Assignment - 4

[1]Arjun Verma : IMT2017008
[1]Naveen Kumar : IMT2017029
[1]Sushranth Hebbar : IMT2017042
[1]Swasti Shreya Mishra : IMT2017043
T2-20-CS 606: Computer Graphics
Course Instructors : Prof. Jaya Sreevalsan Nair
Prof. T.K Srikanth

3D Rendering with Animation, Camera setup, and Textures

Abstract. *This technical report contains a brief overview of the methodology involved in solving the given problem statement. The github repository for the submission can be found here.*

1. Introduction

This mini-project intended us to develop a simplified VR application to understand it from the perspective of Computer Graphics. Concepts such as animation, camera setup, lighting and textures were implemented in the project using the popular Three.js library. The scene in the application comprises of a track amidst a lawn filled with trees. There are multiple cars circulating on the track. Only one particular car among these can be controlled by the user which also serves as a leader car for the rest of the cars on the track. Features such as collision avoidance and leader-following have been implemented with respect to the cars. The scene also consists of a virtual avatar of the user. The user can move the avatar anywhere on the scene and can also opt for viewing the scene from the avatar's perspective just like a typical VR application.

2. Methodology

In this section, we discuss the distinct characteristics of the approaches involved in designing the application.

2.1. Scene and Animation

Feature A : We have implemented a simple leader-follower model where once a vehicle starts following its leader, it accelerates when the leader accelerates and decelerates when the leader decelerates by maintaining a safe distance ($delta$) between them. In our application, there exists only one leader which goes on its own path and other vehicles travel on the path the leader traces along with maintaining the safe distance. If there are multiple vehicles between the follower and the leader, the vehicle immediately in front of the follower acts like a leader to it, with which it should maintain a distance of $delta$.

Feature B : Our application does allow for the detection and avoidance of static and dynamic objects. The avatar in the scene, when on track serves as a static obstacle for the leader car. The rest of the moving vehicles serve as dynamic obstacles for the leader car as they may suddenly appear on its path of motion. Collision detection has been implemented by encapsulating the object within multiple cylindrical objects which serve as the bounding box or “hit zones” for the objects. Collision avoidance has been implemented by computing the distance between the hit zones of objects and limiting the speed of the controlled car based on whether this distance crosses a defined threshold.

Feature C : When the leader car detects a collision possibility with the avatar on the track and stops just before it, the avatar has an option to attach itself to the leader car. If the avatar chooses to mount the car, it sits atop the vehicle and now moves along with the car while still retaining the ability to turn around and jump.

2.2. Appearance and Textures

We create our own 2D elements using html canvas elements to serve as textures in various places. The two prominent places where these have been applied are on the cars’ side and front panels, where these textures serve the purpose of representing windows. Similarly, for the curbs on the scene, a line marking texture was created and applied. A plain green texture was applied to the lawn and tree crowns in the scene.

2.3. Lighting

Feature A : We have implemented the street lights in two ways:

- We create a lamp post object and place a point light at the center of the lamp.
- We create another type of lamp post object where we place 4 spotlights at the center of the lamp facing in 4 different (symmetrical) directions. The reason 4 spotlights have been used is because we use a cuboidal object for the lamp of the lamp post.

We also experimented with RectAreaLight, where we make the surfaces of the cuboidal lamp of the lamp post as RectAreaLight objects.

Feature B : A search light has been implemented by fixing a spotlight at a fixed location. The target of this spotlight has been set to a moving car. A spotlight target is created and it’s position is set to that of the target car. This position is updated along with the car’s position in each frame. Finally, this spotlight target is added to the scene.

Feature C : The headlights have been implemented by attaching two spotlights to the front of the cars. Two points just in front of these light sources serve as respective targets for these lights at all times during the entirety of the cars’ motion. This has been achieved by adding the ‘headlight objects’ to the car group.

2.4. Camera

Feature A : A perspective camera object is created. It is then fixed at a certain coordinate. It is made to look at the origin. This is the default view and it is invoke by pressing the key R. It is updated with the fixed coordinates whenever the key is pressed.

Feature B : The drone view is invoked by pressing the key C. The arrow keys are used to move the drone. The translation of the drone is stored in a vector. The same camera object is reused for this mode. The position of the camera is reset to provide a top down view of the scene. Then it is updated again with the translation vector. And then the new frame is rendered with this updated camera object.

Feature C : A new perspective camera is created for this mode. It is invoked by pressing the Q key. Then the position of this camera is set to that of the avatar. So, for each frame, the camera position is updated and set equal to that of the avatar. This way, the camera provides a first person view of the avatar when it is mounted on a vehicle and when it is unmounted and free to move.

3. Conclusion

In conclusion, the report has successfully elaborated on all the required deliverables. The methodology section briefly discusses the approaches behind the implementation of the application. Sample images of the scene from multiple views has also been added to the repository. The instruction set to run the different components of the code have been included in the ReadMe of the github link mentioned in abstract.