Computer Graphics

Assignment 2

Aashish Vaswani
MT2024167

Instructor: Prof. Jaya Sreevalsan Nair

1. Brief Overview

This project renders a star system in WebGL with a stationary star at the origin and multiple planets on concentric elliptic orbits in the x–z plane. The system supports two camera modes: a 3D orbit camera and an orthographic Top View. The scene shows world axes, orbit paths, imported 3D meshes for at least three regular solids in addition to a sphere, and user controls for play or pause, speed, selection in Top View, and add or delete planet while keeping at least three planets present.

2. Implementation Summary

- Rendering: per-fragment Phong lighting, depth test, back face culling
- • Geometry: sphere for the star and planets, plus at least three regular solids from OBJ files
- • Cameras: perspective orbit camera and orthographic Top View with instant switching via a key
- • Interaction: pause or play, speed up or down, toggle axes and orbits, Top View picking, add or delete planet with orbit conflict checks
- • Transforms: scene graph with parent to child updates per frame; matrices assembled as T · R · S

3. User Controls

- V: toggle 3D and Top View
- Space: pause or play
- Up or Down: speed up or down
- A: axes toggle
- O: orbit paths toggle
- R: reset camera

- Mouse in Top View: pick a planet
- Buttons: add planet, delete planet

4. Answers to the Question

Q1. To what extent were you able to reuse code from Assignment 1?

-> I reused the application shell that includes the animation loop, time step control, keyboard input mapping, and the resize handler. Utility code for seeded random, clamping, linear interpolation, and a simple state machine was reused with minimal edits. I replaced the 2D transform helpers with 4×4 matrix math for 3D, added quaternion math for local rotation, and introduced a scene graph for hierarchical updates. The triangulation, density coloring, and obstacle logic from Assignment 1 were not reused since the current task focuses on 3D models and camera views.

Q2. What were the primary changes in the use of WebGL when moving from 2D to 3D?

• Coordinate space: moved from implicit 2D to full 3D with model, view, and projection matrices.

• Projection: added perspective and orthographic projections through explicit camera objects.

• Lighting: introduced normals, a normal matrix, and per-fragment Phong shading. Assignment 1 was unlit or flat.

• Visibility: enabled depth testing and back face culling to handle overlapping geometry correctly.

• Mesh handling: added an OBJ loader and attribute binding for positions, normals, and uvs.

• Picking: restricted to Top View and implemented screen to world mapping in orthographic mode with x–z hit tests.

Q3. How were the translate, scale, and rotate matrices arranged? Can your implementation allow rotations and scaling during movement?

The per object model matrix is M = T · R · S.

T moves the object to its current orbit point in world space.

R is a local rotation built from quaternions.

S is a local uniform or non-uniform scale.

During movement the assignment requires that the object stays at original size and without extra spin. I follow that by applying T and using R = I and S = I while the planet is revolving. When I pause the planet, the app enables local rotate about x, y, or z and local scale. On resuming motion the scale is restored to unit and rotation is cleared unless explicitly allowed. This matches the requirement that rotate and scale apply when stationary.

Q4. How did you ensure there are no conflicts when adding or deleting a planet along with its orbit?

I maintain a sorted list of occupied semi major axes a for all planets. When adding a planet the generator proposes a candidate a and an eccentricity e within bounds, then enforces a minimum spacing Δa from neighbors. If the candidate conflicts I search the nearest feasible gap using a binary search over existing a values and adjust within display bounds so both cameras can view the full orbit. I also clamp eccentricity to keep the pericenter above the previous orbit and the apocenter below the next orbit. When deleting a planet I remove its a from the registry and ensure the system still has at least three planets by preventing deletion when the count would drop below three.

5. Limitations and Future Work

- • Orbits are kinematic. There is no N body gravity.

- • Lighting is simple Phong without shadows.
- • OBJ loading is basic and does not handle materials.
- Future work can add instancing for moons, soft shadow mapping, and optional spin while moving under a separate rule if permitted.

