American International University Bangladesh

Computer Graphics

Course Code: CSC4118

Fall Semester 2023-24

Project Report

[Journey Through the Solar System]

Under the Guidance of

Rahul Biswas

Lecturer

Department of Computer Science, FST

I. Group Member Details

Group Member Name	ID
1. Sakib, Md. Zahid Sadman	21-45904-3
2. Shakhawat, Mahmud Ibna	20-43909-2
3. Mahin, Md. Mosaddek Hossain	20-43905-2
Team Leader Name:	Sakib, Md. Zahid Sadman
Section:	C

II. Table of Contents:

Content	Page No
Introduction	4
Motivation	6
Diagram	7
List of Objects	8
Functions to Represent the Objects	8
Output Screenshot	9
Demo Link	10
Uniqueness of Our Project	10
Conclusion	10
Contribution	11
Reference	11

- ## Introduction:

  Idea: The animation will feature planets and the sun of the solar system, set against a black background to mimic the vastness of the cosmos. The planets will be seen revolving around the sun, creating a dynamic and realistic representation of our solar system. Meteor showers depicted as a streak of light will add another dynamic element. Distant stars will twinkle, adding depth and a sense of wonder to the animation.

  In this project, OpenGL is instrumental in achieving a range of functionalities, including rendering celestial bodies, applying textures, handling lighting effects, and enabling user interaction.

OpenGL Functionalities in Detail:

1. **Initialization (`init` function):**
   - **`glEnable(GL_DEPTH_TEST);`** Enables depth testing to ensure proper rendering of objects in 3D space.
   - **`glFrustum(-1, 1, -1, 1, 2, 10);`** Sets up a perspective projection frustum.
   - **`glEnable(GL_LIGHTING), glEnable(GL_LIGHT0), glEnable(GL_LIGHT1);`** Enables lighting and specific lights.

2. **Lights Configuration:** The project leverages OpenGL's lighting model to simulate the interaction of light with planetary surfaces. Lighting is enabled using `glEnable(GL_LIGHTING)`, and specific light sources, such as the sun, are defined using `glLightfv`.
   - **`glLightfv(GL_LIGHT0, GL_POSITION, Position_Of_Light);`** Sets the position of the first light source.
   - **`glLightfv(GL_LIGHT0, GL_DIFFUSE, Color_Of_Light);`** Sets the diffuse color of the first light.
   - **`glLightfv(GL_LIGHT1, GL_POSITION, Position_Of_Light1);`** Sets the position of the second light source.
   - **`glLightfv(GL_LIGHT1, GL_DIFFUSE, Color_Of_Light1);`** Sets the diffuse color of the second light.

3. **Material Properties Configuration:** Material properties like diffuse, ambient, and specular reflection are configured to enhance the visual representation of celestial bodies.
   - **`glMaterialfv(GL_FRONT, GL_DIFFUSE, material_diffuse);`** Sets the diffuse reflection color of materials.
   - **`glMaterialfv(GL_FRONT, GL_AMBIENT, material_ambient);`** Sets the ambient reflection color of materials.
   - **`glMaterialfv(GL_FRONT, GL_SPECULAR, material_specular);`** Sets the specular reflection color of materials.

- **`glMaterialf(GL_FRONT, GL_SHININESS, 0.0f)`:** Sets the shininess property of materials.

4. **Camera Configuration:** The `gluLookAt` function is crucial for defining the camera position, look-at point, and up vector, allowing users to observe the solar system from different perspectives. This function facilitates the creation of immersive views, including close-up views of Earth and distant views of the entire solar system.
    - **`gluLookAt(0, 20, 0, 0, 0, 0, 0, 0, 1)`:** Defines the camera position, look-at point, and up vector for the initial view.

5. **Texture Loading (`LoadTexture` function):** The `LoadTexture` function utilizes the STB image loading library to load texture images for celestial bodies. The `glGenTextures` and `glBindTexture` functions are employed to create and bind textures, enhancing the visual realism of planets like Earth by applying detailed surface textures.
    - **`stbi_load("earth_texture.jpg", &width, &height, &channels, 3)`:** Loads the Earth texture image using the STB image library.
    - **`glGenTextures`, `glBindTexture`, `glTexParameteri`:** Generates and binds a texture, sets texture parameters.

6. **Drawing Earth with Texture (`drawEarthWithTexture` function):**
    - **`glEnable(GL_TEXTURE_2D)`,`glBindTexture(GL_TEXTURE_2D,earthTexture)`:** Enables texture mapping and binds the Earth texture.
    - **Texture Coordinates (`glTexCoord2f`):** Specifies texture coordinates for each vertex.
    - **Drawing Triangles (`glBegin(GL_TRIANGLES)`):** Defines the triangles for rendering the Earth.
    - **`glDisable(GL_TEXTURE_2D)`:** Disables texture mapping after drawing.

