VISVESVARAYA TECHNOLOGICAL UNIVERSITY

BELGAVI, KARNATAKA -590 018

A Minor Project Report on

"FERRIS WHEEL"

Submitted in partial fulfillment for the Computer Graphics Laboratory with Mini-Project[18CSL67] Course of Sixth Semester of Bachelor of Engineering in Computer Science & Engineering during the academic year 2022-23.

By

DRUPAD S	4MN20CS015
SYED ZEESHAN	4MN20CS050

Under the Guidance of

Mr. Bharath Bharadwaj B S

Assistant Professor

Dept. of CS&E

MIT Thandavapura

2022-23

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

NH 766, NANJANGUD TALUK, MYSURU – 571302

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

MAHARAJA INSTITUTE OF TECHNOLOGY THANDAVAPURA

CERTIFICATE

*Certified that the minor project work entitled "**Ferris Wheel**" is a bonafide work carried out by **Drupad S** (4MN20CS015) & **Syed Zeeshan**(4MN20CS050) for the course **Computer Graphics Laboratory with Mini-Project** with course code **18CSL67** of Sixth Semester in Computer Science & Engineering under Visvesvaraya Technological University, Belagavi during academic year **2022-23**.*

It is certified that all corrections/suggestions indicated for Internal Assignment have been incorporated in the report. The report has been approved as it satisfies the course requirements.

_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _	_ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _
Signature of Lab Staff In-Charge	Signature of the HoD
Mr. Bharath Bharadwaj B S	**Dr. Ranjit K N**
Assistant Professor	Associate Professor & Head
Dept. of CS&E	Dept. of CS&E
MIT Thandavapura	MIT Thandavapura

<u>External viva</u>

Name of the Examiners	**Signature with date**
1)……………………………………………………………………………………	
2)……………………………………………………………………………………	

ACKNOWLEDGEMENT

It is the time to acknowledge all those who have extended their guidance, inspiration and their whole hearted co-operation all along our project work.

We are grateful to **Dr.Y T Krishne Gowda**, Principal, MIT Thandavapura, **Dr.H K Chethan,** Professor and Mentor, CS&E, MIT Thandavapura and **Dr.Ranjit K N**, Associate Professor and Head, CS&E, MIT Thandavapura for having provided us academic environment which nurtured our practical skills contributing to the success of our project.

We would like to sincerely thank our project guide **Mr. Bharath Bharadwaj B S**, Assistant Professor, CS&E, MIT Thandavapura for providing relevant information, valuable guidance and encouragement to complete this project.

We wish to place a deep sense of gratitude to all Teaching and Non-Teaching staffs of Computer Science and Engineering Department for whole-hearted guidance and constant support without which this endeavor would not have been possible.

Our gratitude will not be complete without thanking our parents and also our friends, who have been a constant source of support and aspirations.

DRUPAD S
[4MN20CS015]

SYED ZEESHAN
[4MN20CS050]

ABSTRACT

In this project, A 3D graphics-based Ferris Wheel is great for those who start learning computer graphics and visualization. The purpose of this project is to simulate the working of a Ferris Wheel. I have used OpenGL utility toolkit to implement it, which is written in C++ language. A Ferris wheel is an amusement park ride consisting of a large vertical wheel with places for people to sit or stand spaced evenly around the outer circumference. In operation, the Ferris wheel revolves about a horizontal axis, and the riders are alternatively lifted and then lowered as they are carried around the wheel in a circle. The Ferris Wheel has control keys which used to control the speed of rotation, clockwise & anticlockwise movement and also change the number of riders. This project is designed in such a way that one can view it from any directions using the keyboard function

CONTENTS

LIST OF FIGURES

CHAPTER – 1

INTRODUCTION

Computer Graphics is concerned with all aspects of producing pictures or images using a computer. Graphics provides one of the most natural means of communicating within a computer, since our highly developed 2D and 3D pattern-recognition abilities allow us to perceive and process pictorial data rapidly and effectively. Interactive computer graphics is the most important means of producing pictures since the invention of photography and television.

