Rocket in Space

CSCI 480 - Final Project

Liam Gleeson, Cole DeMaris, Daniel Shtunyuk

GitHub Repository: https://github.com/csci480-21w/fp-gleeson_shtunyuk_demaris

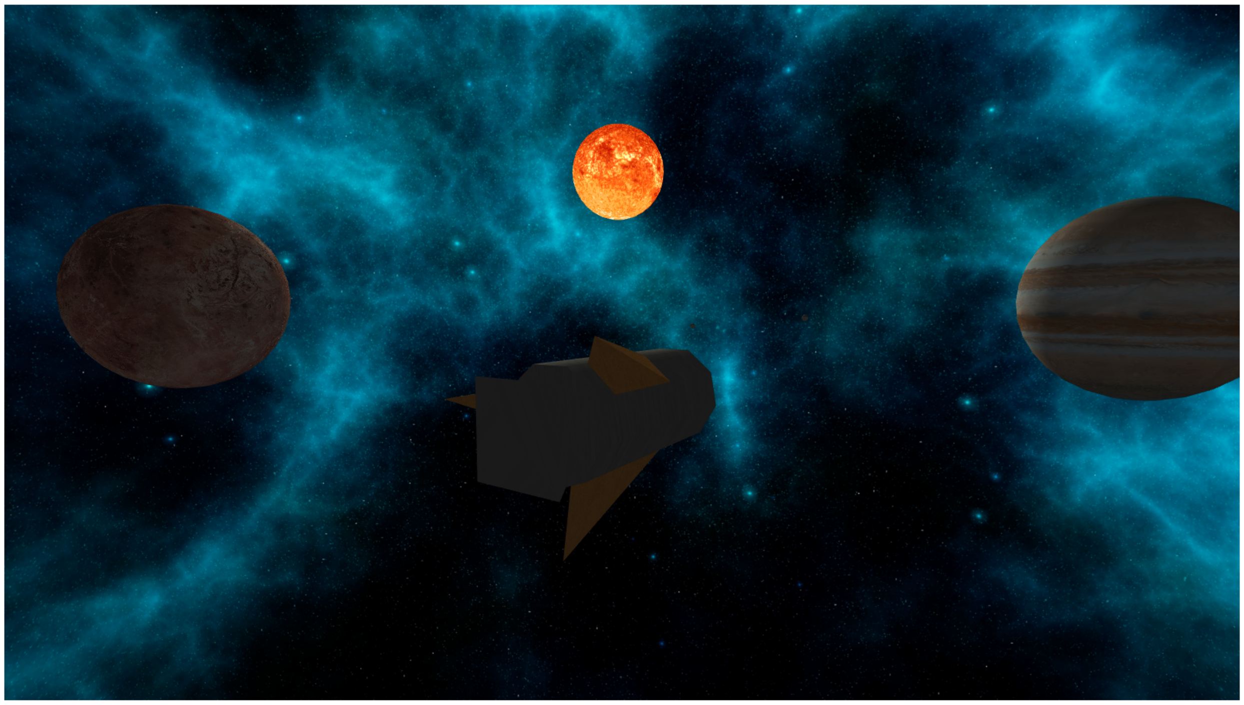
Summary:

For this project, we planned to create a scene that has an implementation of animations applied to a rocket as it travels through outer space, flying by planets and being lit up by a sun. We created a rocket ship object in an *.obj* format along with our planets (spheres) that were generated with a Julia program from Assignment 1 and we used a *.obj* parser to import the files. We spent some time trying to figure out the best way to import an *.obj* file in our JavaScript code, but after finding out we would have to use JavaScript *Promises* or setting a web server, we resorted to just copying the contents of the *.obj* files into their own dedicated *.js* files to import them as String-literal variables. Some time was set aside into modularizing the codebase from a single file into multiple files for different parts of the project.

We determined that we would prefer to create a fixed set of planets in our scene with a rocket ship that would follow a path around all the planets and have it loop around that path, instead of programmatically adding and removing planets as they move in and out of the viewport. As we set the layout of the scene, we also began creating the splines using control points that we plotted manually, then brought them into our code and used the coordinates generated from the splines to translate the camera and rocket. The camera view is moving along the spline path as well. A skybox was created to give the atmosphere of outer space with stars far away.

The shaders were quite a bit of work, but we managed to get textures mapped on the surfaces of objects that include diffuse color maps and normal maps. We didn't get to implementing the lighting shader correctly, so we don't have the shadows properly shading the rocket as it hides behind a planet from the sun. We also attempted the Particle System towards the end of the project and didn't manage to get it working properly in our time constraint, so our rocket is flameless.

Results:



Instructions:

Our project has been tested on both Chrome and Firefox and it seems to only run in Chrome at the moment, so we are dedicated to make sure it runs in Chrome.

Online:

1. Launch the Chrome browser and follow this link:

   https://csci480-21w.github.io/fp-gleeson_shtunyuk_demaris/

Locally:

1. Make sure you have Python 3 installed on your system.
2. Download or clone this repository.
3. In the root directory of the project, run a local HTTP server on port 9900:

```
> python3 -m http.server 9900
```

4. Visit the site with the link provided in the terminal window.

```
Serving HTTP on 0.0.0.0 port 9900 (http://0.0.0.0:9900/) ...
```

Code guide:

The project is divided into several files. The file initializeScene.js creates meshes and their respective textures and transformations for the objects in the scene. All of the WebGL initialization and the logic for 360 camera panning is done here also. The loading of meshes is done with objparser.js. Within initializeScene.js, we also create all the control points for the spline curve for our rocket. We also have a rotation value, rocketDirectionRot, that constantly calculates what the next angle of rotation would be for the rocket and applies it to our rocket model. This way our rocket is always facing the direction it is turning.

ShaderSource.js holds all the shaders used in the scene. Each shader is stored as a String Literal variable. This is where the implementation of normal maps is done.

shadedTriangleMesh.js holds the data unique to each object, obj file data as arrays, the shader program, and the textures used. The drawing of the objects to the canvas are done with this file's render method.

skybox.js mirrors the functionality of shadedTriangleMesh but is curated for implementing the skybox.

Matrix.js is the matrix math function file that helps us do all the necessary calculations with matrices.

Rocket.js and spher2.js are two files containing the obj data for a rocket and for a sphere. Using our objparser.js we are able to read this data and create the objects into our scene.

Splines.js is a function file that reads in an array of 4-array of Control points that are used to create the path with multiple splines around the scene. It has helper functions to keep track of which spline the rocket is moving along currently in the animation.