7. **Update Elapsed Time (`update` function):**
    - **`elapsedTime += rotationSpeed / 360.0`**: Updates the elapsed time based on rotation speed.

8. **Window Resizing:** The `reshape` function ensures the appropriate adjustment of the viewport and projection matrix when the window is resized. This functionality maintains the aspect ratio and provides a seamless user experience across different window dimensions.
    - **`glViewport`, `glMatrixMode`, `glLoadIdentity`, `gluPerspective`:** Configures the viewport and projection matrix for proper resizing.

9. **Drawing Celestial Bodies (`drawPlanet` function):** The `glutSolidSphere` function is extensively used to create spherical representations of celestial bodies, including the sun and planets. These spheres serve as the foundation for rendering the three-dimensional aspects of the solar system.
    - **`glPushMatrix`, `glRotatef`, `glTranslatef`:** Manipulates the modelview matrix for rotation and translation.
    - **`glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, color)`:** Sets the material color for the celestial body.
    - **`glutSolidSphere`**: Draws a solid sphere representing the celestial body.

10. **Drawing Saturn Rings (`drawSaturnRings` function):**
    - **`glBegin(GL_QUADS)`:** Begins drawing quadrilaterals for the rings.
    - **Loop (`for`):** Iterates through angles to define vertices of the rings.
    - **`glVertex3f`:** Specifies the vertices of each quad.
    - **`glEnd()`:** Ends the drawing of quadrilaterals.

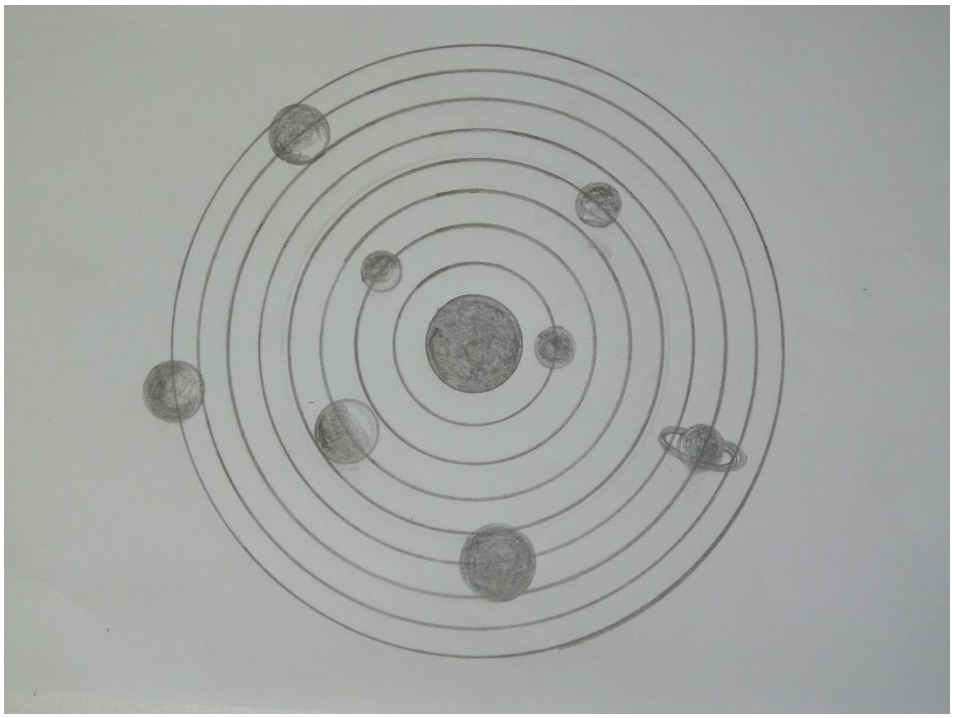
11. **User Interaction:** The `keyboard` function responds to user input, allowing for the interactive switching of views between a global perspective of the solar system and a closer view centered on Earth. This feature enhances user engagement and exploration.
    - **`GLUT_KEY_RIGHT`, `GLUT_KEY_LEFT`:** Identifies the right and left arrow keys.
    - **`switchToEarhView` flag:** Controls the switch between global and Earth-centered views.

