Computer Graphics Project

—

Ahmed Adel Abudef
ID: 19015264

Code Flow

the code flow with a sequence of steps:

1. The code begins with the inclusion of necessary header files and the definition of a macro for the fill probability of asteroids.
2. The global variables and constants are declared, including the font selection, window size, spacecraft angle and position, collision flag, display list index, frame count, light properties, and more.
3. The **writeBitmapString** function is defined to draw a bitmap character string on the screen.
4. The **radians function** is defined to convert degrees to radians.
5. The **Asteroid class** is defined, with a default constructor and a parameterized constructor. It also includes getter and setter methods for the various asteroid properties and a draw method to render the asteroid.
6. The **setup function** is called to initialize the OpenGL environment and create the display lists for the spacecraft and asteroids. It also initializes the global array of asteroids with specific properties for each celestial body.
7. The **frameCounter function** is defined to count the number of frames drawn per second.
8. The **checkSpheresIntersection function** is defined to check if two spheres intersect.
9. The **collisionCheck function** is defined to check if the spacecraft collides with an asteroid.
10. The **drawScene function** is defined to render the entire scene, including the spacecraft and asteroids.
11. The **resize function** is defined to handle window resize events.
12. The **update function** is defined to update the position and properties of the asteroids, as well as check for collisions between the spacecraft and asteroids.
13. The **keyboard function** is defined to handle keyboard input events.
14. The **mouse function** is defined to handle mouse input events.

15. The **idle function** is defined to continuously update and render the scene.
16. The **main function** is called to initialize the GLUT library, create the window, set the display mode, and register various callback functions. It then enters the event processing loop.

This is a general overview of the code flow. Each function performs specific tasks related to initialization, rendering, collision detection, and user input handling. The main loop continuously updates and renders the scene based on user input and the elapsed time.

Snippets for Challenges

Animation:

```
arrayAsteroids[i].setAngleOrbit( angleO: arrayAsteroids[i].getAngleOrbit() + arrayAsteroids[i].getVelocity());
arrayAsteroids[i].setCenterX( x: arrayAsteroids[i].getDistance() * std::cos( x: radians( degrees: arrayAsteroids[i].getAngleOrbit())));
arrayAsteroids[i].setCenterY( y: 0);
arrayAsteroids[i].setCenterZ( z: arrayAsteroids[i].getDistance() * std::sin( x: radians( degrees: arrayAsteroids[i].getAngleOrbit())));
arrayAsteroids[i].draw( sun: i == 0);
```

ViewPorts Scissors:

```
glEnable( cap: GL_SCISSOR_TEST);
glScissor( x: 0,  y: 0, width, height);
glClear( mask: GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glDisable( cap: GL_SCISSOR_TEST);
glViewport( x: 0,  y: 0, width, height);
```

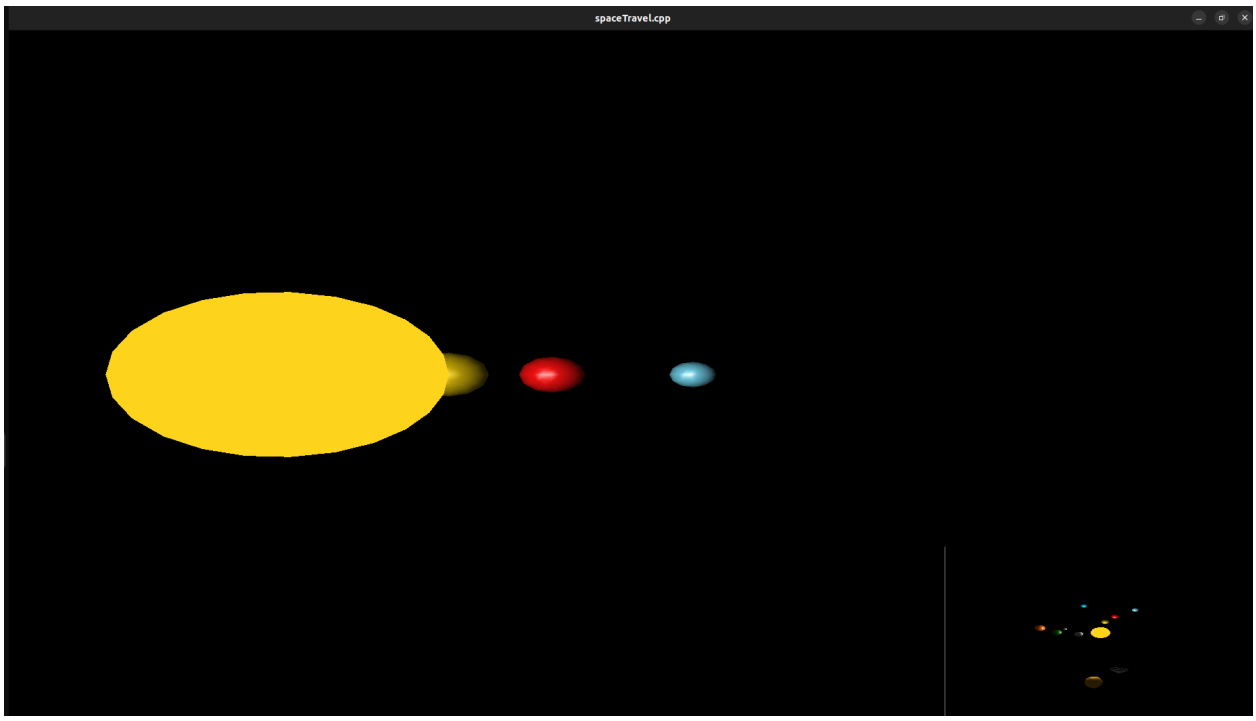
Moon:

