CS606: Computer Graphics / Term 1 (2025-26) / Programming Assignment 2
International Institute of Information Technology Bangalore

Announcement Date: October 05, 2025
Submission Deadline: 11:59 pm IST, November 03, 2025 (Monday)

Summary: Rendering and manipulation of 3D models. (15% of final grade)
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

Assignment: Star System Generator

The generator must help the user create a star system with a star (spherical object) at the center, and planets orbiting around it. Figure 1 shows the rendering of a solar system, an example of a star system.

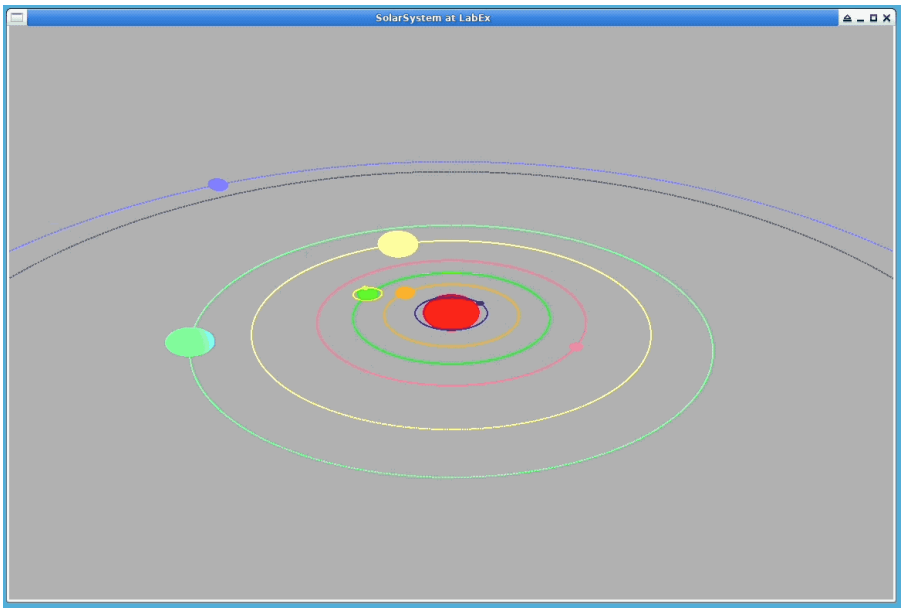

Figure 1: A solar system rendering, as an example of a star system. Image credit: https://labex.io/tutorials/cpp-creating-the-solar-system-in-opengl-298836

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.

I. Create a set of sphere and "regular solid" models (at least 3). Regular solids are 3D geometric shapes whose faces are all congruent, convex, regular polygons, and an equal number of faces meet at each vertex. These include cube, tetrahedron, octahedron, dodecahedron, and icosahedron. You can create them using Blender or other modeling tools. Save the models to a file using a common format such as .ply, .stl, or .obj.

A. A star is a sphere, and the 3+ planets can be a sphere or a regular solid. There should be at least two regular solids used as planets.

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.

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

B. Draw the axes of the scene (world coordinates with the star location as the origin). 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 or more 3D models generated in Step A. Each model is read in from one file, in one of the common formats.

D. Render the model objects with their initial positions

1. The star is always at the origin, and is stationary.
2. Each planet is at 0° angle from the x-axis in their designated orbits. The planets should be moving, but can be made stationary using user-input.

Each model object is assigned a different color.

E. Picking objects: Objects can be picked only in Top view mode. Clicking on an object selects the model object where the mouse was clicked. Highlight the selected object in a color distinctly different from those assigned in step D.

F. Defining the movement path:
    1. Set the window in Top View mode.
    Assume the current position of the selected object is p0. The planets can move (i.e., revolve around the star) along their respective orbits.
    2. Generate an elliptic curve for each orbit with the star at its center. The elliptic curves must be concentric, and only one planet can use one orbit. The orbits can have non-uniform spacing between each other as shown in Figure 1. All orbits must lie on x-z plane to simplify the setup, and they should not collide/overlap.

G. Automated Movement:
    1. The planets should move along a smooth curve at its designated speed.
    2. The speed of the moving object can be controlled - increased or decreased - using key bindings.
    a) The moving planet can also be made stationary using a key binding.

H. When the planet is stationary, it can be rotated about the x-, y- or z-axis using key bindings (i.e., rotation about an axis, but without revolution around the star. In this assignment, only axis-rotation or revolution can happen at a time, and both needn't occur at the same time.)

I. The size of the planet can be scaled up or down when it is stationary, again using key bindings. But it should resume its original size when it starts moving.

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.

K. There should be a key binding to add/delete a planet along with its orbit, without colliding/overlapping/clashing with any existing orbit. Deletion should not be allowed when the star system has only 3 planets. Thus, ensure the star system always has at least 3 planets.

Implementation Notes:

1. Model: Create simple solid models using Blender or other modeling tool. Learn what file formats are usable for interface files in a modeling-rendering paradigm. Learn more about ply files, for instance, which will be needed in future assignments.
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. There can be a key binding to show/hide the 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 **quaternions and virtual trackball**, thus allowing arbitrary rotations.
4. Movement Path: Use an equation of an ellipse to determine each concentric orbit and render using a piecewise line.
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.

a. The key binding must be intuitive. 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.
b. The key bindings should be kept unique, so that these operations can be done in any order and may be interleaved.

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 modeled 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 movement?
4. How did you ensure that they are no conflicts when adding/deleting a planet along with its orbit?

Rubrics:

15% of final grade – Marks out of 30

- 10 marks for functionalities
- 8 marks for code
- 4 marks for report
- 4 marks for video
- 4 marks for complete and neat submission
