CS606: Computer Graphics / Term 2 (2022-23) / Programming Assignment 2
International Institute of Information Technology Bangalore

Announcement Date: Feb 21, 2023
Submission Deadline: 11:59 pm IST, Mar 19, 2023

Summary: Rendering and manipulation of 3D models.
Learning Objectives:

- Creating 3D models (using a modeling tool)
- Importing 3D mesh models
- View transformations
- Manipulating 3D objects using transformations
- Computing animation paths and transforms
- Using the mouse to manipulate the scene

Assignment: Manipulating 3D models

This assignment is a simulation of a simple game: a playground is shaped as a regular ***n***-sided polygon. ***m*** players ($m < n$) are positioned at different corners of the polygonal ground. One of the players is chosen (by the user) as the "catcher". The catcher moves to any of the other corners of the ground in a straight line. A corner can accommodate at most one player at a time. So, if there is a player at the corner to which the catcher is moving, that player should move to one of the other unoccupied corners.

To illustrate the motion of the players, we make them move through ***k*** steps on the path from the starting corner to the target corner, each step being taken on a user command.

The game can be viewed in a graphics window from different camera angles.

The features to be covered in the assignment are listed below. Please see the section **Implementation Notes** at the end for suggestions on the design and implementation.

Part I
<u>Create a set of solid models</u> (at least 3) using Blender. Each model should incorporate at least one of the modelling operations such as boolean, cut, sculpt etc, starting with primitives (cube, cylinder, cone etc). The models should be sufficiently different from each other. Save the models to files using a common format such as .ply, .stl, or .obj.

(As part of the submission, you should include Blender screenshots of the modeling steps used for each of the objects).

Part II

Implement a WebGL program with the following features:

The steps below should be triggered using keyboard, mouse or UI events. It should be possible to perform steps E to I in any order, and any number of times.

A. Select n, m - the number of sides of the polygonal field, and the number of players.
B. The graphics window can be toggled between 2 modes:
    a. Mode-1 “Top View”: Camera is looking along the z-axis at the x-y plane of the scene.
    b. Mode-2 “3D View”: Camera looking at the origin of the scene from any direction.

    Define a key binding that will allow switching between these modes at any time during the game. Both modes should support the ability to zoom in and out.
C. Render the n-sided polygon representing the field. Assign numbers 1 to n for the corners, in any sequence.
D. Import the 3D models generated in part I. Each model is read in from one file, in one of the common formats. Create multiple instances of objects if needed, such that you have **m** objects, representing the players. For each object, identify a direction, relative to its original position, that is the “forward” direction.
E. Start the game by positioning the players at randomly chosen corners. Each model object is assigned a different color.
F. Select the “catcher” by selecting one of the players (by “picking” or “clicking” on or near the player). This player is now the catcher.
    a. The catcher is highlighted by being rendered in a distinct color.
    b. The catcher chooses the destination corner at random.
    c. the catcher turns (rotates about its axis that is perpendicular to the ground) such that its “forward” direction points to the destination corner.
        i. Draw an arrow positioned at the catcher to show this direction. You could represent the arrow by a cylinder+cone, positioned appropriately.
G. The user “drags'' the catcher (using the mouse) towards the selected corner in one or more steps. The catcher moves in a straight line towards the destination, using the distance dragged by the user as an approximate measure of how far to move. You can also assume that the user “drags” the catcher in the approximate direction of the destination corner.

