CS606: Computer Graphics / Term 2 (2020-21) / Programming Assignment 4
International Institute of Information Technology Bangalore

Announcement Date: Apr 10, 2021
Submission Deadline: 11:59 pm IST, Apr 27, 2021
Demo and walk-through: Apr 28-30

Summary: 3D rendering with animation, camera setup, and textures.
Learning Objectives:

- 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
- Basics of a VR application

Assignment: VR Animation and Textures

This is a mini-project to be worked on in teams of 3 (or as discussed with the instructors). This is intended to be a simplified VR application, and to help understand this from a CG perspective. This also brings together various aspects of CG that have been used in earlier assignments.

The description below provides the outlines of a storyboard and the broad guidelines of the features expected in the assignment. You can design any reasonable scene, objects, and animation that conform to these guidelines. The assignment consists of 4 major parts (weightage of each part for grades are as shown):

1. Designing the scene and animating it in a controlled manner (40%)
2. Changing the appearance of objects by manipulating textures (20%)
3. Producing varying lighting effects (20%)
4. Camera control and changing views (20%)

Each of these parts has 3 sets of features/capabilities (labelled A, B, C). You should implement at least 2 of the features in each part. The work will be assessed both on the basis of completeness of the functionality as well as the quality of the rendering. Extra credit will be given if all 3 features of a part are correctly implemented.

In the following, the term “user” refers to the person running the program and manipulating controls such as keyboard and mouse, and hence triggering actions in the scene.

At the same time, as a VR-like application, we also have an *avatar* of the user as part of the scene. Since we do not have HMD's etc, the user also controls the movements of the avatar, to simulate a VR setting.

Scene and animation:

A set of objects are moving as one or more groups in a defined path. You join the scene as a VR avatar. You can choose any shape for your avatar. You should be able move around anywhere in the scene (simple turn left or right and move/stop controls)

A. Each group has a leader, and the others follow such that each object follows only one other object in the group. E.g. the coaches of a train, a convoy of trucks, a line of ants, a flight of birds etc. The leader decides the actual path of motion, and the others adjust accordingly.
B. The path has both static and dynamic obstacles. When the leader comes close to an obstacle, it changes its path to avoid and move past the obstacle without colliding with the obstacle(s). Think of a road with potholes/rocks and vehicles that suddenly appear in your path. Note that the avatar can also be an obstacle to the other scene objects and vice versa
C. Your avatar can attach itself to one of the moving objects (E.g. sitting on one of the trucks). In addition, the avatar can also jump up and down or turn around while moving along with the object.

Appearance and Textures

All the objects in the scene, including the ground, are rendered using textures. Use any reasonable texture images. Please use texture images that will show the correctness of the texture maps. Thus, use images such as photos of people/animals/natural scenes, world maps, etc, and avoid featureless textures like sand, wood, brick, fur etc. Use at least 3 distinct texture images across objects in the scene

A. Based on user control, the textures of objects can be changed. That is, the image associated as the texture for an object can be changed on user input. You can use either the default texture coordinates for the object (if it exists) or a simple planar map
B. User controls enable switching to cylindrical maps for some subset of objects
C. User controls enable switching to spherical maps for some subset of objects

Lighting

The scene is illuminated by multiple lights. The user should be able to switch any of these sets of lights off or on. Adjust the intensity of the lights so that all the effects are clearly perceptible.

A. One set of lights is fixed to the ground, and each light illuminates a small region around it (e.g. streetlights).

B. Another light is at a fixed location, but tracks one of the moving objects (e.g. a searchlight).
C. Another light third is attached to one of the moving objects (e.g. a car headlight or a flashlight in a person's hand).

Camera

The scene is observed by 3 cameras:

A. One camera is at a fixed position observing the whole scene. This is the default view.
B. An overhead camera is mounted on a drone, with the camera facing forward and down. The user flies the drone, by adjusting its height (move up or down), turning it (left or right) and adjusting its speed.
C. Another camera is attached to your avatar and simulates the avatar's view of the scene.

The application has only one viewing area, and the user can choose which camera feed should be shown in the view area. You could also set up multiple view areas to simultaneously render the views of the different cameras. You can see examples here, though many of these are more complex than we need here.

(Note: for an actual HMD, we would need stereo rendering, which you can simulate by attaching 2 cameras to the avatar's head, and rendering each to a different half of the view area).

Implementation Notes:

1. Use a Scene Graph to model the scene. This should capture the positional relationships between objects, if they exist. Each scene node contains information on the transforms needed to move the object relative to its parent in the scene graph
2. The process of animation involves recomputing the relative transforms from each successive frame that is to be rendered.
3. Thus, you structure the code as a two-pass process: the first pass of the scene graph is used to update the transforms, and the second pass is used to render the scene.
4. Use a simple collision detection mechanism, such as intersection of bounding boxes.
5. Use positional information from objects in the scene graph to control the position and orientation of lights and cameras.
6. You can find WebGL specific information at
    a. Sample SceneGraph design and usage. You should try to design something similar for your purposes.
    b. Textures, including sample code
    c. Animation - specifically invoking requestAnimationFrame

Deliverables:

Submissions must be made on LMS. One submission per team.

1. The deliverables in a zipper folder must include the following:
    a. a source code folder
    b. a folder with screenshots
    c. a demo video not longer than 5 minutes. The video should show switching between the different camera views.
    d. A brief writeup on the design - scene graph organization, how the position and orientation of lights and cameras are calculated, and computation of animation including collision detection/avoidance. This should be not more than 3 pages.
2. More details on the submission are given in the course handbook on the LMS course page. If the deliverables are larger than the maximum allowable size on LMS, submit a text document with a repository URL (Google Drive, OneDrive, etc.). Care must be taken to not change this repository until grading is done.