3D Portfolio

Zechao Lin
zechao.lin001@umb.edu
University of Massachusetts Boston

Figure 1: The landing page.

ABSTRACT

This project is to create a website as a personal portfolio. On top of that, there is a space scene created using Three.js to showcase how the web technologies can be combined to achieve some interesting effects, especially 3D rendering.

KEYWORDS

Three.js, HTML, CSS, JavaScript

ACM Reference Format:
Zechao Lin. 2022. 3D Portfolio. In *CS460: Computer Graphics at UMass Boston, Fall 2022*. Boston, MA, USA, 3 pages. https://CS460.org

1 INTRODUCTION

https://mrcode3.github.io/cs460student/finalProject/

The goal of this project is to learn Three.js, and it is a great learning journey along the way by building this website. The theme of this website is the cosmos. I try to render the most beautiful objects in the universe in my scene, and also with some spaceships. Apart from 3D scene, there is a layer of text layout which will be revealed when scrolling the page.

2 RELATED WORK

To get ideas of 3D portfolio in Three.js [1]. To better understand the CSS layout in grid [2]. To learn about post-processing in Three.js [3]. To learn about loading page when assets are being loading [4].

3 METHOD

There are three parts to this project. One is the 3D scene, with six major 3D objects (text/icons, 2 spaceships, sun, earth, black hole) and infinite background stars. All of these objects have animations excluding sun, with only glowing effect in place. And objects like spaceships and black hole are animated 3D models formatted in GLTF, these are free-to-use models from the internet. Second part is the text layout, it is a simple layout achieved by CSS grid and flex box. There are plenty of lorem ipsum worked as placeholders, and one can customize the content to make it a real web portfolio. Third is a loading page. While assets are not ready, it will display

CS460, Fall 2022, Boston, MA

ACM ISBN 1337.
https://CS460.org

a loading indicator to visitors, instead of showing an incomplete scene, this way it can be more user-friendly.

3.1 Implementation

Key implementations:

3.1.1 Loading Manager.

```
const loadingManager = new THREE.LoadingManager();
const progressContainer = document.querySelector
(".progress-container");
loadingManager.onLoad = function () {
  progressContainer.style.display = "none";
};
```

Loading manager makes sure the scene will be shown after assets have been loaded.

3.1.2 Loaders for Font/GLTF.

```
const fontLoader = new FontLoader(loadingManager);
const starshipLoader = new GLTFLoader(loadingManager);
const blackholeLoader = new GLTFLoader(loadingManager);

earthTexture = new THREE.TextureLoader(loadingManager).
load(
  "./public/earth.jpg"
);
normalTexture = new THREE.TextureLoader(loadingManager).
load(
  "./public/earth-normalmap.jpg"
);
```

Passing 'loadingManager' as the argument for it to know when assets are ready. The globe also has both textures for map and normal map in order for it to reflect lights. This is to make sure it does not look 'flat'.

3.1.3 Atmosphere for Globe.

```
atmosphereMaterial = new THREE.MeshStandardMaterial({
  emissive: 0x87ceeb, // glowing
  emissiveIntensity: 0.05,
  color: 0x87ceeb,
  transparent: true,
  opacity: 0.2,
});
atmosphere = new THREE.Mesh(
  new THREE.SphereGeometry(41, 32, 32),
  atmosphereMaterial
);
```

The easiest way I found on how to make an atmosphere for the globe is to use emissive property, and adjust the opacity of the material. Besides that, we also need the atmosphere slightly bigger than our globe, which is 41 (globe is 40). This way we create a sphere object at the same position as the globe, which looks like a globe with atmosphere.

3.1.4 Animations on 3D Models.

```
blackHoleMixer = new THREE.AnimationMixer(blackhole);
blackHoleAction = blackHoleMixer.
clipAction(loader.animations[0]);
blackHoleAction.timeScale = 0.001; // slow down
blackHoleAction.play();
```

Here is an example on how to trigger animations on an animated 3D models. As a side note, we can also use object's method like rotation or translation to achieve some basic animations.