6. Links

GitHub: https://github.com/aashishvaswani/Computer-Graphics-

Video Link : https://drive.google.com/drive/u/1/folders/152GZPRLxgJYE5eJMFNNHx4D9plKlTg_B

7. Screenshots

Figure 1: Default 3D view

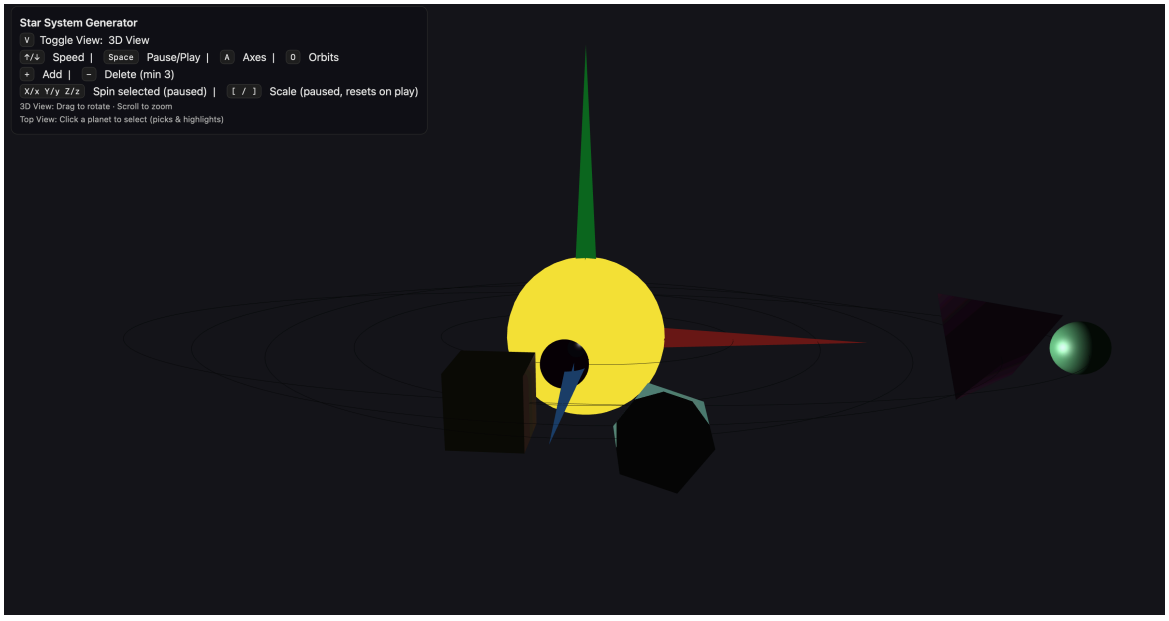