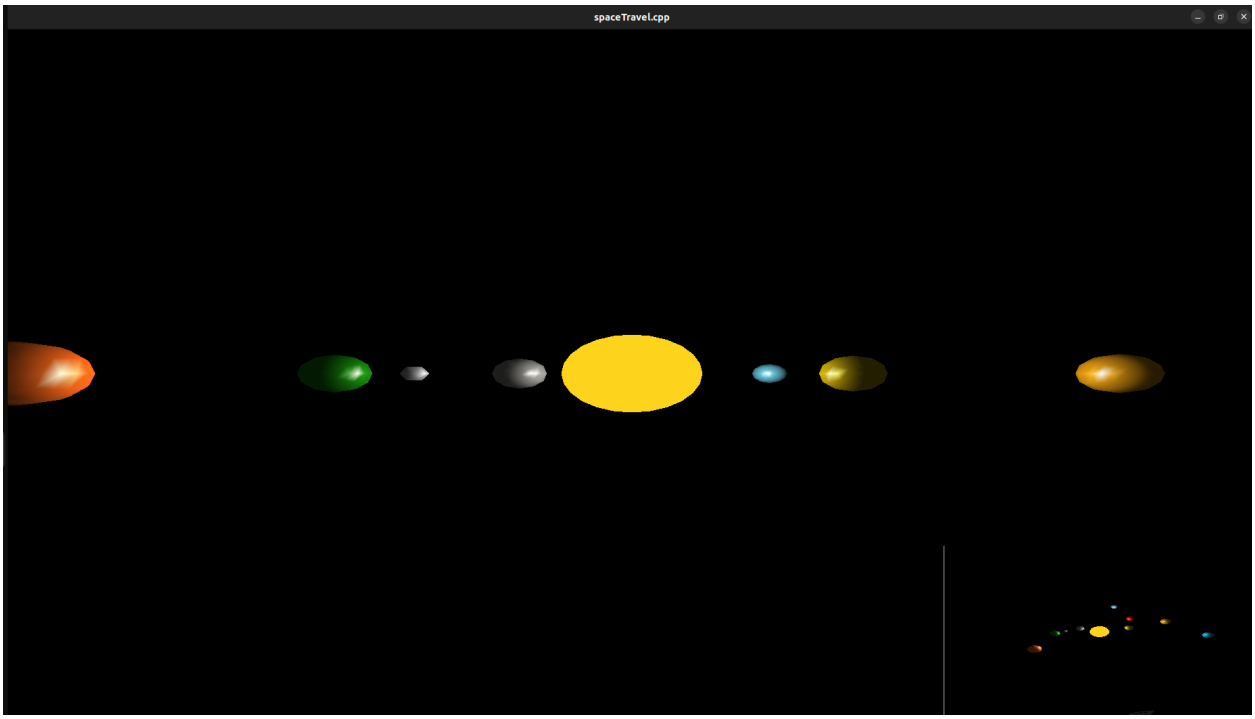
```
Asteroid earthMoon = Asteroid( colorR: 170.0,  colorG: 170.0,  colorB: 170.0);
earthMoon.setVelocity( v: 100 * relativeSpeed) ;
earthMoon.setAngleOrbit( angleO: 0);
earthMoon.setAngleRotation( angleR: 0);
earthMoon.setDistance( d: 5);
earthMoon.setRadius( r: 1);
earthMoon.setCenterX( x: earth.getCenterX() + (earthMoon.getDistance() * std::cos( x: radians( degrees: earthMoon.getAngleOrbit()))));
earthMoon.setCenterY( y: 0);
earthMoon.setCenterZ( z: earth.getCenterZ() + (earthMoon.getDistance() * std::sin( x: radians( degrees: earthMoon.getAngleOrbit()))));
```

