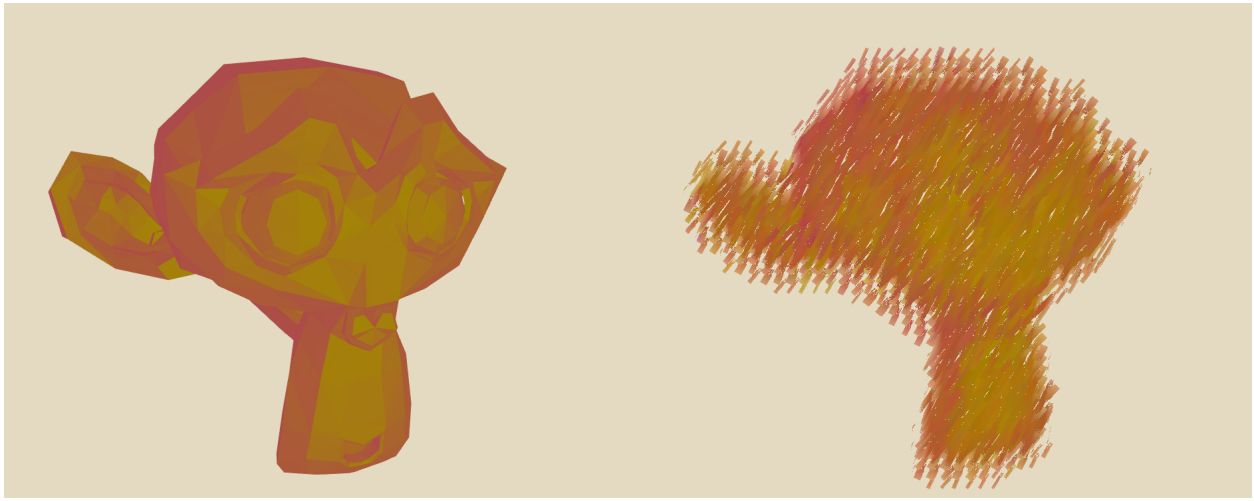
Lastly, I added some variation to the brush strokes by scaling and rotating the brush strokes randomly by some variation parameters. This randomization is done before we start rendering the screen, so it is not calculated per-frame.

```
void initializeScreenspaceQuadGrid(int grid_dimensions,
    std::vector<glm::vec3>& quad_grid_verts, std::vector<glm::vec2>& quad_uvs,
    std::vector<unsigned short>& quad_indices,
    std::vector<float>& brush_widths, std::vector<float>& brush_lengths,
    std::vector<float>& brush_rotations, std::vector<glm::vec3>& color_variance) {

    const int STRIDE = 4;
    const float delta = 2.0 / grid_dimensions;

    const float BRUSH_SIZE = delta;
    const float BRUSH_SIZE_VARIANCE = delta / 3.0; // Best with delta / 3.0
    const float ANGLE = -30.0; // Degrees // Best with -30
    const float ANGLE_VARIANCE = 15.0; // Degrees // Best with 10
    const float COLOR_VARIANCE = 0.05; // Best with 0.05
```