Figure 2: Orthographic Top View

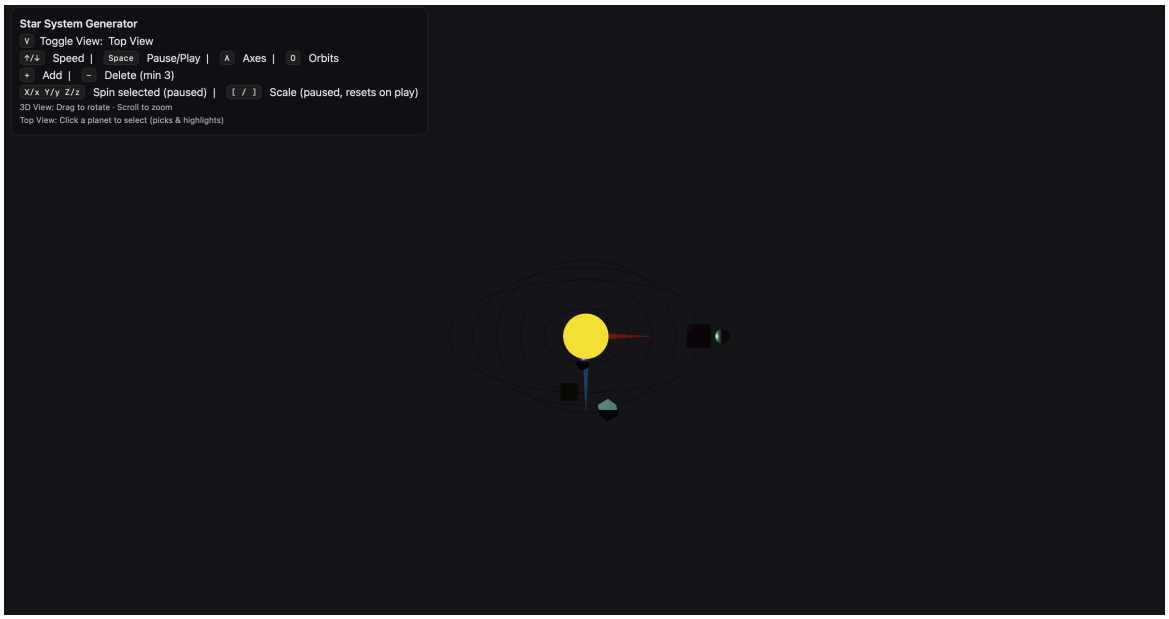

Figure 3: 3D View without axes and orbit

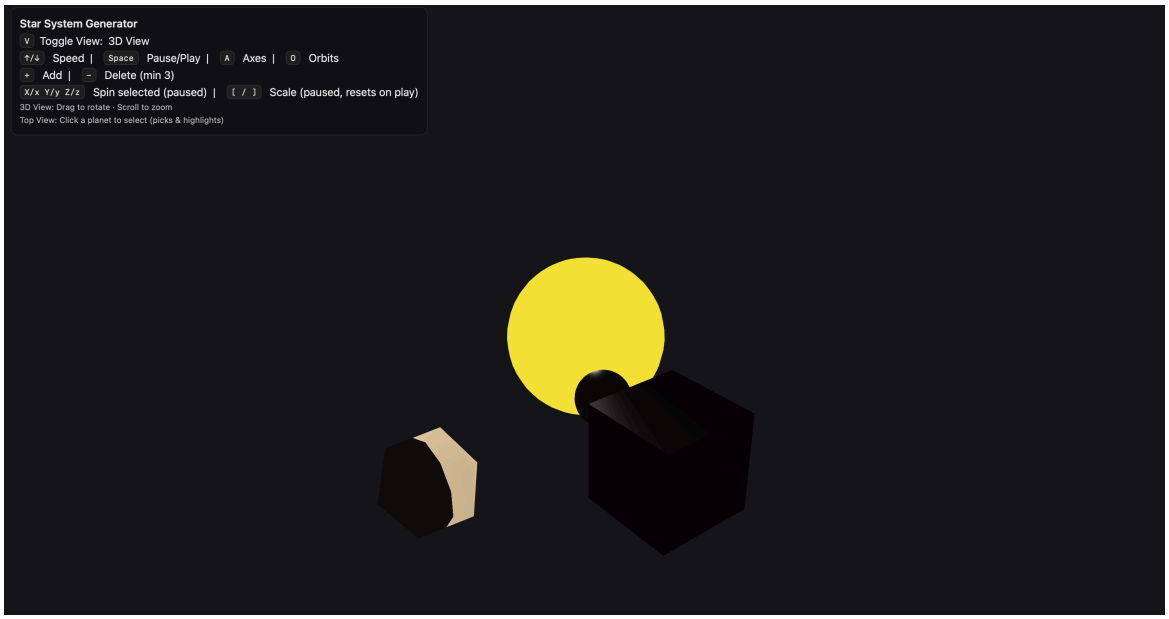

Figure 4: Increased the number of planets

Figure 5: Scaled up the size of selected planet