CS32310: Advanced Computer Graphics
Scene Modelling and Navigation – Report

Punit Shah

pus1@aber.ac.uk

Abstract.
This report describes how I modelled the rooms of a house using WebGL and *Three.js*. It details the contents of my scene and how I achieved them. It discusses the techniques used to build the scene, and any difficulties that arose from them.

A live demo is available at https://punit-shah.github.io/cs32310_assignment/.

1 Assignment structure

- **assignment/** – the root directory
  - **index.html** – the HTML document that loads the JavaScript and runs the application
  - **style.css** – the stylesheet with styles that enable the canvas to fill the page
  - **img/** - contains the texture image files for the application
  - ***js/*** – contains the JavaScript files for the application, including classes for each of the objects in the scene
    - ***lib/*** - contains third-party JavaScript libraries

2 Running the application

2.1 Serving the assignment directory.

To run this application, you must serve the assignment directory locally using a local development server. The content cannot be loaded by trying to open the file in the browser with the `file://` protocol, because the application loads textures from external files and the browser's security restrictions will prevent it from doing this. There are many ways to serve the code over localhost, but I used http-server[1], a simple command-line HTTP server that doesn't require any configuration. It requires node.js[2] to be installed on your computer before you can install it.

[1] https://www.npmjs.com/package/http-server

[2] https://nodejs.org/

To serve the assignment directory using http-server, navigate to it in a terminal, then run the following:

```
http-server .
```

By default, running this will host the code on port 8080. You can now run the application by visiting http://localhost:8080 in your browser.

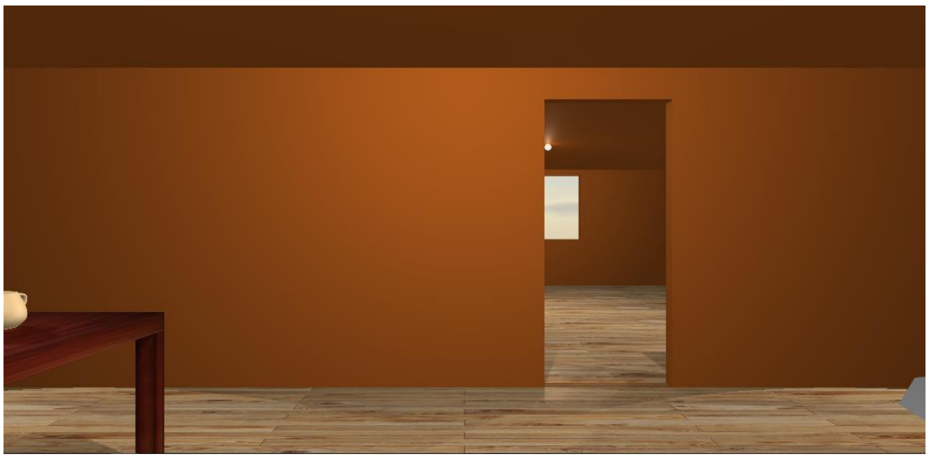
2.2 Controls

Once the application has loaded, the user will be able to navigate around the scene. I implemented a simple keyboard input control system that uses the arrow keys to move the camera.

Use the following keys to interact with the scene:

- The up arrow key will move the camera forward
- The down arrow key will move the camera backward
- The left and right arrow keys will rotate the camera in the corresponding direction

3 The scene

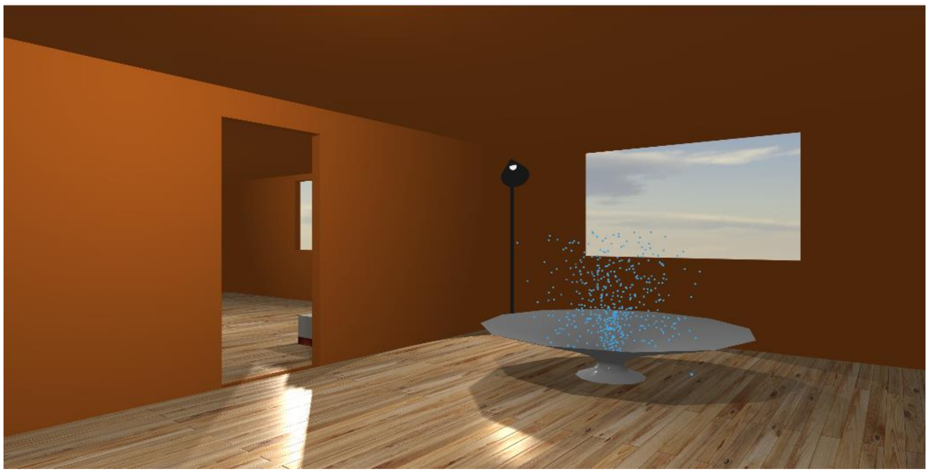


When the application has loaded, you should see the page filled with the above scene. Use the arrow keys to navigate the camera around it.

