Mini Project

3D – Solar System

Name: Potnuru Yashwanth

Reg: 11914818

Course Name: Computer Graphics and Visualization

Section: KE076

Submitted to

Dr. Om Prakash Yadav (26121)

Asst Professor,

School of Computer Science and Engineering

LOVELY PROFESSIONAL UNIVERSITY PHAGWARA, PUNJAB

March - 2023

Abstract: This project involves the development of a three-dimensional solar system model with a central sun and rotating planets using OpenGL and Dev C++. The aim of the project is to create a visually appealing and interactive simulation that accurately represents the solar system.

The solar system model will include a central sun and eight planets, with the planets rendered as 3D objects with accurate size, shape, and texture mapping. The planets will be animated to simulate their orbits and rotations around the sun. The project will be designed using the OpenGL graphics library, which will enable advanced features such as lighting and shading effects. The code will be implemented in Dev C++, an IDE for C++ programming language.

The simulation will allow for interactivity, with users being able to control the camera angle and animation speed, allowing them to explore the planets in detail. The project will also be accompanied by a user manual, which will explain the controls and features of the simulation. The code will be modular and organized into separate classes for the different components of the solar system model. This will make it easy to modify and add new features to the simulation, such as adding additional celestial bodies.

Overall, this project aims to provide an immersive and educational experience for users who want to learn about the solar system through an interactive 3D simulation.

Key Words: OpenGL, Dev C++, GL Functions, 3D Graphics.

Introduction: The three-dimensional (3D) solar system simulation is an exciting project that aims to provide an interactive and educational tool for learning about our solar system. This project has been developed using OpenGL functions and implemented in Dev C++, an IDE for C++ programming language.

The solar system simulation includes a central sun and eight planets, each of which is represented as a 3D object with accurate size, shape, and texture mapping. The planets are animated to simulate their orbits and rotations around the sun, allowing users to explore and learn about the solar system in an interactive and engaging way.

The project incorporates advanced graphics features, such as lighting and shading effects, to create a visually stunning simulation. It also includes user controls, such as the ability to adjust the animation speed and camera angle, providing users with a customizable and immersive experience. The development of this project involved careful planning and design, with the code structured into separate classes for the different components of the solar system model. This makes the code easy to modify and expand upon, enabling the addition of new features or celestial bodies.

Overall, this project demonstrates the power of OpenGL and Dev C++ in creating visually stunning and interactive simulations of complex systems. This project serves as a valuable tool for education and provides an engaging and immersive experience for users to learn about the solar system.

Methodology: OpenGL using Dev C++

```
#include <GL/gl.h>
#include <GL/glut.h>
#include <iostream>
#include <string.h>
using namespace std;

float xRotated = 90.0, yRotated = 0.0, zRotated = 0.0;

//________________

//defining some MACROS

#define FPS 1000 // frame per seconds

// define another constant to hold ASCII for Escape key.
#define KEY_ESC 27
//________________
void reshapeFunc(int x, int y)
{
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluPerspective(40.0, (GLdouble)x / (GLdouble)y, 0.5,20.0);
	glMatrixMode(GL_MODELVIEW);
	glViewport(0,0,x,y);
}
//________________
void Draw_Spheres()
{
	glClearColor(0, 0.0,0.0, 0 ); // Red==Green==Blue==1 --> White Colour
	glClear(GL_COLOR_BUFFER_BIT); //Update the colors
	glMatrixMode (GL_MODELVIEW);
	glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
	glLoadIdentity();
	glTranslatef(0.0,0.0,-15.0);
```