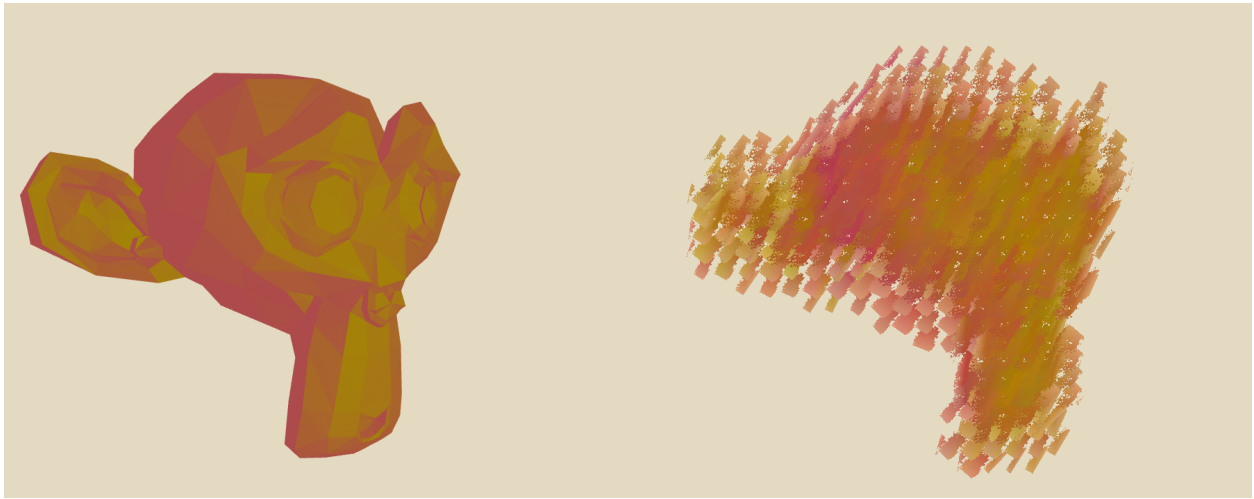
Render using the above parameters



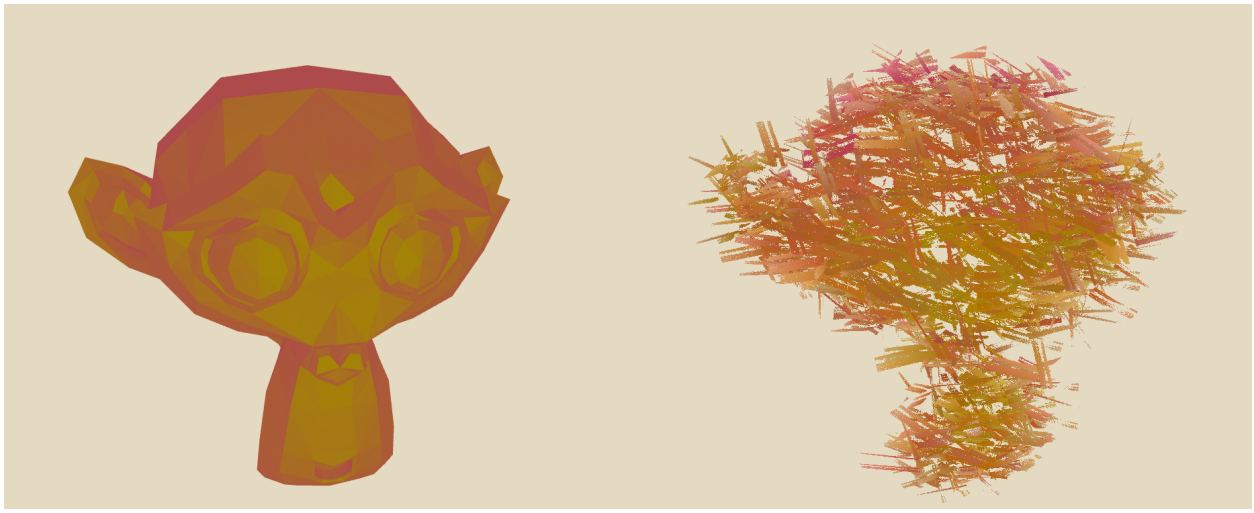
Using Smaller Brush Stroke Texture



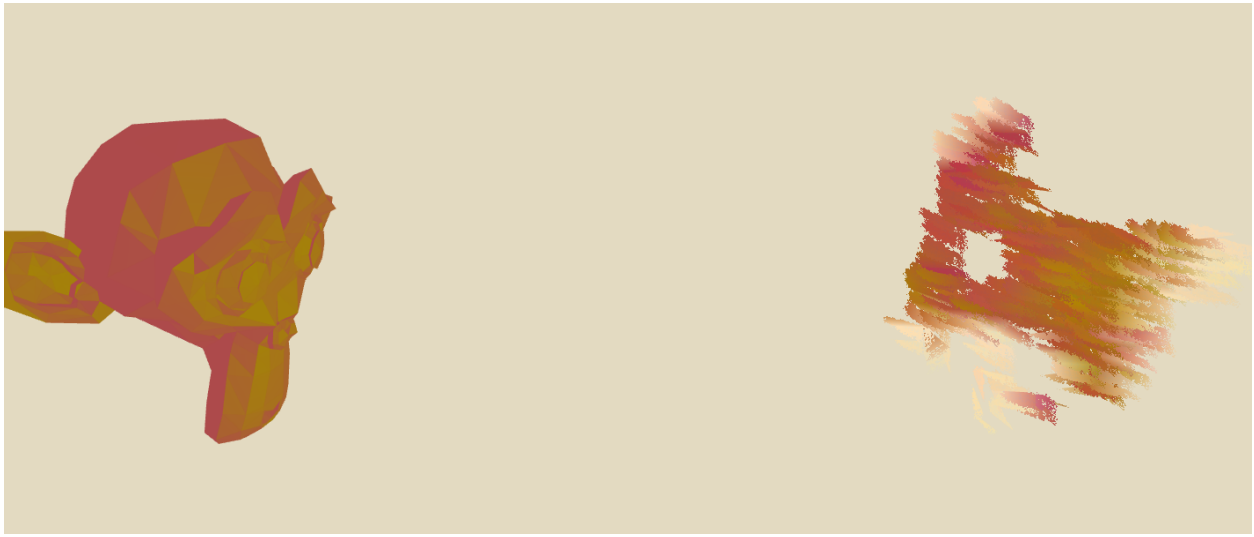
Using Splatter Brush Stroke Texture



Using Line Brush Texture With High Variation in Angle



Unfortunately I was not able to implement having brush direction and scale corresponding to the model normals and distance. I attempted to get the brush stroke to align with the tangent and bi-normals of each individual vertex, but had trouble getting things to render. The code is left in the shaders, but essentially, I would project the tangents and bi-normals of each vertex to screen space, then record the direction of the vectors in their own separate textures. Those textures get passed to the post-processing shaders where I attempt to align the brush strokes. However, the alignment didn't seem to work when I tried. This is the result:



The brush strokes did move with the model, but not in a way that outlined the form. It also appeared that many brush strokes had been translated by the rotation.

If I continue with this project, I would like to get that working first. Next, I would like to have the brush strokes correspond with the model in-world rather than have static brush strokes in screen space. I think this may be possible by translating, rotating, and perhaps shearing the brush stroke quads.

References:

Opengl-Tutorial Project. http://www.opengl-tutorial.org/

Brush Strokes Shader. https://gpuhacks.wordpress.com/2012/01/30/brush-strokes/

Paul Haeberli. 1990. Paint by numbers: abstract image representations. SIGGRAPH Comput. Graph. 24, 4 (Aug. 1990), 207–214. https://doi.org/10.1145/97880.97902

Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. 1998. A non-photorealistic lighting model for automatic technical illustration. In Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98). Association for Computing Machinery, New York, NY, USA, 447–452. https://doi.org/10.1145/280814.280950