3.1 Rooms

The scene consists of two rooms. They are modelled with a realistic scale, with each unit representing one centimeter. Both rooms are 2 meters high. The first room, the room that the camera starts in, is rectangular and has a width of 8m and a depth of 4m. The second room is square, and has a width and height of 6m. The rooms are connected with a single doorway. Both rooms have rectangular windows that let the sunlight in, wooden floors, and smooth orange-brown walls.

To create the floor and the ceiling, I used Three.js's PlaneGeometry class. The floor uses Three.js's MeshStandardMaterial, with a reasonably high roughness value and low metalness value. I found that the inverse made the floor too bright and shiny. The floor has a diffuse map for the wood texture, and makes use of a bump map to simulate raised wood planks. It also uses a roughness map to make the wood planks less rough and shinier, and the gaps between them more rough and less shiny. I am quite pleased with the result, as it reacts to light in a very realistic manner.


Realistic specular highlights on the floor from the sunlight coming through the window, and the lamp in the corner

Initially, I found that the floor texture furthest from the camera looked very blurry if you looked through the door from one room to the other. I fixed this by increasing the anisotropy value on the texture to make use of anisotropic filtering. This increases the number of texture samples taken[3]. Doing this resulted in a less blurry result, though it is quite intensive on memory.

The walls are created with Three.js's ExtrudeGeometry class. This geometry class takes a Shape instance and creates an extrude from it. A particularly useful feature of being able to use Shapes is being able to cut holes from them, which I used to create the windows and doorway from. One downside to this was that regular texture maps don't work well with the geometries created by this class, and creating UV maps for them are quite difficult. I settled for smooth walls, as the result is still pleasing.

[3] Three.js documentation on the Texture class - https://threejs.org/docs/api/textures/Texture.html

3.2 Lighting

I used three different types of Lights provided by Three.js.

. DirectionalLight

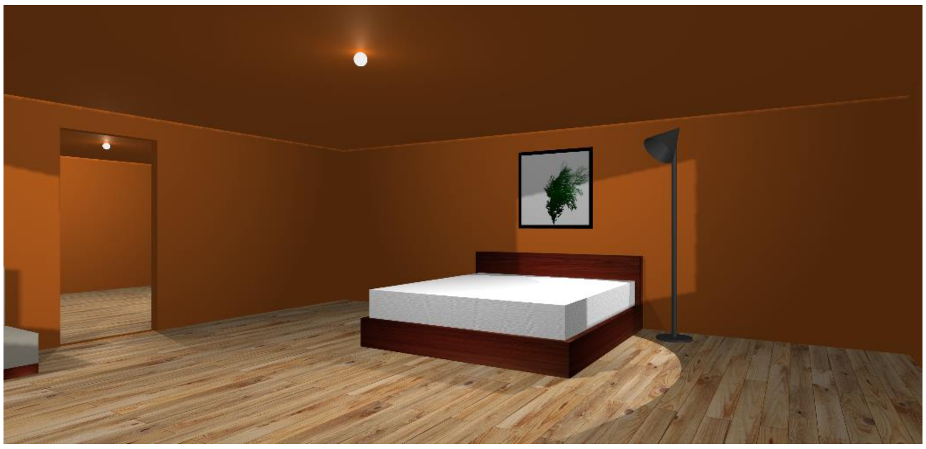
This class creates a light that behaves as though it is infinitely far away, and shines from a specific direction rather than a specific position[4]. I used this to model the Sun. Its light shines through the windows from a north-easterly direction outside.

. PointLight

This class creates a light at a specific position and shines light in all directions[5]. I used this to create the ceiling light in the middle of each room. The light rays decay realistically, and have a distance of 4m.

. SpotLight

This class creates lights similar to PointLight, but creates a falloff cone in one direction. I used these to create the light from my lamp objects, with the same decay and distance properties as the ceiling lights. The spotlight cone always faces the same direction as the lamp's shade to create a realistic lamp.