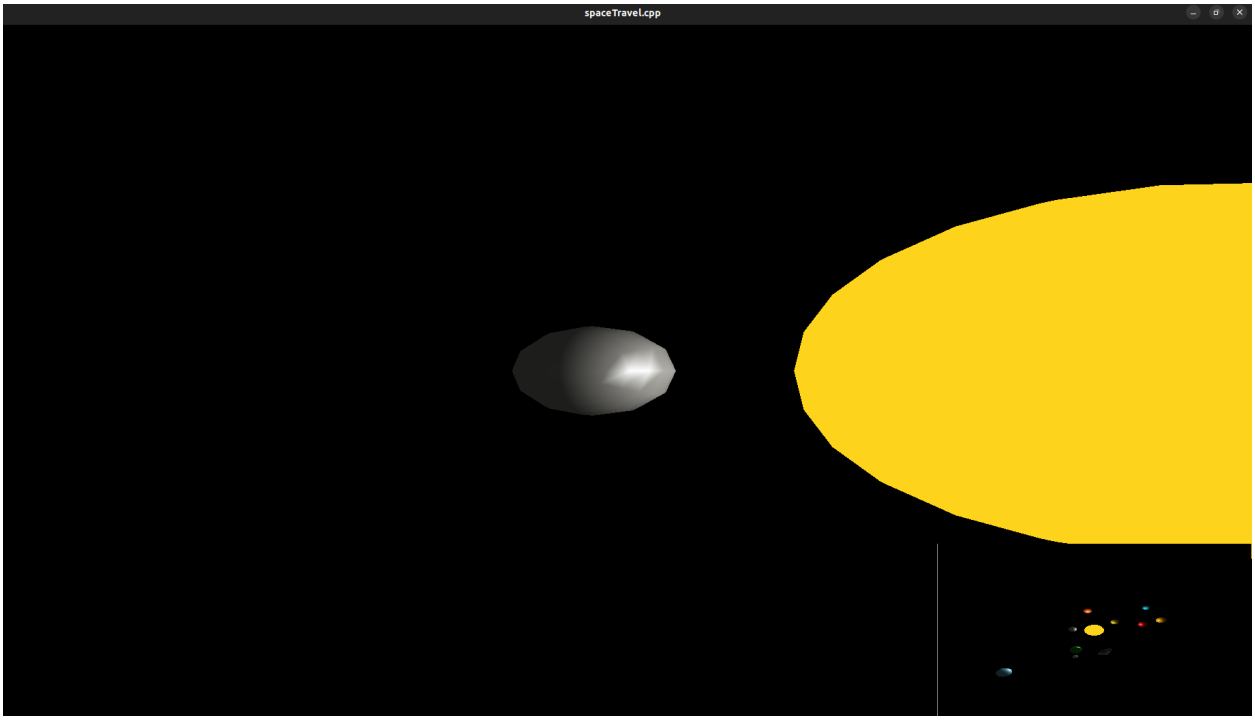
SpaceCraft Lookat:

```
// Locate the camera at the tip of the cone and pointing in the direction of the cone.
gluLookAt( eyeX: 10.0,  eyeY: 50.0,  eyeZ: -50.0,
           centerX: 0.0,  centerY: 0.0,  centerZ: 0.0,
           upX: 0.0,  upY: 1.0,  upZ: 0.0);
glLightfv( light: GL_LIGHT0,  pname: GL_POSITION,  params: lightPos0);
```

Sample Runs




spaceTravel.cpp


spaceTravel.cpp