COMPUTER GRAPHICS - MINI PROJECT

IMT2018026 Gayathri V.
IMT2018040 Manasa Kashyap
IMT2018047 Nachiappan S K
May 24, 2021

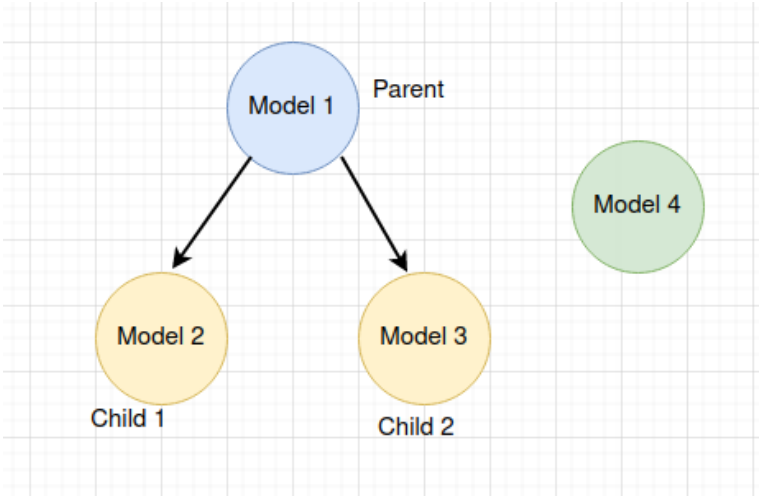
1 Problem Statement

This project requires us to do the following:

- 3D rendering with animation
- Camera setup, different kinds of lighting
- Texture mapping on different kinds of objects
- Modeling a dynamic scene with a Scene Graph
- Handling collisions

2 Scene and Animation

A scene graph is a data structure commonly used to represent hierarchical relationships between transformations applied to a set of objects in a three-dimensional scene.



The above diagram shows the relationship between the 4 models in the scene. 3 of the avatars have a parent-child relationship (children move relative to the position of the parent), while the fourth avatar is an independent node. This choice was made in order to allow one of the avatars to independently move and attach itself to different objects in the scene, without affecting the positions of the other avatars.

1. Movement and Animation: As shown above, since the models are linked through a scene graph, we only change the position of the parent node and the other independent node. The children will follow suit and move in the exact same direction as the parent.
2. Collision Detection: We have done collision detection using bounding boxes. To demonstrate this, we have included both static and dynamic obstacles. Static obstacles are stationary spheres that are present in the scene, and dynamic ones pop up at different time intervals and positions. When the arrow keys are pressed, the object is translated. We use *THREE.Box3.setFromObject(object)* to create a bounding box around the object as well as obstacles, and check if any obstacle intersects with the new position of the avatar. If collision is detected, the object is moved back to its original position.

3. Attaching to a moving object: Like stated above, the independent avatar (model 4) is used for this process. One of the obstacles present in the scene is in continuous motion. Using certain keys, the user can place the avatar over this moving obstacle, make him jump up and down, as well as rotate while being attached to the obstacle. The model can then jump off the mesh (detach itself) and stop in the same position.

The keyboard mappings for these are mentioned at the end of the report.

3 Texture Mapping

Texture mapping refers to the process of wrapping a given 2D texture sample around a 3D mesh. There are different kinds of mapping available - planar mapping, spherical mapping, cylindrical mapping etc. Each method works best for specific types of meshes and helps minimise distortions.

1. Background texture: This is a simple planar mapped texture of a 2D checkerbox image. On pressing the key 't', the texture sample changes from a checkerbox to a world map.

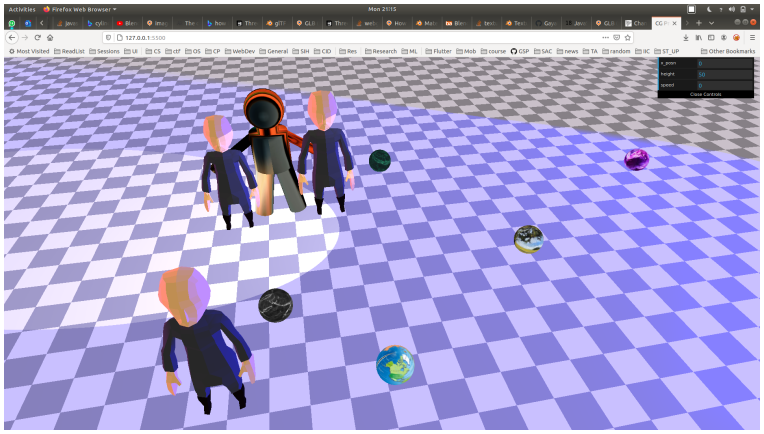


Figure 1: Background mapping using planar mapping with simple checkerbox

2. Spherical Mapping: This kind of mapping is applied to models that are sphere-like, for example, a tea pot. The texture is first mapped to a sphere, which is used as an intermediate object in the mapping process. The texture is then mapped from each point on the sphere to a point on the tea pot. Since our obstacles are sphere-like meshes, we have used spherical mapping on them.

