1980s Retro Futurism

Emily Gagne
Emily.Gagne001@umb.edu
University of Massachusetts Boston

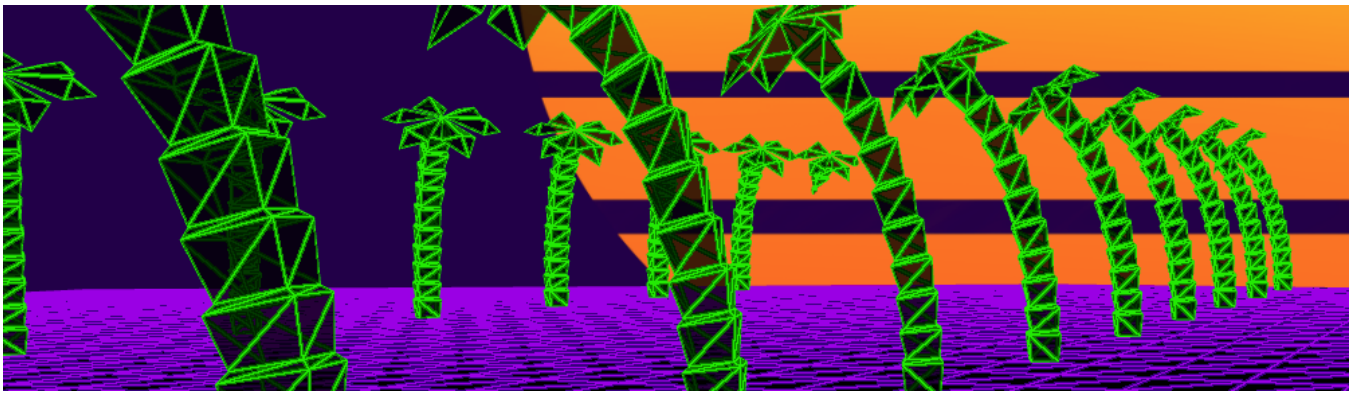

Figure 1: Sample output from the custom shader.

ABSTRACT

The goal of this project was to create a GLSL shader to mimic the neon and black grid common in 1980s-era futurism depictions.

KEYWORDS

WebGL, Visualization, GLSL, futurism, shader

ACM Reference Format:
Emily Gagne. 2022. 1980s Retro Futurism. In *CS460: Computer Graphics at UMass Boston, Fall 2022*. Boston, MA, USA, 2 pages. https://CS460.org

1 INTRODUCTION

This project is an example of how to display a wireframe with solid faces, preventing the model from being transparent. It allows a 3D model artist to easily apply this effect to their artwork. This could also be achieved by rendering each model twice, once as a solid and once as a wireframe, however, this provides no control over wire width and is expensive in rendering costs.

2 RELATED WORK

This project was based upon the Three.js Wireframe Material Example [1].

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
CS460, Fall 2022, Boston, MA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 1337.
https://CS460.org

3 METHOD

3.1 Implementation

The shader was implemented in much the same way as the Three.js Wireframe Material Example, however, the color assignment has changed. I used the edge calculation, which was previously used to determine the opacity, to instead determine which color is displayed. I use it as input into a smoothstep function, the output of which is stored in colorStep and is then used in the following code:

```
gl_FragColor.rgb =
(colorStep == black) ?
    face
:
    (alpha < 1.0) ?
        glow
    :
        inner;
```

This utilizes the result of the smoothstep function and the raw alpha value to determine if a fragment should be the supplied face, glow, or inner color. The alpha value for all fragments is determined by the color, as supplied by the user.

The implementation can be viewed at https://ceg9498.github.io/cs460student/final/, and the code is available at https://github.com/ceg9498/cs460student/tree/main/final.

3.2 Milestones

3.2.1 Milestone 1. The conditions necessary for satisfactory completion of the shader, along with some stretch goals, were determined.

3.2.2 Milestone 2. I researched methods of computing edge shading, including Sobel Edge Detection and the use of built-in Wireframe materials.

3.2.3 Milestone 3. Coding work on shader.

3.2.4 Milestone 4. Add a wrapper to the shader to allow for easy implementation.

3.2.5 Milestone 5. Assemble an interesting example of how the shader looks.

3.3 Challenges

- Challenge 1: Methods seen during research for implementing edge detection relied on GLSL Geometry Shader, which is not implemented for WebGL and would not work for this project.
- Challenge 2: Creating a wireframe with both a width larger than one pixel and the ability to have faces, which also places the inner, glow, and face colors appropriately.

4 RESULTS

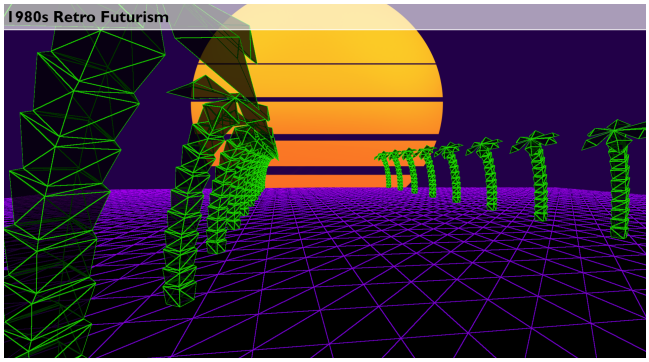

Figure 2: A sample of the final result.

5 CONCLUSIONS

The shader resulting from this project is an example of how to color edges and faces differently as a result of a single pass of rendering, while also giving the user full control over colors and opacity. This avoids doubling the render cost incurred by rendering each object as two different meshes - for a solid and a wireframe.

REFERENCES

[1] alteredq. 2012. WebGL - Materials - Wireframe. *URL: https://threejs.org/examples/#webgl_materials_wireframe* (2012).