CS606: Computer Graphics / Term 2 (2021-22) / Programming Assignment 2
International Institute of Information Technology Bangalore

Announcement Date: Feb 20, 2022
Submission Deadline: 11:59 pm IST, Mar 18, 2022

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

- Creating 3D models (using a modeling tool)
- Importing 3D mesh models
- Transformations of 3D objects
- Computing animation paths and transforms
- View transformations
- Picking model objects

Done

Assignment: Simple 3D Model Animator

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.

To be done

I. Create a set of solid models (at least 2). You can create them using Blender or other modeling tools. Each model should incorporate at least one of the modelling operations such as boolean, cut, sculpt etc. Save the models to a file using a common format such as .ply, .stl, or .obj

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 D to I in any order, and any number of times.

Done

A. Create a graphics drawing window. The window can be toggled between 2 modes:
   1. Mode-1 “Top View”: Camera is looking along the z-axis at the x-y plane of the scene.
   2. 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 animation

Binding Key for changing the mode is 'm'.

Done

B. Draw the axes of the scene (world coordinates). You could use a cylinder topped with a cone for each axis. Color the x, y, z axis with R, G, B respectively.
C. Import the three 3D models generated in Step A. Each model is read in from one file, in one of the common formats. Common Format: .obj
D. Render the model objects with their initial position at the origin. Each model object is assigned a different color. Assumption: Different colors for the shapes and the selected shape.
E. Picking objects. Objects are picked only in Top view mode
    1. Clicking on an object selects the model object where the mouse was clicked
    2. Highlight the selected object in a color distinctly different from those assigned in step D
F. Defining the animation path:
    1. Set the window in Top View mode.
    Assume the current position of the selected object is p0. Choose a path for the selected object to move, by selecting 2 more points, p1 and p2, by clicking in the window. The selected points need to be mapped from screen coordinates back to world coordinates. Since this view only provides a 2D view, use the x,y values as selected here, and assign an arbitrary non-zero value for the z-coordinate of the picked point.
    2. Generate a quadratic curve through these points that serves as the path of the moving object. Evaluating the points along this curve produces the points that the (center or origin of the) object moves through.
G. Animation:
    1. The selected object should move along a smooth curve that interpolates p0, p1, p2.
    2. The object stops when it reaches p2, and is no longer the selected object
    3. The speed of the moving object can be controlled - increased or decreased - using key bindings
H. When the object is stationary (i.e. at point p0 or p2), it can be rotated about the x-, y- or z-axis using key bindings
I. The size of the moving object can be scaled up or down when it is stationary, again using key bindings. For scale up or down
J. 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.

Implementation Notes:

1. Model: Create solid models using Blender or other modeling tool. The models should clearly demonstrate at least a subtraction or intersection operation. In addition to these 2 models, you can also import readily available mesh files such as at https://people.sc.fsu.edu/~jburkardt/data/ply/ply.html. We will point you to sample code for importing ply/obj files to WebGL.
2. Primary axes: It would be easiest to 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 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. We will be using quaternions in the subsequent assignments only.
4. Animation Path: Given the points p0, p1, p2, you can fit a quadratic curve through p0, p1, p2 by solving for **a, b, c** in the following. Note that a, b, c are vectors, so, you will solve for coefficients in each of the 3 dimensions.

   $\boldsymbol{p}(t) = \boldsymbol{a}t^2 + \boldsymbol{b}t + \boldsymbol{c}$

   $\boldsymbol{p}(0) = p0$

   $\boldsymbol{p}(1) = p2$

   $\boldsymbol{p}(t1) = p1$ *for some* $0 < t1 < 1$

   Assuming t1 to be 0.5

   We can assume that p0, p1, p2 are not collinear and not too close to each other.

5. User Interactions: Define keyboard mappings or develop UI widgets for the interactive steps. At a minimum, set up a keyboard mapping for each of the steps A to K, and using the key should produce the effect described for that step. For example, the up and down arrows can increase/decrease the speed of the object, left and right arrows can scale the object up or down, "X" and "x" can increase/decrease the rotation about the x-axis, etc. The key bindings should be kept unique, so that these operations can be done in any order and may be interleaved.

   Key bindings should be unique.
6. Camera Manipulation: For this assignment, we will restrict to rotations about one of the three primary axes at the origin. You can select the axis of rotation with a key or other way to select the rotation mode. Use the direction in which the mouse is dragged (right or left) to decide if the rotation should be a positive or negative angle

Give a list of key bindings

Deliverables:

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 or other tool. 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. To what extent were you able to reuse code from Assignment 1?
2. What were the primary changes in the use of WebGL in moving from 2D to 3D?
3. How were the translate, scale and rotate matrices arranged? Can your implementation allow rotations and scaling during the animation?
4. How did you choose a value for *t1* in computing the coefficients of the quadratic curve? How would you extend this to interpolating through *n* points (n > 3) and still obtaining a smooth curve?