3.1.5 Bloom Renderer.

```
const renderScene = new RenderPass(scene, camera);
const bloomPass = new UnrealBloomPass({parameters...});
bloomPass.threshold = 0;
bloomPass.strength = 1; //intensity of glow
bloomPass.radius = 0;
const bloomComposer = new EffectComposer(renderer);
```

The bloom renderer is often used to create a glowing or shimmering effect, especially around bright light sources or glowing objects. I used it to create a glowing sun.

3.1.6 Dynamic Scene while Scrolling.

```
factor = document.body.getBoundingClientRect().y;
starSpeed = factor + 4400; // offset
```

This is the key functionality of this website. If we know the position of the scroll bar, we will be able to move the camera accordingly.

3.2 Milestones

How did you structure the development?

3.2.1 Milestone 1. Looking at existing Three.js examples from their website, skimming through the interesting features for Three.js in the document.

3.2.2 Milestone 2. Start implementing features one by one, when a new idea comes up. I would just Google "how to...in Three.js". For example, how to make an object glow? That is when I found the post-processing feature in Three.js, I used it to implement the glowing sun in my scene.

3.2.3 Milestone 3. Following the same strategies, I was able to figure out how to load a GLTF model with animations.

3.2.4 Milestone 4. Implement dynamic scene when scrolling the page. Inspired by parallax scrolling effect.

3.2.5 Milestone 5. Make use of loading manager provided by Three.js such that loading indicator(HTML/CSS) will take over the screen when assets being loading.

3.3 Challenges

Describe the challenges you faced.

- Challenge 1: In order to make a globe more realistic, I had to add atmosphere on it. There are some ways to do it, but most of them are too complicated to understand (involving shading language).
- Challenge 2: To make the website responsive in terms of 3D scene. Now it is working when we resize the browser window, but not on the mobile phone.
- Challenge 3: It was hard to create a particle system for spaceships to emit gas. I may learn it in the feature, but the quickest way is to use animated models.

4 RESULTS

The best feature is that when scrolling the page, the camera will be moving accordingly. One is able to "travel" the space and enjoy the scenery, the sun, the earth and the beautiful black hole. In addition, there will be text content along the way, allowing a visitor to read just like an online portfolio.

5 CONCLUSIONS

Three.js is a really powerful JavaScript library for creating 3D graphics in the browser. There are so many interesting examples in their website. I have been enjoying reading the documentation and exploring examples to see what Three.js is capable of. I would say this course is an excellent starting point to dive into the 3D world in the web browser. I will keep polishing my work as I learn more in the feature. One thing that I noticed is that the possibility of making Three.js code dry and clean by using react-three-fiber. Basically, it is to write code in React(A JavaScript framework for front-end development) style, which means less code to achieve completed work because functions will be component-based and reusable. That is something I want to integrate into my project if I had more time. Below are some screenshots from this project.

Figure 2: Text content on top of the 3D scene.

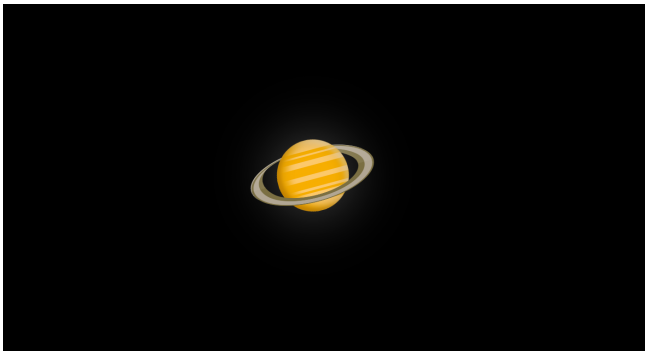

Figure 3: Loading assets.

REFERENCES

[1] Jeff Delaney. 2021. Build a Mind blowing 3D Portfolio Website. *URL:https://www.youtube.com/watch?v=Q7AOvWpIVHU* (2021).

[2] Chris House. 2021. A Complete Guide to CSS Grid. *URL: https://css-tricks.com/snippets/css/complete-guide-grid/* (2021).

[3] Three.js Manual. 2022. How to use post-processing. *URL: https://threejs.org/docs/manual/en/introduction/How-to-use-post-processing* (2022).

[4] Three.js Manual. 2022. LoadingManager. *URL: https://threejs.org/docs/api/en/loaders/managers/LoadingManager* (2022).