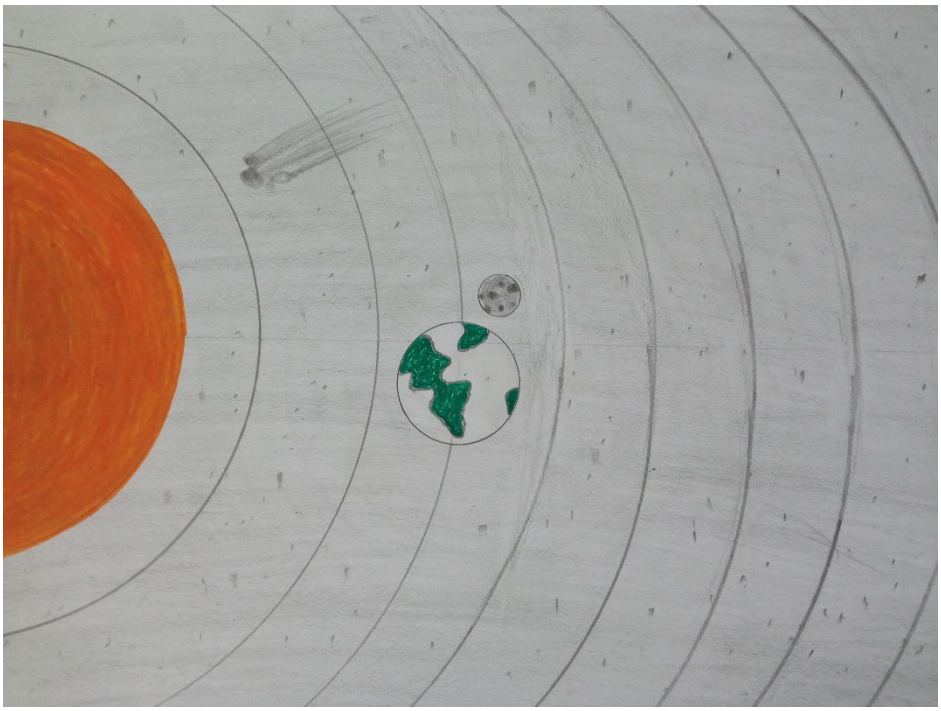
12. **Meteor Shower Animation (`updateMeteorShower`, `drawMeteorShower` functions):**
    - **`glDisable(GL_LIGHTING)`, `glLineWidth`, `glColor3f`:** Configures rendering for meteor shower streaks.
    - **Loop (`for`):** Iterates through existing streaks to update and draw them.
    - **`glEnable(GL_LIGHTING)`:** Re-enables lighting after drawing meteor shower.

- ## Motivation:

  Creating this computer graphics project was fueled by a simple desire—to bring the solar system to life on a digital canvas. Space always sparked our curiosity. With rudimentary graphics knowledge, driven by the desire of exploring OpenGL's capabilities we started this project to craft a visual representation of our solar system, complete with rotating planets and dynamic meteor showers.