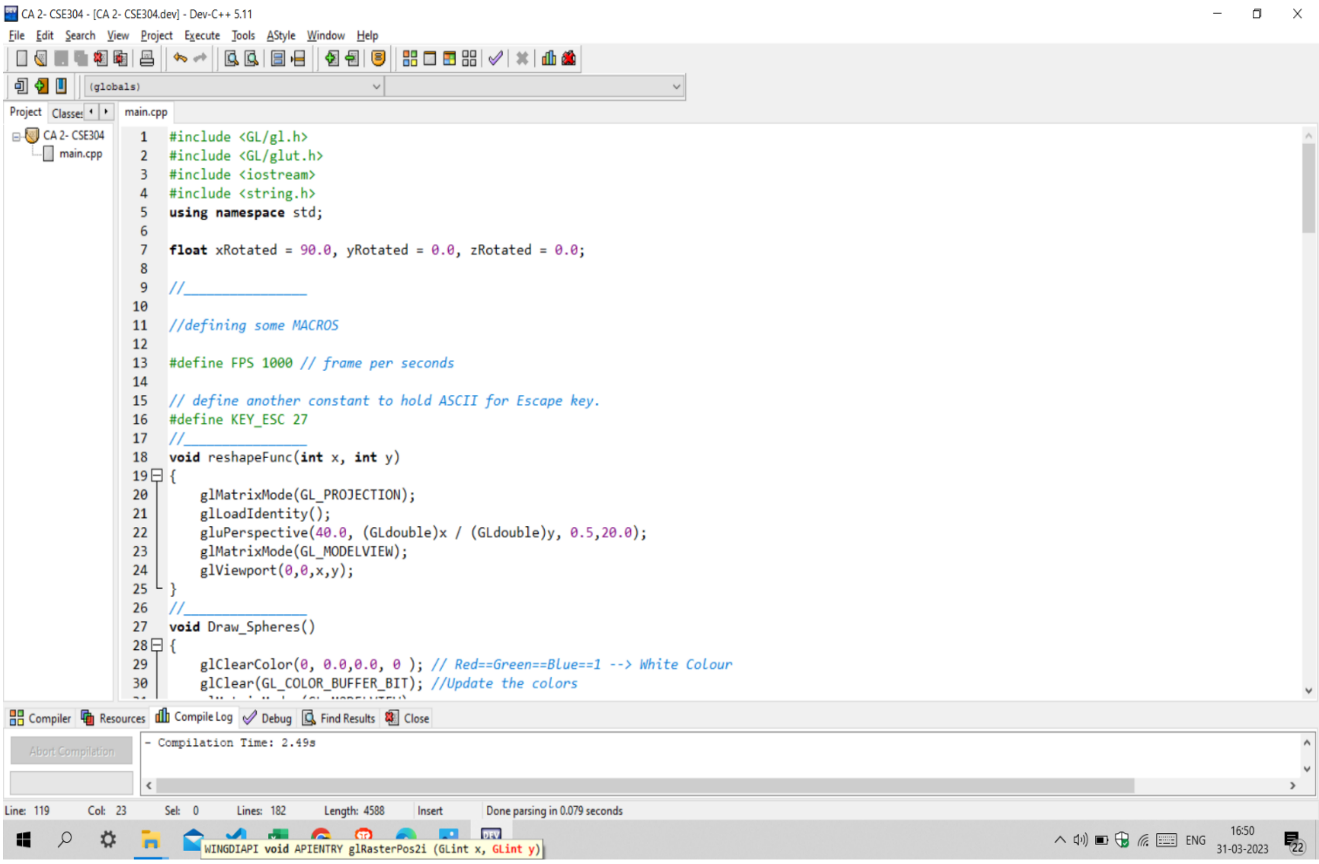
```
// SUN
	glColor4f(1.0,1.0,0.0,0.0);		//Yellow color
	glPushMatrix();
	glTranslatef(-12.5,0.0,0.0);
	glutSolidSphere(4.0,50,50);
	glPopMatrix();

// Mercury
	glColor3f(0.0,0.1,0.1);	//Dark Grey color
	glPushMatrix();
	glTranslatef(-7,0.0,0.0);
	glRotatef(60.0,1,0,0);
	glRotatef(zRotated*1.5,0,0,1);
	glutSolidSphere(0.4,15,10);
	glPopMatrix();

// Venus
	glColor4f(1.0,0.5,0.0,0.0);		//Orange Brown color
	glPushMatrix();
	glTranslatef(-5.5,0.0,0.0);
	glRotatef(60.0,1,0,0);
	glRotatef(zRotated,0,0,1);
	glutSolidSphere(0.7,20,20);
	glPopMatrix();

//Earth
	glColor3f(0.1,0.2,0.8);	//Blue color
	glPushMatrix();
	glTranslatef(-3.8,0.0,0.0);
	glRotatef(60.0,1,0,0);
	glRotatef(zRotated*3,0,0,1);
	glutSolidSphere(0.7,20,20);
	glPopMatrix();

//Mars
	glColor3f(0.8,0.2,0.1);	//Red color
	glPushMatrix();
	glTranslatef(-2.3,0.0,0.0);
	glRotatef(125.0,1,0,0);
	glRotatef(zRotated*4.0,0,0,1);
	glutSolidSphere(0.5,20,15);
	glPopMatrix();

//Jupiter
```

