Animation & Textures*

*CS606: Computer Graphics / Term 2 (2020-21) / Programming Assignment 4

Manan Bansal
IMT2018039
IIIT Bangalore
Manan.Bansal@iiitb.ac.in

Soham Kolhe
IMT2018073
IIIT Bangalore
Soham.Kolhe@iiitb.ac.in

Tanmay Arora
IMT2018078
IIIT Bangalore
Tanmay.Arora@iiitb.ac.in

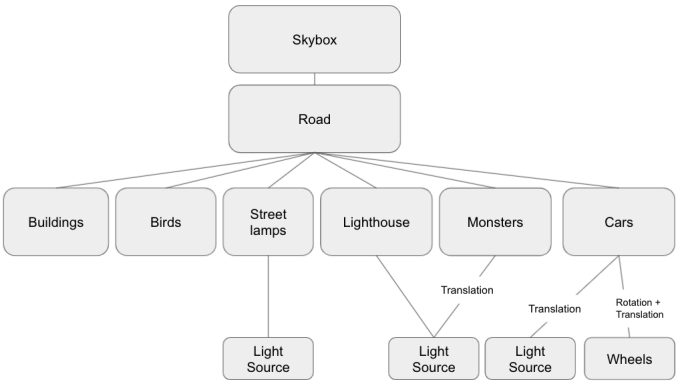
I. Introduction

This assignment aims to create an interactive system using WebGL and Three.js for rendering 3D animations which are capable of basic 3D translation and rotation as done in the previous assignment, with the ability to detect collisions. The scene deals with multiple light sources. It also deals with using blender to create the 3D Models.

II. Background

Earth is being attacked by monsters, and we are looking at the city of Bangalore where even after a monster attack, people are still late for their jobs so they want to avoid the monsters and move on with their lives. The Government has also set up a lighthouse in the middle of the road to keep an eye on the monsters while the army arrives to fight them.

III. Scene Graph Organisation



Our objects follow the above relationships which are further explained in our **Approach** section.

IV. Approach

There are 3 camera modes, Default, Drone and Avatar (attached to monsters) which can be toggled using “C”.

1) **Skybox:** Everything is contained in our **Skybox**, which inturn contains our road (Plane) which contains all our little objects like buildings, birds, street lamps, lighthouse, monsters, cars etc.

   Within our **road**, we have:

2) **Buildings** on either side of the road, which have 2 different texture types, which can be toggled via “T”.

3) **Street-lamps** outside every house on the edge of the road, all street lamps have 1 texture type and have 1 light source each attached to them which illuminates the area around them. You can control the light of the street-lamps on/off using “O”.

   The light object has a value called .visible which when set to false switches off the light source and when set to true switches it on. We have used an inbuilt function in THREE.js called .Spotlight() which associates the lamp object with the light source.

   On the road, we can also see:

4) **Cars** from the busy Bangalore streets minding their own business. All 4 cars have similar textures all around with 4 different colours.

   All cars have 2 headlights which helps them to see the road ahead of them. The wheels of the cars are also dependent on the car’s translation and rotate a certain degree with every unit of moment of their respective cars.

   But, the cars have some obstacles to overcome and to get to their offices, i.e.

5) **The Monsters** have taken over the streets in a group and they follow their leader everywhere they go. The monsters move in all four directions (Forward, Backward, Left, Right) with the help of the "Arrow keys".

   The monsters want to attack the cars and disturb the flow of the city, but the drivers are hell bent on reaching their offices to make big money, so whenever a monster collides with a car, the car changes its lane and keeps going forward, when it sees that the monsters have left their lane, they go back to their original lane.

   The monsters have 1 texture common for all three of them

   To keep the monsters in check, we have:

6) **A lighthouse**, which focuses a light on the monsters so that Bengaluru police can keep a check on the monsters at all times while they prepare to attack them.

   The lighthouse has 2 texture types and a light at the top of it which moves in accordance to the translation of the monsters. The building of the lighthouse is created using the CylindricalGeometery and ConeGeometery Functions.

V. Discussions

How are the position and orientation of lights and cameras calculated?

The lights are created using the Spotlight() function in THREE.js. It consists of two parts, first the light source position and second the target position. The light source position is the position from where the light will be emitted and the target is the position where the light will fall.

For the camera, we have orbital control to move the camera around and perspective camera to set it up.

How is the computation of animation including collision detection/avoidance done?

Every object has its own bounding box. if an object collides with another object (detected using intersection of bounding boxes), the cars will change its path of translation.

We tried ray tracing as well, but since it is an O(n) algorithm, and we had 75 objects, it took a lot of time for the same hence it reduced the number of frames our animation could output in a second. Another reason why didn't go ahead with ray tracing for the same is because we only had to detect collisions with 4 objects (cars), the rest of the calculations were of no use to us.

References

1) https://threejs.org/examples/webgl$_{loader_md2_control}$
2) https://threejs.org/examples/webgl$_{materials_car}$
3) https://threejs.org/examples/webgl$_{mirror}$
4) https://threejs.org/examples/webgl$_{morphtargets_horse}$
5) https://threejs.org/examples/webgl$_{multiple_views}$
6) https://threejs.org/examples/webgl$_{shaders_sky}$
7) https://threejs.org/examples/webgl$_{shaders_ocean}$
8) https://threejs.org/examples/games$_{fps}$
9) https://threejs.org/examples/misc$_{controls_transform}$
10) https://threejs.org/examples/css3d$_{periodictable}$