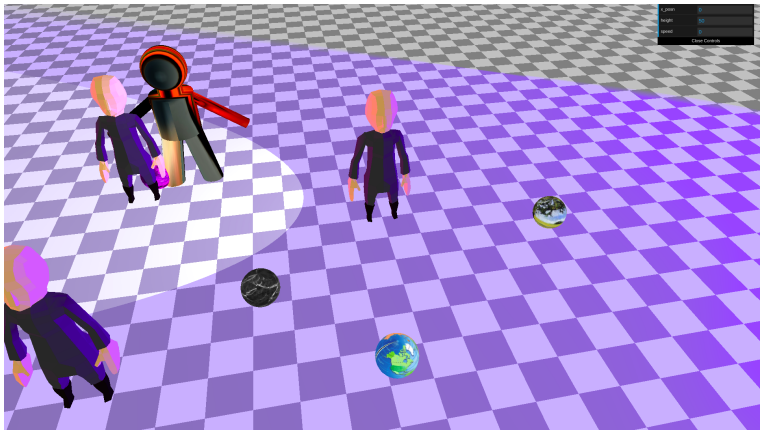
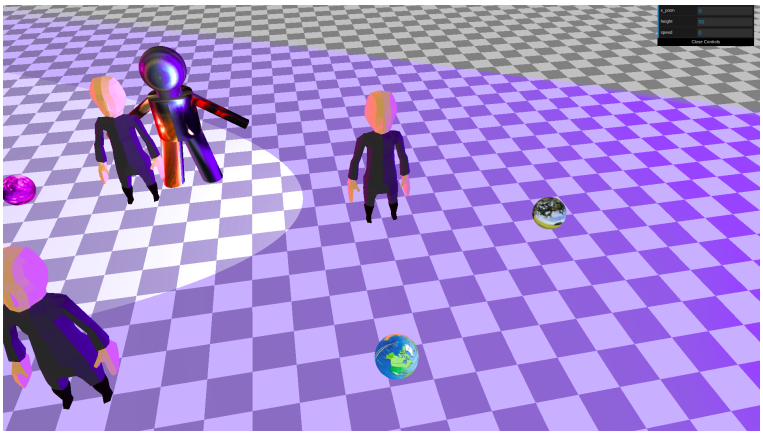


Figure 2: Spherical mapping on a Robo Man from front camera angle

3. Cylindrical Mapping: This is similar to the above, where instead of using a sphere, we use a cylinder as an intermediate object. On pressing the key 'm', we can switch between planar, cylindrical and spherical mapping. The UV map for an cylindrical mapping is generated and created using blender and then the map is converted and associated with an material and the object with the material is exported as a single gltf file so that the image which is inputted in the loader gets the uv map of the material present.



4 Lighting

There are 3 kinds of lights in the scene, each with a distinct colour:-

- Street Lights: These are a set of lights which are fixed to the ground and illuminate a specific region in 3D space. Since street lights are like spotlights in nature, we have used THREE.SpotLight to add 6 blue lights on the right side of the scene. The lights are equidistant from each other sharing the same x and y coordinates. The angle of the spotlight has been set to 30 degrees.
- Search light: Search light can be thought of as a torch which we hold in our hand and continuously point to a moving object. This, again, comes under the category of spotlights. The search light stays fixed vertically above the center of the scene. The target of the searchlight is the leader, and hence he is always illuminated under the white spotlight.
- Headlight: This is a light attached to one of the avatars, and illuminates different regions of the scene as he moves. Although a spotlight could have been used for this purpose, we decided to experiment this with directional light. The position of this pink light coincides with the center of the leader avatar, and illuminates along the positive x, y, and z axes.

By default, all the 3 lights are turned on. To toggle between on and off states, we can use the keys f, g, h for street lights, headlight and searchlight respectively.

5 Camera

For this project we have implemented three cameras - the default (stationary) camera, an overhead drone camera and a camera attached to one of the avatars. The three camera views can be toggled with the key 'c'.

- Stationary Camera: This is the default camera view, created using *THREE.PerspectiveCamera*. Its position is set at (-10, 40, 50) to give a tilted view of the scene, while its *lookAt* (target) is fixed at the origin.
- Drone Camera: This is an overhead perspective camera positioned at 50 units on the y-axis. The *lookAt* of this camera is also fixed at (0,0,0). The drone continuously revolves overhead in clockwise direction and gives us a birds-eye view of the scene. User controls have been provided on the top right corner of the application to adjust the drone height (position on the y-axis), move it along the x-axis and change the drone speed. By default, the speed is set to 0.
- Avatar Camera: This camera is attached to the avatar and simulates his view of the scene. It is similar to the

above two, other than the fact that we keep track of the movement of the avatar and apply the same translations to maintain the position and direction of the camera.

6 Keyboard Mappings

- Four arrow keys are used for translation about the x and y axes. Comma and full stop are used to move in the positive and negative z directions respectively.
- 'j' is used to attach the avatar on to the moving sphere. Clicking 'j' again will detach the object.
- 's' key is used to jump while being attached to the moving sphere. Clicking 's' again will stop the jumping.
- 'r' is used to make the avatar rotate while being attached to the sphere. Clicking again will stop the rotation.
- 't' is used to change the texture sample of the background.
- 'f', 'g' and 'h' are used to toggle between on and off states of the street light, headlight and searchlight respectively.
- 'c' is used to change the viewing camera.
- 'm' is used to switch between planar, cylindrical and spherical texture mappings.