Report

Assignment 2

Team members' details

- Team member 1: Abhirami R Iyer
  Roll No: 112201001

- Team member 2: Nandhana Sunil
  Roll No: 112201008

Codebase

1. Creating spheres
   Spherical models are created by giving inputs as the number of segments and sectors the sphere should have. The `generateSphere` function generates the vertices, base colors for spheres and its normals into a vector. Then VAOs and VBOs are set up for these spheres and corresponding data are sent to these buffers.
2. Giving textures
   The `loadtexture` loads the texture given as an argument to this method. The `draw` method renders the sphere with the appropriate texture.
3. Rotation of sphere
   Rotation angles of each of the spheres are updated in each render loop. Using the `rotate` method from glm, the sphere models are rotated. This is applied to rotate the models of Sun, Earth and Moon.
4. Revolution of Earth and Moon.
   Revolution of Earth and Moon is attained through small incremental translations of earth and Moon in their respective elliptical orbits.

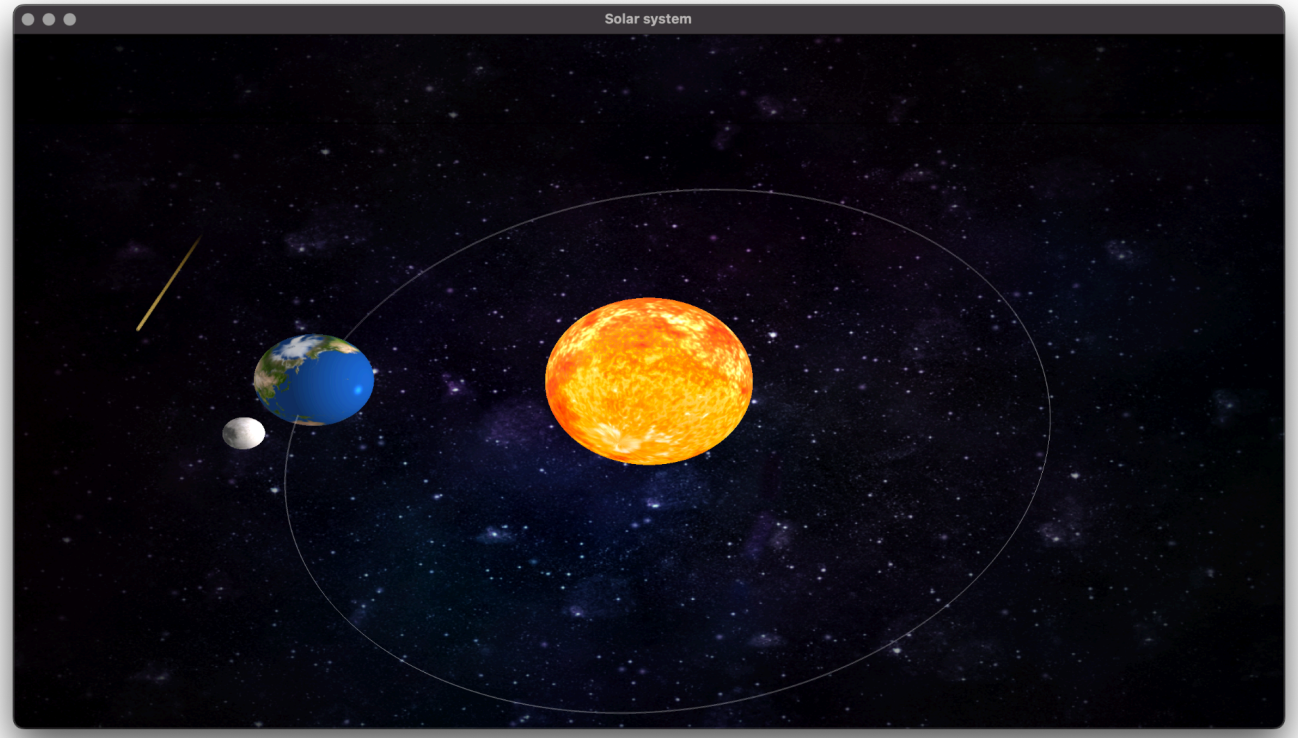
Implementation details

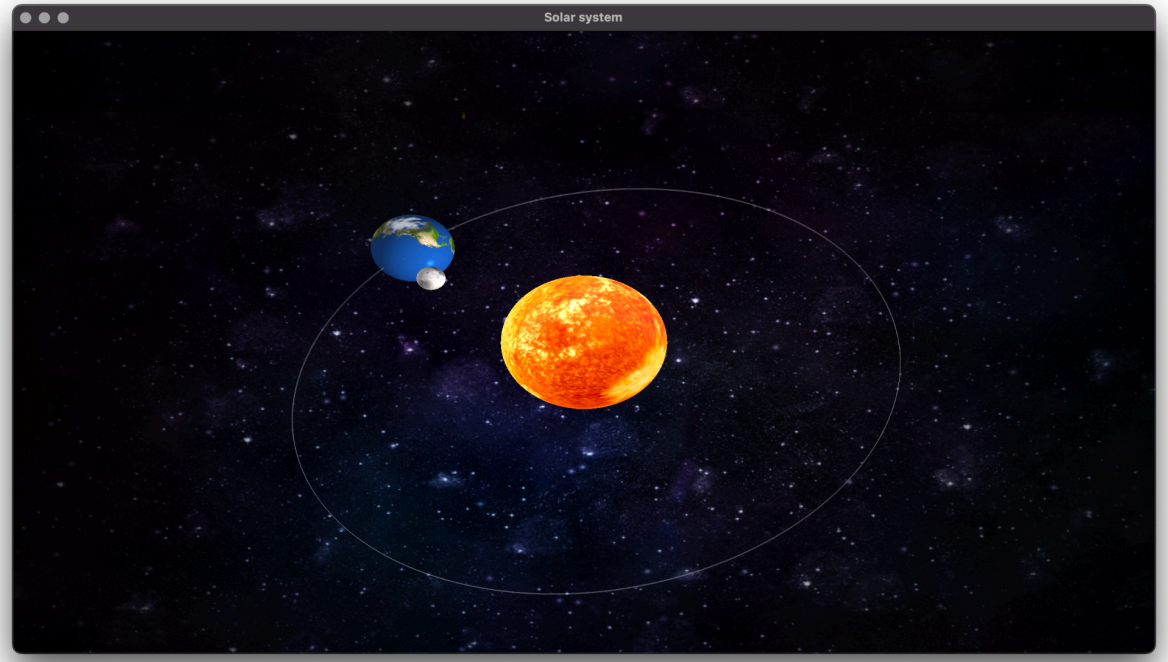
1. Sphere generation
   The method `generateSphere` method in sphere.cpp file generates the vertices, normals, texture coordinates and indices for the order of rendering, for a sphere when its radius, sector count

and segment count are given as input. The tessellation of the spheres can be adjusted by adjusting the sector count and segment count.
The x,y and z coordinates are calculated by geometric calculations involving sine and cosines of sector counts and segment counts of vertices. These coordinates are normalised to get the normals of these spheres.
The texture coordinates are obtained by mapping the vertices' data to the range [0,1]
When a new sphere is created, its vertices, normals and texture coordinates are calculated and sent to a separate VAO and VBO for it.