PointLight and SpotLight being used as a ceiling light and a directional lamp

. Shadows

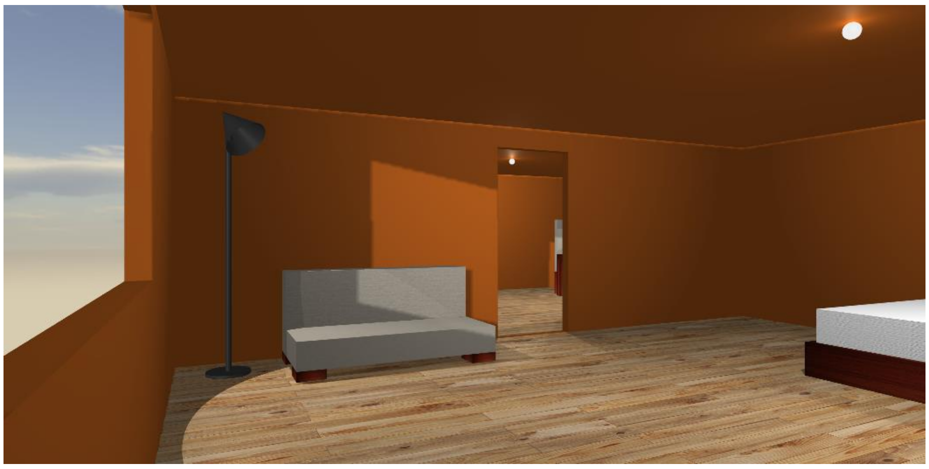
I enabled shadow map rendering to allow my lights to cast shadows, using the PCFSoftShadowMap shadow map type, which uses Percentage Closer Filtering to

---

[4] Three.js documentation on the DirectionalLight class
https://threejs.org/docs/#Reference/Lights/DirectionalLight

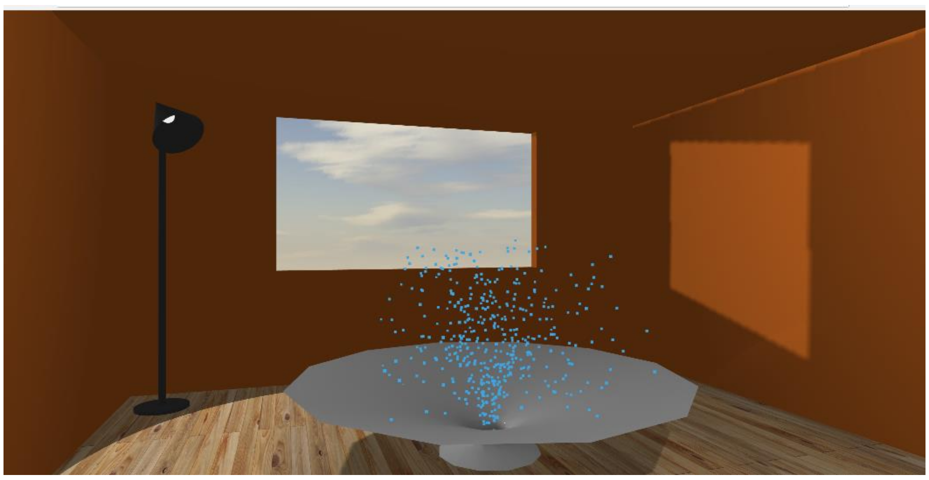
[5] Three.js documentation on the PointLight class
https://threejs.org/docs/#Reference/Lights/PointLight

create smoother shadows, and bilinear filtering to create softer shadows[6]. This resulted in fairly realistic shadows, and a nice effect from the wall shadows that looked like sunlight coming through the windows and falling on the walls inside.



3.3 Objects

. Water Fountain



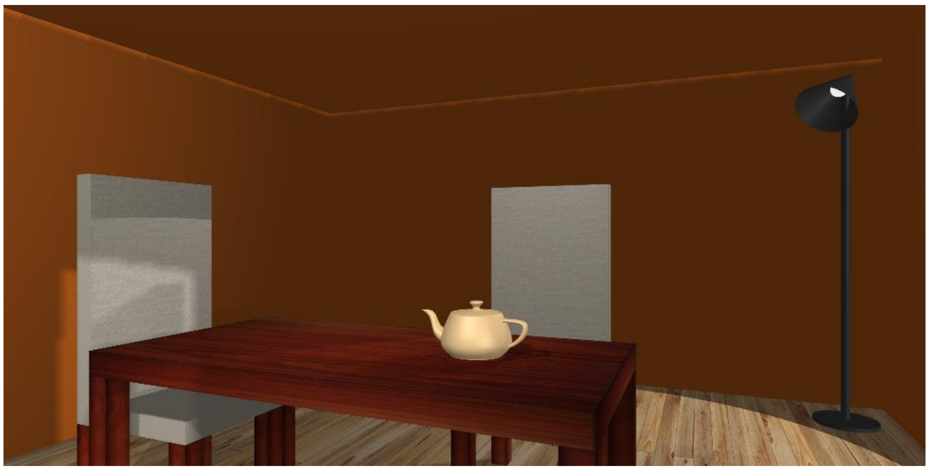
The water fountain in the first room makes use of Three.js's Points class to create particles with randomly generated velocities moving upwards and outwards before falling down to simulate a water fountain.