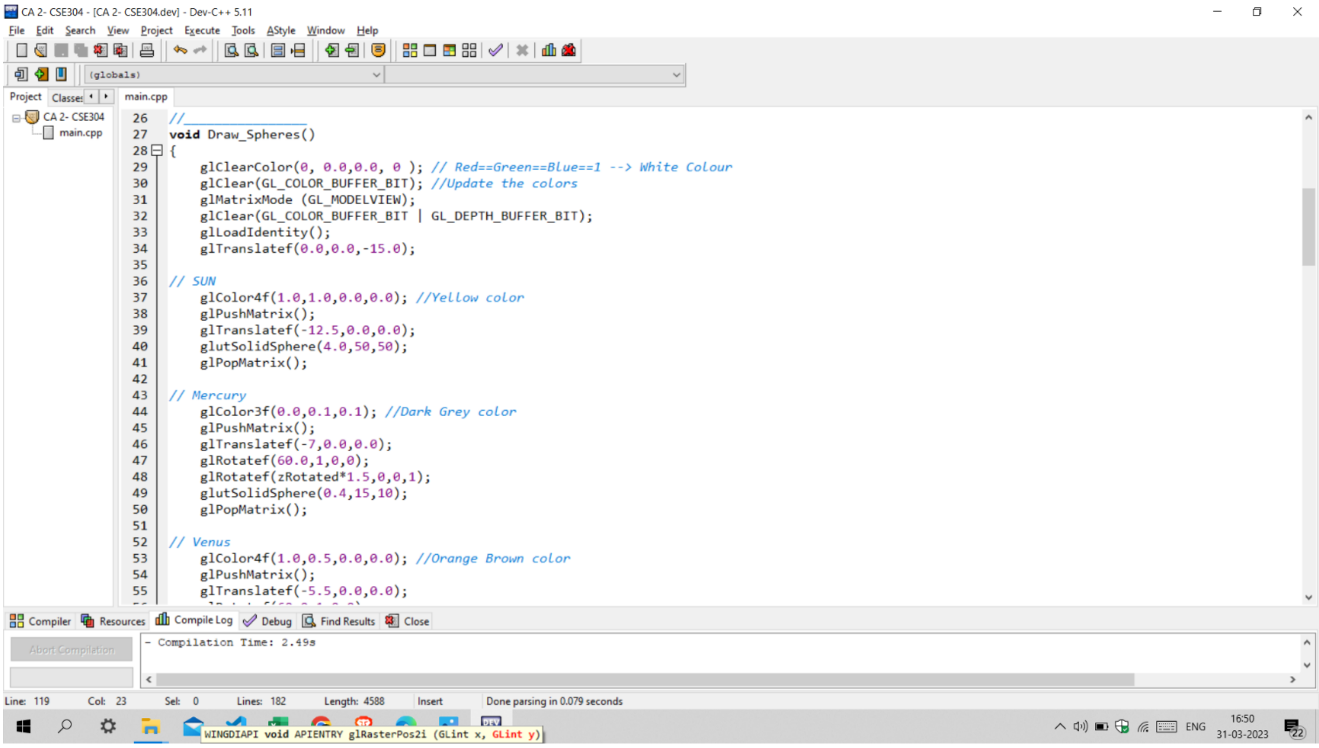
```
        glColor3f(1.0,0.5,0.0);     //Orange color
        glPushMatrix();
        glTranslatef(-0.5,0.0,0.0);
        glRotatef(125.0,1,0,0);
        glRotatef(zRotated*7.0,0,0,1);
        glutSolidSphere(1.0,20,40);
        glPopMatrix();


// Saturn
        glColor3f(0.0,0.5,0.5);     //Bluish Green
        glPushMatrix();
        glTranslatef(2.0,0.0,0.0);
        glRotatef(60.0,1,0,0);
        glRotatef(zRotated*6.5,0,0,1);
        glutSolidSphere(1.0,20,30);
        glPopMatrix();

//Uranus
        glColor3f(0.5,1.0,1.0);     //Cyan Color,
        glPushMatrix();
        glTranslatef(4.0,0.0,0.0);
        glRotatef(125.0,1,0,0);
        glRotatef(zRotated*5.0,0,0,1);
        glutSolidSphere(0.7,20,20);
        glPopMatrix();


//Neptune
        glColor3f(0.1,0.0,1.0);     //Purple Color,
        glPushMatrix();
        glTranslatef(6.0,0.0,0.0);
        glRotatef(60.0,1,0,0);
        glRotatef(zRotated*5.5,0,0,1);
        glutSolidSphere(0.7,20,20);
        glPopMatrix();
        // Set the color to white
        glColor3f(1.0f,1.0f,1.0f);
        // Set the position for the text
        glRasterPos2i(4,-3);
        // Display the text
        char number[] = "11914818";
        char name[] = "Potnuru Yashwanth - ";
        for(int i = 0; i < 20; i++){
```

```
		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,name[i]);
	}
	for(int i = 0;i < 8; i++)
	{
		glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,number[i]);
	}
	glutSwapBuffers();

	glFlush();

}
//________________

void idleFunc()
{
	zRotated+=0.1;
	glutPostRedisplay();
}

//________________

void SetCanvasSize(int width, int height) {
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	glOrtho(0, width, 0, height, -1, 1); // set the screen size to given width and height.
}
//________________

void PrintableKeys(unsigned char key, int x, int y) {
	if (key == KEY_ESC/* Escape key ASCII*/) {
		exit(1); // exit the program when escape key is pressed.
	}
	glFlush();
}

//________________


int main(int argc, char**argv) {
	int width = 1300, height = 800; // i have set my window size to be 800 x 600
	glutInit(&argc, argv); // initialize the graphics library...
	//glutDisplayFunc(display);
	glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA); // we will be using color
display mode
```