2. Rotation angle updation
   In the render loop, rotation angle of Sun, Moon, and Earth are updated. So, in the next frame, they all will be rotated by the incremented rotation angle.
3. Rendering background
   Background is rendered using special vertex and fragment shaders skybox_vs.glsl and skybox_fs.glsl. Background is rendered as a quad. Depth test is temporarily disabled, background is rendered, and then enabled again. Background is textured using a 2D image as a starry background.
4. Texturing
   Texturing of spheres is done by the sphere class. `loadTexture` method loads and maps the texture in the 2D image to the sphere.
5. Orbit Rendering
   Orbits are rendered using special vertex and fragment shaders - orbit_vs.glsl and orbit_fs.glsl.
   To get the coordinates of the orbit, the semimajor axis, semiminor axis and revolution angle are taken, and the vertices are calculated. These, along with the color values (white) are uploaded to a VAO and VBO and rendered using shaders mentioned above.
   The orbit model is also given a slight rotation to match the Earth's orbit tilt.
6. Orbital Tilt
   The Earth's revolution is given to simulate the actual Orbital tilt of the Earth. Same tilt is given to the orbit model also.
7. Shooting stars
   Shooting stars spawn at random locations and times to create an active environment. This is done in the shaders shooting_vs.glsl and shooting_fs.glsl. The function `smoothstep` is used to smoothen the trail of the shooting star.
8. Camera movements handling

9. Key call back
   Up, Down, Left and Right keys are used to control the position of camera in the scene.
   Also on pressing the P key, the motion of all objects will stop.
10. Mouse movements
   The scroll button movement in upward direction zooms in the camera towards the sun and up ward direction is for zooming out.




Solar system