[6] Three.js documentation on its WebGL renderer
https://threejs.org/docs/api/renderers/WebGLRenderer.html

The code for the water fountain was inspired by Matthew Chase's online tutorial on Three.js Particles[7].

For the fountain base, I generated a set of vertices and used LatheGeometry to create an axial symmetry with them. The result is a wide dish shape with a small stand.

. Table, Chairs, Teapot



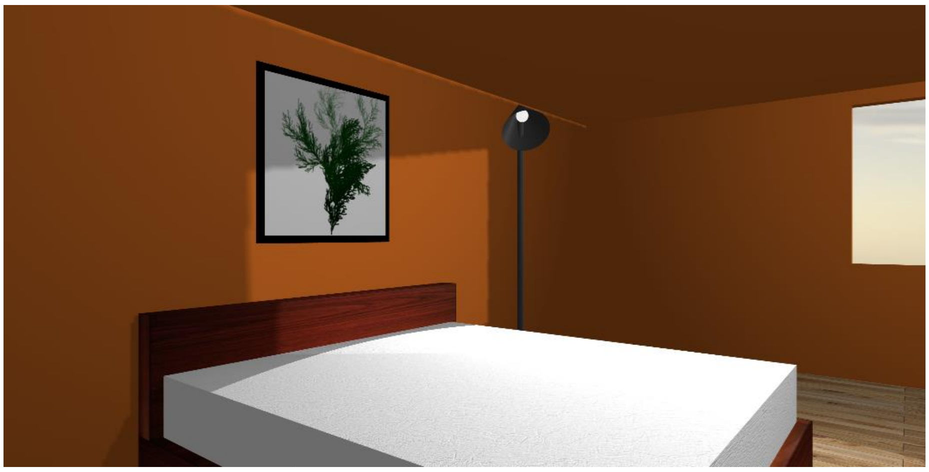
The wrote classes to generate table and chair objects that take parameters for their height, width, and depth. They are generated using BoxGeometry, and textured to look like a wooden table or a fabric chair with wooden legs.

The famous Utah Teapot on the table was created with an external Geometry class by Eric Haines[8].

[7] THREE.js Particles by Matthew Chase
https://codepen.io/antishow/post/three-js-particles

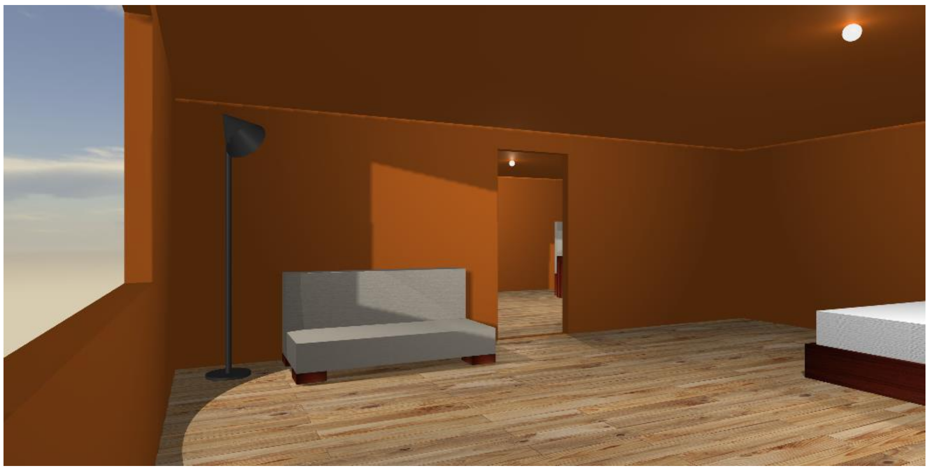
[8] Interactive 3D Graphics course on Udacity
https://www.udacity.com/course/interactive-3d-graphics--cs291

. Bed, Sofa, Tree painting



The bed was created in a similar way to the chairs and table, using the BoxGeometry class to create its parts. There is a bump map on the mattress to simulate creases in the fabric.

The tree painting is randomly generated with each time the application is loaded, using Lindenmeyer systems to generate the branches. The thickness decreases and the color brightness increases the further up the tree it goes.



The sofa is created with my Chair class, by giving it a wider width and a shorter height, to demonstrate how it can be used differently.