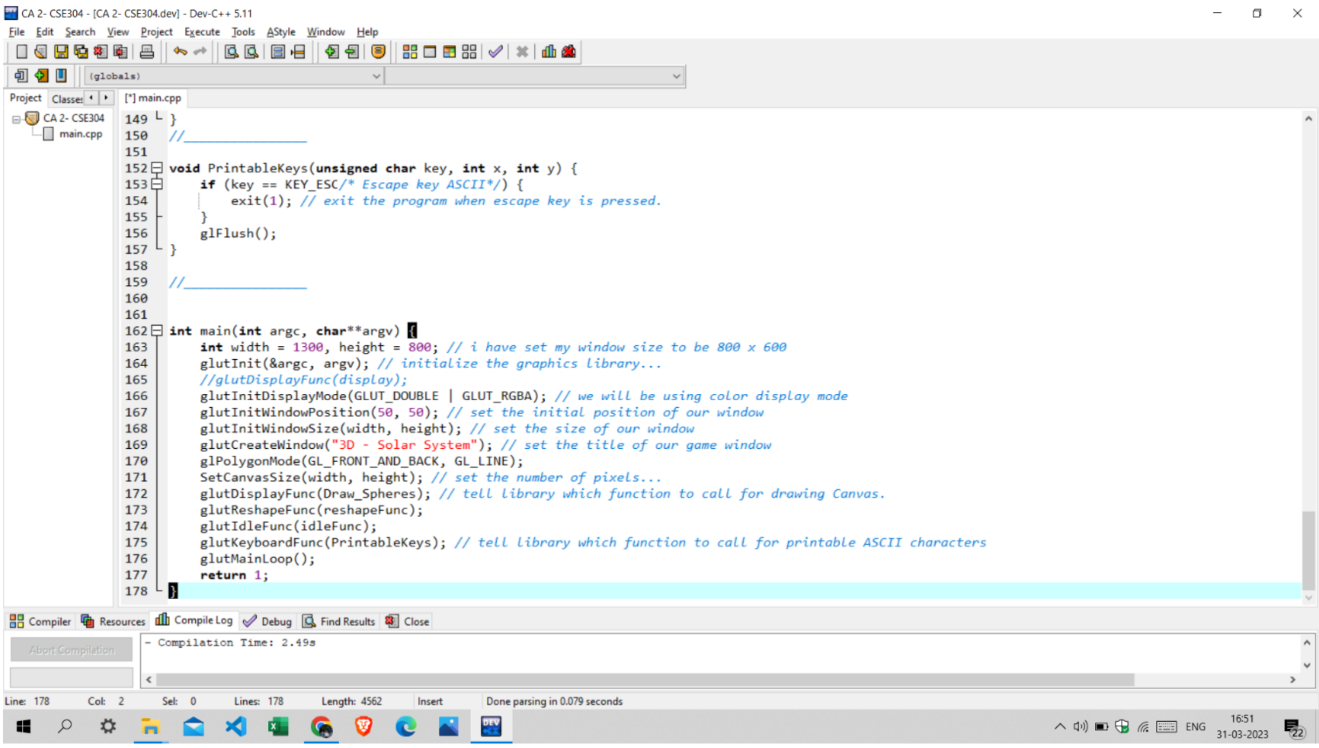
```
	glutInitWindowPosition(50, 50); // set the initial position of our window
	glutInitWindowSize(width, height); // set the size of our window
	glutCreateWindow("3D - Solar System"); // set the title of our game window
	glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
	SetCanvasSize(width, height); // set the number of pixels...
	glutDisplayFunc(Draw_Spheres); // tell library which function to call for drawing Canvas.
	glutReshapeFunc(reshapeFunc);
	glutIdleFunc(idleFunc);
	glutKeyboardFunc(PrintableKeys); // tell library which function to call for printable ASCII characters
	glutMainLoop();
	return 1;
}
```

