VR Animation and Textures

Ganesh Barma
IMT2018024
IIITB
ganesh.barma@iiitb.org

Amith Konda
IMT2018037
IIITB
amith.sai@iiitb.org

Sai Snehith
IMT2018062
IIITB
sai.snehith@iiitb.org

***Abstract*—In this assignment, we are doing a 3D rendering with animation, camera setup, and textures. Learning objectives of this assignment are: Procedural animation of a scene, Modeling a dynamic scene with a Scene Graph and relative transforms, and enabling animation through the scene graph, Generating texture mappings for objects of different shapes, Managing lights and camera, Handling collisions and Basics of a VR application.**

***Index Terms*—**

I. Problem Statement

Build a VR application with a animation storyboard. Design a scene, objects and implement animation using a hierarchical scene-graph. Design the animation in a controlled manner such that a group of objects move in a predefined path with leader deciding the path and other members following it. The path should have both static dynamic obstacles, and when the leader comes close to an obstacle, it changes its path to avoid and move past without colliding. An avatar object should be added, which gives the application VR like feature. The user controls the movements of the avatar to simulate a VR setting. The avatar can attach itself to a moving object and can perform jump, turning operations while moving along the object. All the objects in scene should be rendered using textures. Based on user control, the textures of objects can be changed. User controls should also enable switching to cylindrical and spherical maps for some of the objects. The scene should be illuminated by multiple lights such as fixed light, fixed but tracking the moving object and a light source attached to moving objects. The scene should be observed by 3 cameras, where one of them is at fixed position observing the whole scene (default), overhead drone camera (movement controlled by user input), and another camera attached to the avatar to simulate it's view of the scene

II. Introduction

In this assignment we have built an VR application with a animation story board. We generate some scenes, objects and an avatar, in which the avatar explores the place. The place consists of things like lights,a crossroad intersection, railway track and a train. Some of these objects can be controlled by the user like switching on/off the lights. The user can control the avatar such that, he can move around the place and he can also jump on the train. So based on user control, the textures of objects can be changed.

The scene's rendering can be divided into four main modules. Scene and Animation, Appearance and Textures, Lighting and Camera are the four categories. In the following section, we'll go through the features and core implementation of each of these modules.

III. Functionalities

The application is programmed using THREEJS graphics library, with output scene rendered on a browser. Imported threejs, ObjLoader, MTL Loader from on three js fundamentals website.

A. Scene Graph

We designed a animation scene graph with a road, grass, train track, train, avatar. The avatar is a 3d model which is loaded using OBJ loader and MTL loader. We can translate the avatar using the keyboard,

1) Arrow Up to move avatar in +y direction.
2) Arrow Down to move avatar in -y direction.
3) Arrow Right to move avatar in +x direction.
4) Arrow Left to move avatar in -x direction.
5) Key x to move avatar in +z direction.
6) Key z to move avatar in -z direction.

We have written an automated code for train to translate by its own, in animate function, we are changing the x and y positions of train by incrementing a variable angle and changing x by the track radius multiplied by cos(angle) and similarly y by sin(angle). And added two additional meshes

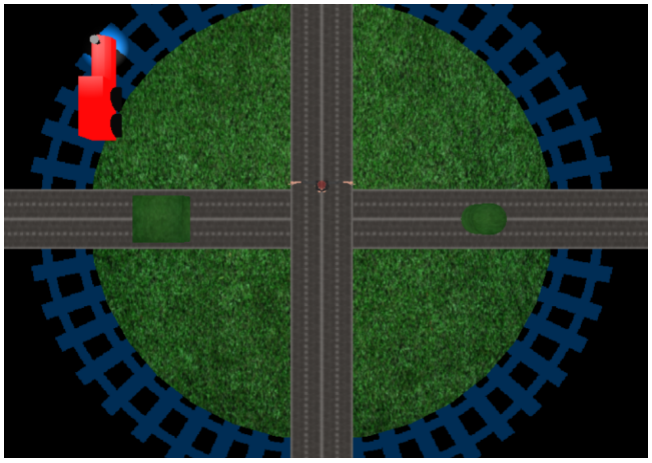

Fig. 1.

cube and cylinder, to show cylindrical mapping and spherical mapping

B. Avatar attaching itself to moving objects

In this functionality, our avatar can be attached to one of the moving objects. We have added a GUI control to toggle one\off to attach or not attach to the objects. The avatar only sticks to object when the bounding box of avatar is intersecting with bounding box of train. We have used three.js functions to get the bounding box. If they are intersecting then we are adding the avatar to the train group. The avatar can jump while it is attached to object or can rotate around.

