Computer Graphics Assignment 4: VR Animation and Textures

Aarushi Goenka IMT2018001
Ayush Mishra IMT2018013

1 Objective

This is intended to be a simplified VR application, and to help understand this from a CG perspective.

2 Approach

For this project, we decided to take the scene of an airport with airplanes as our objects. We have an image of the sky with clouds as the background. For the ground we have an image with the texture of a runway and grass around. We have an ambient lighting and a directional lighting in the scene. For the obstacle we have a stone in the middle of the runway.

3 Scene and Animation

We define a user for our screen. It is in the shape of a human being. It is a gltf model which we have imported from the internet. We can move our avatar with the help of the arrow keys. We can also make it jump by pressing the z key. The leader plane moves in a fixed math in our scene. It is an infinite path where we keep moving straight, avoiding obstacles and taking a u-turn when we reach the end of the runway.

- **To move in a straight line :**
  This the most basic of the three movements . For this we simply get the direction in which our object is looking and then move our object

in that direction by a certain amount. We get the direction in which an object is looking using the function obj.getWorldDirection().

- **To take a u-turn :**
  Now for this part in our movement we take a u-turn at both our end-points so that we keep on going in an infinite path. Once we reach an end-point we start moving in a circular path of radius 10 around a fixed point and then once the circle is complete we reach the position from where we started our circular path but facing in the opposite direction. Now after this we again move forward in a straight line and further ahead in an infinite loop.

. **A)** We have 4 planes in our scene. All of them are the child of the scene. All of them are placed in a single line. We have defined a path for only the first plane. The second plane follows the first plane and each plane after it follows the one in front of it. To implement this, we created an object3D at the end of the plane and one at the front of it. We added these objects as the child of the plane. To implement the following function, we defined a vector from the front of the follower plane to the back of the leader plane. This vector would be the direction in which the follower plane is supposed to move. We then normalised this vector and moved the follower plane in this direction. We also added the lookAt function which makes the follower plane to look at, i.e., to rotate to face the leader plane. We did this for all the 3 planes.

. **B)** We only implemented collision avoidance for static obstacles. This part can be divides in 2 subparts.
First, at each step, we check if the distance between the obstacle and the place is less than or equal to 12 units. If it is we consider that it would collide in the next few steps. At this point we call the escaping object function which is the next subpart on avoiding the collision.
Now, once we know of the collision we will take a semi - circular path around the object. We take the centre of our radius to be the centre of the obstacle and our radius to be 12 units for our semi- circular path. After completing the semi - circular path we again move ahead in a straight line. For moving in a circular/semi-circular path we used the polar-coordinate system (rcos0,rsin0) to move in a circular path and since we knew the centre and radius of the circle it was easy to

compute the (n+1)th coordinate from our nth coordinate by changing my theta by a minute factor.

. **C)** We can also attach our avatar to any one of the moving objects. When we press 's', the avatar checks whether it has any plane near it. If there is no plane in the radius of 8 units, the avatar does not jump. However, if there are more than one plane in a distance of 8 units, we find which is the closest plane and the avatar jumps on that. To jump on it, we detach the avatar from the scene object and attach it as a child to the plane object. To get down from the plane we remove the avatar from plane object and attach it as a child to the scene object. While our avatar is on the plane, we can jump up and down or even turn around while moving along with the object. Since the avatar is a child of the plane, any movement is with the plane as the reference. To turn our avatar, we just rotate it by an angle when we press a button.

4 Appearance and Textures

We have added textures to our floor, the sky, the planes, the building and the rock. The texture on the planes have the wrap property set to repeat wrapping so it looks like they are covered with texture of metal. The ground is a single texture image. The building is defined with the default texture. It defines that each face of the building tries to map to the image and hence the image is repeated on each face of the building.

. A) When we press the 't' button, we can change the texture of our planes. We load both the textures in our file and we have a global variable which keeps track which texture should be applied. When we press the key, we change the value of the variable and set the texture to another texture. This way we can easily alternate between the two texture we have.

. C) We implemented the function to set spherical map for the building. We also have a function to set the map as the planar map. Before applying transformations to the building, we call a function to set the UV values for all vertices and store this array since we need to calculate the textures in the local space of the object. The formula for the planar map:

U = z;
V = y;

The formula for the spherical map:
U = (atan2(x, y) / π*0.5)+0.5 ;
V = 0.5 - (asin(z) / π);

When we press the button to change the textures the function applies the corresponding array to the UV attribute of the geometry of the object, and then we update it.

5 Lighting

We have an ambient lighting and a directional lighting in the scene along with the following lights.

. A) We have two street lights, both of which have a point lighting at the top of the street light object. It illuminates a small region around it and on the ground.

. C) We add a blue colored light to the front of the leader plane. We add this light as a child to the plane and set it's relative position so that it looks like it is coming from the front of the leader plane.

6 Camera

. A) We have our initial camera at a fixed position observing the whole view. We fixed this camera so as to be able to see all the planes and the building and the other functionalities.

. B) We added a camera and made it a child of our avatar. We then set it relative value to make it look like we are watching through the eyes of the avatar.