Snapshot of Project:



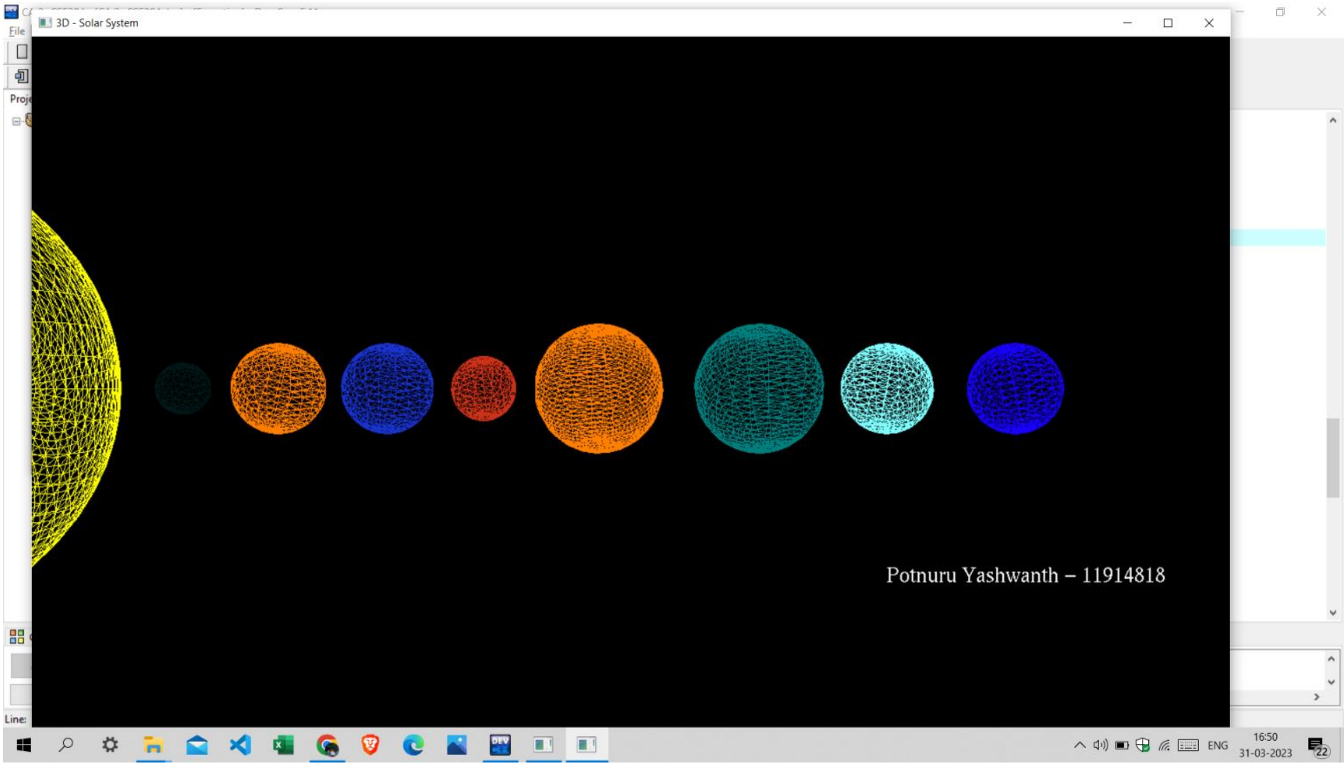


```
//________________
void Draw_Spheres()
{
    glClearColor(0, 0.0,0.0, 0 ); // Red==Green==Blue==1 --> White Colour
    glClear(GL_COLOR_BUFFER_BIT); //Update the colors
    glMatrixMode (GL_MODELVIEW);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
    glLoadIdentity();
    glTranslatef(0.0,0.0,-15.0);

// SUN
    glColor4f(1.0,1.0,0.0,0.0); //Yellow color
    glPushMatrix();
    glTranslatef(-12.5,0.0,0.0);
    glutSolidSphere(4.0,50,50);
    glPopMatrix();

// Mercury
    glColor3f(0.0,0.1,0.1); //Dark Grey color
    glPushMatrix();
    glTranslatef(-7,0.0,0.0);
    glRotatef(60.0,1,0,0);
    glRotatef(zRotated*1.5,0,0,1);
    glutSolidSphere(0.4,15,10);
    glPopMatrix();

// Venus
    glColor4f(1.0,0.5,0.0,0.0); //Orange Brown color
    glPushMatrix();
    glTranslatef(-5.5,0.0,0.0);
```



```
}
//________________

void PrintableKeys(unsigned char key, int x, int y) {
    if (key == KEY_ESC/* Escape key ASCII*/) {
        exit(1); // exit the program when escape key is pressed.
    }
    glFlush();
}

//________________


int main(int argc, char**argv) {
    int width = 1300, height = 800; // i have set my window size to be 800 x 600
    glutInit(&argc, argv); // initialize the graphics library...
    //glutDisplayFunc(display);
    glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA); // we will be using color display mode
    glutInitWindowPosition(50, 50); // set the initial position of our window
    glutInitWindowSize(width, height); // set the size of our window
    glutCreateWindow("3D - Solar System"); // set the title of our game window
    glPolygonMode(GL_FRONT_AND_BACK, GL_LINE);
    SetCanvasSize(width, height); // set the number of pixels...
    glutDisplayFunc(Draw_Spheres); // tell library which function to call for drawing Canvas.
    glutReshapeFunc(reshapeFunc);
    glutIdleFunc(idleFunc);
    glutKeyboardFunc(PrintableKeys); // tell library which function to call for printable ASCII characters
    glutMainLoop();
    return 1;
}
```