1) Key "K" is used to jump while attached to train.

C. Adding Textures

To make the scene more realistic, we have used textures, the textures we used are train track, road, grass and created two objects to show Cylindrical mapping and Spherical mapping.For roads and tracks, We used texture loader from three.js to load the texture and used plane geometry and material used is Phong material with map as texture and on double side. Then we can create a mesh with above geometry and material. For grass, we have done the same thing as above but used circular geometry.

D. Spherical Mapping

To compute a custom mapping for any geometry we should map the geometry coordinates (x, y, z) to a texture space (u,v) programatically.

So for every geometry we can extract the u,v values and to show spherical mapping we need to change those, the formula to changes the u,v values are

$$u = (Math.atan2(z, x))/(Math.PI * 0.5) - 0.5$$

$$v = 0.5 - (Math.asin(y))/Math.PI$$

and then we need to make geometry.UVneedsUpdate true to change the values. Then we can create a mesh with above geometry with a texture material to show spherical mapping.

E. Cylindrical Mapping

This is like spherical mapping except assume there is tiny cylinder projecting on to your model.

So for every geometry we can extract the u,v values and to show cylindrical mapping we need to change those, the formula to changes the u,v values are

$$u = (Math.atan2(x, z))/(Math.PI * 0.5) + 0.5$$

$$v = y$$

and then we need to make geometry.UVneedsUpdate true to change the values. Then we can create a mesh with above geometry with a texture material to show cylindrical mapping.

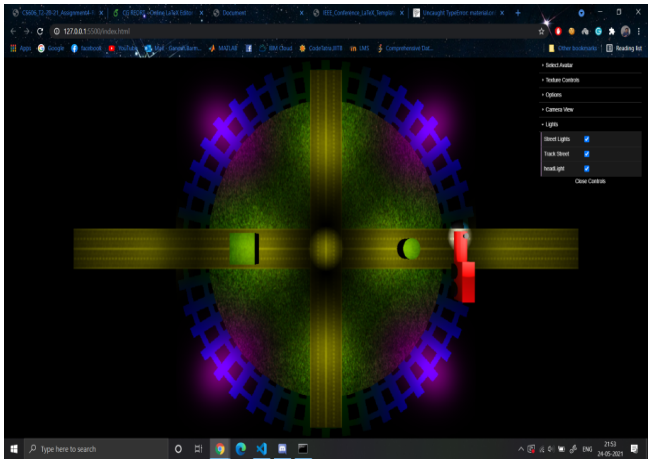

Fig. 2.

F. Lighting

The scene is illuminated by multiple lights. We are be able to switch any of these sets
of lights off or on. The light sets we used are street lights, street lights for tracks, and headlight for train. For street lights, we used point light from three.js, I used 9 of these lights and included positions, color, intensity, radius of light to appear and added to the scene. We have added GUI control to toggle one\off to switch on and off the lights.

For Track street lights, I used same as above but used different

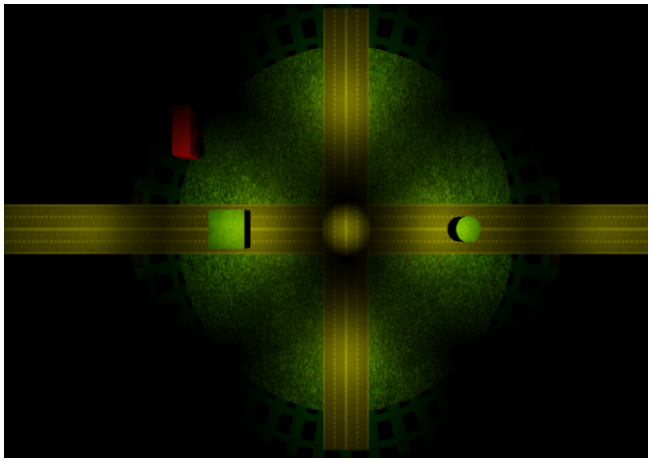

Fig. 3.

color, intensity and created four of them and added them to the scene. We can on\off the lights.