- **Diagram:**





- **List of objects:**

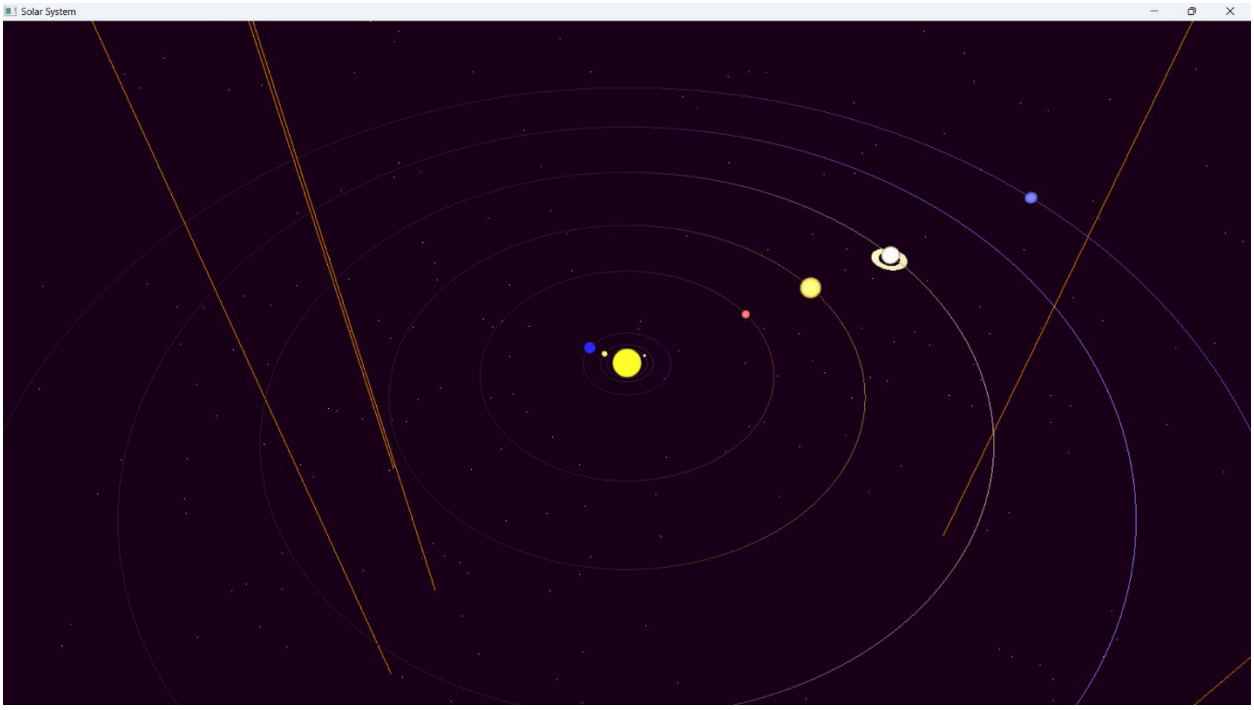
1. **Sun:**
   - Represented by the function `drawSun()`.
   - A solid sphere colored yellow to depict the sun.
2. **Planets:**
   - Mercury, Venus, Earth, Mars, Jupiter, Saturn, Uranus, Neptune.
   - Each planet is represented by the function `drawPlanet()`.
   - Planets have varying radii, orbit radii, revolution speeds, and colors.
   - The Earth has a special function `drawEarthWithTexture()` for rendering with a texture.
3. **Orbit Circles:**
   - Circles representing the orbits of planets.
   - Drawn using the function `drawOrbitCircle()`.
4. **Saturn Rings:**
   - The rings of Saturn are drawn using the function `drawSaturnRings()`.
5. **Meteor Shower Streaks:**
   - Represented by the vector `meteorShowerStreaks`.
   - Lines resembling meteor shower streaks drawn using the functions `updateMeteorShower()` and `drawMeteorShower()`.
6. **Textures:**
   - Earth has a texture loaded using the `LoadTexture()` function.

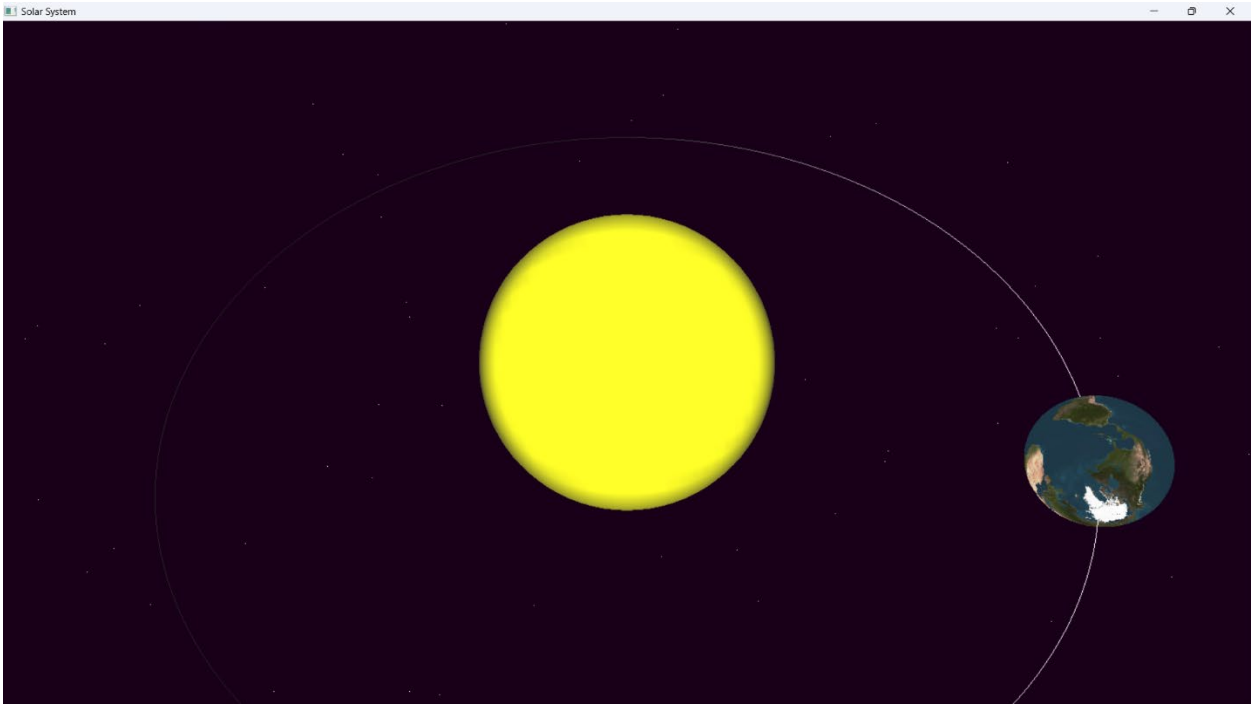
- **Functions to represent the objects:**