Output:



Future Enhancement: There are several potential future enhancements for the 3D solar system simulation using OpenGL and Dev C++. Some possible enhancements include:

Improved accuracy**:** The current simulation uses simplified models of celestial bodies and assumes circular orbits for all planets. Future enhancements could include the use of more accurate models and simulations, taking into account factors such as elliptical orbits, gravitational interactions between objects, and the effects of other celestial bodies in the solar system.

User interactivity: The current simulation allows users to adjust the camera angle and animation speed, but future enhancements could include the ability for users to interact with the simulation in more sophisticated ways. For example, users could be able to select individual planets to view more detailed information about them, or to manipulate the simulation in real-time.

Expanded content: While the current simulation includes the sun and eight planets, there are many other interesting objects in the solar system that could be added, such as asteroids, comets, and moons. Future enhancements could include the addition of these objects, as well as more detailed information and visualizations about each one.

Improved graphics: While the current simulation includes advanced graphics features such as lighting and shading effects, future enhancements could include even more advanced graphics techniques, such as more detailed textures, improved rendering algorithms, and more realistic atmospheric effects.

Virtual reality: As virtual reality technology becomes more advanced and accessible, it may be possible to create a fully immersive VR version of the solar system simulation. This could provide an even more engaging and interactive experience for users, allowing

them to explore the solar system in a truly immersive way.
Overall, there are many exciting possibilities for future enhancements to the 3D solar system simulation using OpenGL and Dev C++. These enhancements could make the simulation even more engaging, informative, and enjoyable for users, and could help to inspire a deeper appreciation and understanding of our solar system.

Conclusion: In conclusion, the 3D solar system simulation project using OpenGL and Dev C++ provides an immersive and informative tool for users to explore the planets and their orbits. It showcases advanced graphics techniques and simulations, and has the potential for future enhancements such as improved accuracy, user interactivity, expanded content, improved graphics, and virtual reality. Overall, the project demonstrates the power of computer graphics and simulations to create engaging educational tools.

References:

- OpenGL Programming Guide: The Official Guide to Learning OpenGL, Versions 3.0 and 3.1, by Dave Shreiner, Graham Sellers, and John Kessenich.
- "Solar System Simulation using OpenGL," a video tutorial by the YouTube channel "The Coding Train," available at: https://www.youtube.com/watch?v=ZI1dmHv3MeM
- Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL, Seventh Edition, by Edward Angel and Dave Shreiner.
- Computer Graphics: Principles and Practice, Third Edition, by John F. Hughes, Andries van Dam, Morgan McGuire, David F. Sklar, James D. Foley, Steven K. Feiner, and Kurt Akeley.