For train head light, we used point light from three.js and set it to the initial position of train. In animate function, we are updating everytime the position of light so that it moves with the train. We can on\off the lights.

G. Camera

The scene is observed by 2 cameras: default camera and drone camera.

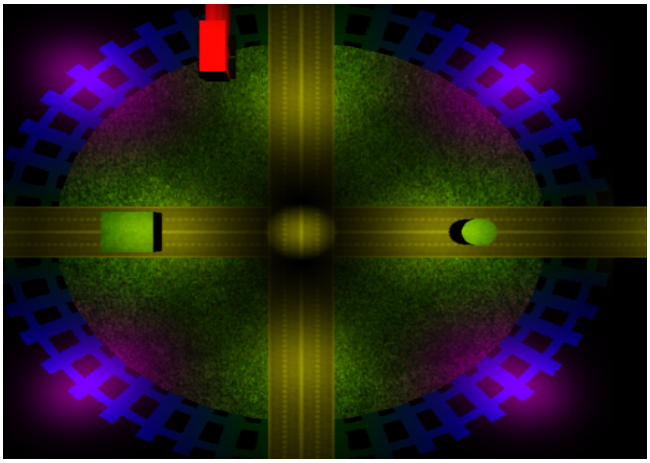

Fig. 4.

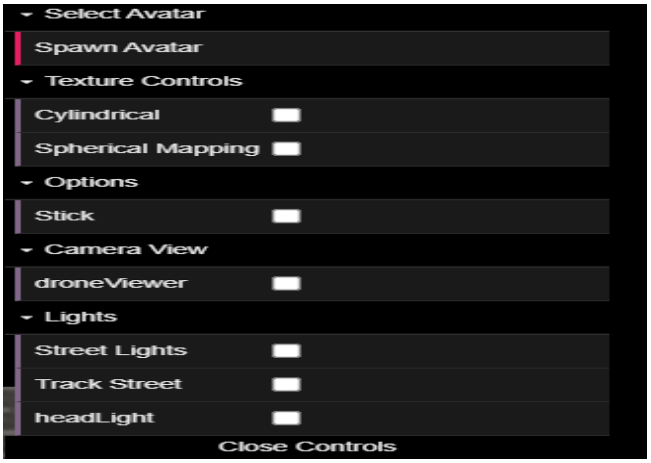

Fig. 5.

1) Default Camera: This is a fixed and default camera which is used to see the whole scene. I have also added Orbit contols to the camera so we can rotate or move the camera if we want.

2) Drone Viewer: This is also a default camera but with controls. We can translate the drone camera using keyboard keys,

1) Arrow Up to move camera in +y direction.
2) Arrow Down to move camera in -y direction.
3) Arrow Right to move camera in +x direction.
4) Arrow Left to move camera in -x direction.
5) Key x to move camera in +z direction.
6) Key z to move camera in -z direction.

We have used gui control to go into drone view or be in default view.

IV. Gui Controls

We have used dat-gui javascript libray to add gui controls.

V. Acknowledgements

We would like to thank Prof Jaya and Prof Srikanth for coming up with such an interesting project and the TAs for their tutorials and quick responses to doubts we had when attempting this assignment.

VI. Conclusion

We were able to make a animation scene using scene-graph and apply motion to the objects, following a pre-defined path.We rendered the scene with different cameras, specifically included a VR character as an object to the scene. A camera was attached to this VR object to give real-time experience. We experimented with different lights such as static, tracking and moving lights attached to objects. We learnt how to apply different textures to objects and make it appear visually aesthetic. To perform user operations, we added GUI components to control various aspects of the scene.

VII. References

1) https://threejs.org/docs/index.htmlmanual/en/introduction/Creating-a-scene
2) https://threejsfundamentals.org/threejs/lessons/threejs-scenegraph.html
3) https://stackoverflow.com/questions/20774648/three-js-generate-uv-coordinate
4) https://www.cgtrader.com/free-3d-models/avatar - Used for avatar 3d model