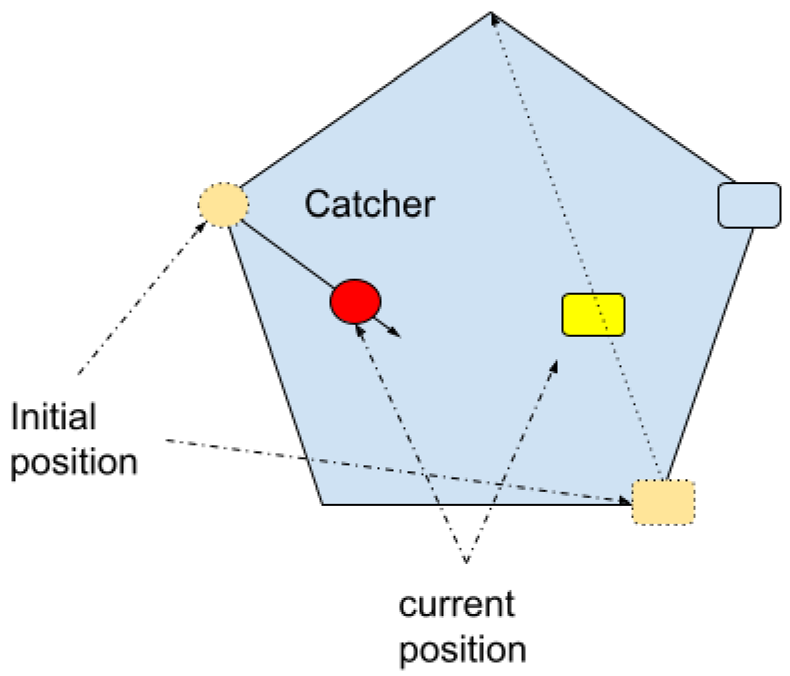
H. At the same time, the player at the catcher's destination moves towards a different unoccupied corner. This player should move towards the destination by an amount proportional to how far the catcher has moved (in its path from source to destination). Thus both the catcher and this player reach their respective destinations at about the same time.

I. The catcher. once selected, can be rotated about the object's x-, y- or z-axis using key bindings. Assume that no rotations are possible once the catcher turns toward the destination and until the destination is reached.

J. The size of the catcher can be scaled up or down at any time, again using key bindings.

K. In 3D View mode, use the mouse to rotate the camera about one of the x/y/z axes and the origin of the scene.

The figure below illustrates a snapshot of the game.



Implementation Notes:

1. Model: Create solid models using Blender. The models should clearly demonstrate at least a subtraction or intersection operation. Save the models to a convenient formation such as .ply or .obj. We will point you to sample code for importing ply/obj files to WebGL.
2. Direction Arrow (for catcher): You can create an instance of a cylinder+cone, appropriately positioned, using Blender, save that out as a model file, and import

that into your WebGL program. You can then copy and rotate/translate as needed to create the three axes.

3. Object Transformations: All object transformations should be implemented by generating/modifying transformation matrices - one each for scale, rotate, translate - of the respective object, and applying these during the render process. Rotations are to be done using Euler rotations about three principal axes. DO NOT directly modify the coordinates of the objects in the application. Use the vertex shaders for any transformations.
4. User Interactions: Use mouse actions for dragging the catcher. Define keyboard mappings or develop UI widgets for the other interactive steps. For example, the up and down arrows can scale the object, X/Y/Z can rotate the object about the appropriate axis. The key bindings should be kept unique, so that these operations can be done in any order and may be interleaved.
5. Camera Manipulation: When in “3D View”, the mouse can be used to rotate the camera. Model the camera manipulation as a “trackball”, and use quaternions to implement the changes to the view matrix. Also, implement a mechanism to zoom in and out of the scene.
6. It would be preferable to define classes to encapsulate the properties of the players - position, orientation, color, geometry etc. Thus, you would have a Game class, a Player class and any other helper classes.
7. Explore the use of the Model-View-Controller design approach, so that your code is easier to manage and extend. In this, you would clearly separate the Model (the geometric objects and their state), the Controller (all the logic about the actions, the movement etc), and the View (rendering the scene) and managing user inputs.
8. Design your game such that the players are autonomous, and have to figure out for themselves if they are the target for the catcher. That is, the players know the identity of the catcher, can query it for its position and orientation, and need to decide if they need to move. You can assume, for example, that the Game informs all players about each “move” command (step G).

<u>Deliverables:</u>

Submissions must be made on LMS.

1. The deliverables in a zipped folder must include a source code folder, a folder with screenshots, and a demo video not longer than 5 minutes, and a brief report describing what was done in the assignment, as well as answers to the questions below. The demo video should also show one of the objects being modelled in Blender. More details on the submission are given in the course handbook on the LMS course page.

2. 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.

Questions to be answered in the report:

1. What were the primary changes in the use of WebGL in moving from a 2D world to a 3D world?
2. How were the translate, scale and rotate matrices arranged and updated?
3. Describe at least 2 different approaches for using a mouse click to "select" an object in the scene.
4. Assume you had a large number of objects with the same original geometry (i.e. mesh, in this case). How would you organize the models so that you minimize the data stored?