- **Sun:**
  - `void drawSun()`: Function to draw a solid sphere representing the sun.
- **Planets:**
  - `void drawPlanet(GLfloat radius, GLfloat orbitRadius, GLfloat revolution, GLfloat color[])`: Generic function to draw a planet.
  - `void drawEarthWithTexture(GLfloat radius, GLfloat orbitRadius, GLfloat revolution)`: Function to draw Earth with a texture.
- **Orbit Circles:**
  - `void drawOrbitCircle(GLfloat radius)`: Function to draw a circle representing the orbit of a planet on the xz plane.
- **Saturn Rings:**
  - `void drawSaturnRings(GLfloat innerRadius, GLfloat outerRadius, GLfloat thickness)`: Function to draw the rings of Saturn.
- **Meteor Shower Streaks:**
  - `void updateMeteorShower(int value)`: Function to update the position of meteor shower streaks.
  - `void drawMeteorShower()`: Function to draw meteor shower streaks.

- **Textures:**
  - `GLuint LoadTexture(const char* filename)`: Function to load a texture.
- **Initialization and Display:**
  - `void init()`: Function to initialize the OpenGL environment, lights, camera, and load textures.
  - `void display()`: Function to display the solar system simulation.
  - `void reshape(int width, int height)`: Function to handle window resizing.
- **Time Update:**
  - `void update(int value)`: Function to update the elapsed time and schedule the next update.
- **Keyboard Input:**
  - `void keyboard(int key, int x, int y)`: Function to handle keyboard input, specifically for switching views.

- **<u>Output ScreenShot:</u>**





- **Demo Link:**
  https://aiubedu60714-my.sharepoint.com/:v:/g/personal/21-45904-3_student_aiub_edu/EXNJW1qAWJ5BoKbtP6aHOU8B4lgYUMvEkG4Sm_7EdCqTZQ?nav=eyJyZWZlcnJhbEluZm8iOnsicmVmZXJyYWxBcHAiOiJPbmVEcml2ZUZvckJ1c2luZXNzIiwicmVmZXJyYWxBcHBQbGF0Zm9ybSI6IldlYiIsInJlZmVycmFsTW9kZSI6InZpZXciLCJyZWZlcnJhbFZpZXciOiJNeUZpbGVzTGlua0NvcHkifX0&e=YFxCjf

- **Uniqueness of Our Project:**
  This computer graphics project stands out for its incorporation of two distinctive elements—3D rendering and texture mapping. As the planets and the sun are spheres, it was not possible to render them realistically in 2D as mentioned in the project proposal.

  Texture mapping further enriches the experience by seamlessly applying detailed and realistic textures to the planets, enhancing their visual appeal. It was not a feasible way to draw the planets by drawing geometric shapes on the surface of a sphere since it would require hundreds of vertices. Hence, the drawing of the continents necessitated the incorporation of texture mapping. An image of the earth's surface was wrapped around the sphere representing the earth.

- **Conclusion:**
  In our crude representation of the solar system, we got to learn more about OpenGL than expected. Although the visuals may seem very simple, it took many hours of work. The time constraints put extra pressure. We had to take different approaches for rotating the planets around the sun until the final version proved to be correct. The texture mapping

presented before us another challenge. Because `freeglut` does not provide native support for texture mapping. So, we had to search online for an external library. At first, we stumbled upon SOIL library. But it was very outdated and moreover the website that used to host that library had gone down. Then we went on to find another library and found one called `stb_image`. With the help of `stb_image` functionalities, we were able draw a realistic surface of the earth.

- **Contribution:**

Name & Id	Work Details
Sakib, Md. Zahid Sadman (21-45904-3)	Texture mapping, keyboard, rotation, reshape
Shakhawat, Mahmud Ibna (20-43909-2)	Saturn rings, meteor shower, distant stars
Mahin, Md. Mosaddek Hossain (20-43905-2)	Sun, planets, orbits

Reference:

These websites and youtubers have been immensely helpful for the project. We are grateful for their quality and comprehensive learning materials:

1. https://learnopengl.com/
2. https://www.youtube.com/@LearningBits1024
3. https://github.com/nothings/stb
4. https://chat.openai.com/