Applications of Computer Graphics

1. Display of information
2. Design
3. Simulation and animation
4. User interfaces

About OpenGL The Computer Graphics is one of the most effective and commonly used methods to communicate the processed information to the user. It displays the information in the form of graphics objects such as pictures, charts, graphs and diagram instead of simple text.

OpenGL is supported on all platforms. Drivers are available virtually for all graphics cards. OpenGL's extensive capabilities and well-defined architecture supports advanced concepts such as texture mapping, compositing and programmable shaders. It is easy to learn, and it possesses most of the characteristics of other popular graphics systems. The main features of OpenGL are

• It provides 3D geometric objects such as lines, polygons, triangle, meshes, spheres, cubes, quadric surface, and curved surfaces. It provides 3D modeling transformations and viewing functions to create views of 3D scenes using the idea of a virtual camera.

• It supports high-quality rendering of scenes, including hidden-surface removal, multiple light sources, material types, transparency, textures, blending, fog

1.1 Aim

- The aim of the project is to visually demonstrate and explain how the Ferris wheel work using graphics and animations.
- The main aim of this project is to implement the concepts of computer graphics using OpenGL.

1.2 Overview

- This project contains Ferris Wheel. When the user presses ‘c/C’ key, wheel will start rotating and vice-versa.
- The user can control the speed of rotation by pressing ‘v/V’ or ‘x/X’ keys from keyboard where ‘v/V’ increases the speed and ‘x/X’ decreases the speed.
- When the user press ‘w/W’,’s/S’, ’a/A’ & ’d/D’ key, it will rotate the whole wheel into Top, Bottom, Right and Left direction.
- The user can also increase and decrease the number of cabins by pressing key ‘+/1’ and ‘-/0’ respectively.

1.3 Outcome

- The outcome of this project will be a fully functional 3D Ferris Wheel computer graphics application.
- The project demonstrates keyboard interface for specific action that needs to be carried out
- The user will be able to interact with the Ferris Wheel, observing the rotating cabins of the wheel and experiencing a sense of motion.
- Overall, the project aims to create an immersive and realistic 3D Ferris Wheel experience, showcasing the capabilities of computer graphics in simulating motion and creating visually appealing scenes..

CHAPTER – 2

DESIGN AND IMPLEMENTATION

2.1 Algorithm

Step 1. Initialize the OpenGL environment and create the window:

- Include the necessary OpenGL libraries.
- Initialize GLUT and create the window.

Step 2. Initialize the necessary variables and constants for the Ferris Wheel:

- Define variables for rotation angles, animation flags, and light properties.
- Set initial values for these variables.

Step 3. Implement the menu function to handle menu selections for sun and moon light modes:

- Create a menu and add options for selecting sun and moon light modes.
- Implement the menu callback function to handle the selected mode.

Step 4. Create the initialization function (init):

- Set up the lighting properties for the scene, including ambient, diffuse, and specular components.
- Enable lighting and depth testing.
- Create display lists for the wheel and cabin objects.

Step 5. Implement the draw_scene function:

- Set the lighting properties based on the selected light mode.
- Clear the color and depth buffers.
- Set up the perspective projection.
- Apply transformations to position the camera.
- Draw the ground using triangles.
- Draw the trees using cubes for trunks and cubes/scaled cubes for branches.
- Apply translations and rotations to position and animate the Ferris Wheel.
- Call the display lists to draw the wheel and cabin objects.

Step 6. Implement the toRad function to convert degrees to radians:

- Define the toRad function.
- Convert the given degree value to radians and return the result.

Step 7. Set up the main function:

- Initialize the OpenGL environment and create the window.
- Set the window size and position.

- Set the display mode and window title.
- Register the callback functions for display, reshape, and keyboard events.
- Create the menu and attach it to the right mouse button.
- Initialize the scene and enter the main event loop (glutMainLoop).

Step 8. Within the display callback function:

- Call the draw_scene function to draw the scene.

Step 9. Implement the necessary callback functions:

- Implement the reshape callback function to handle window resizing.
- Implement the keyboard callback function to handle key presses

2.2 Flow Chart

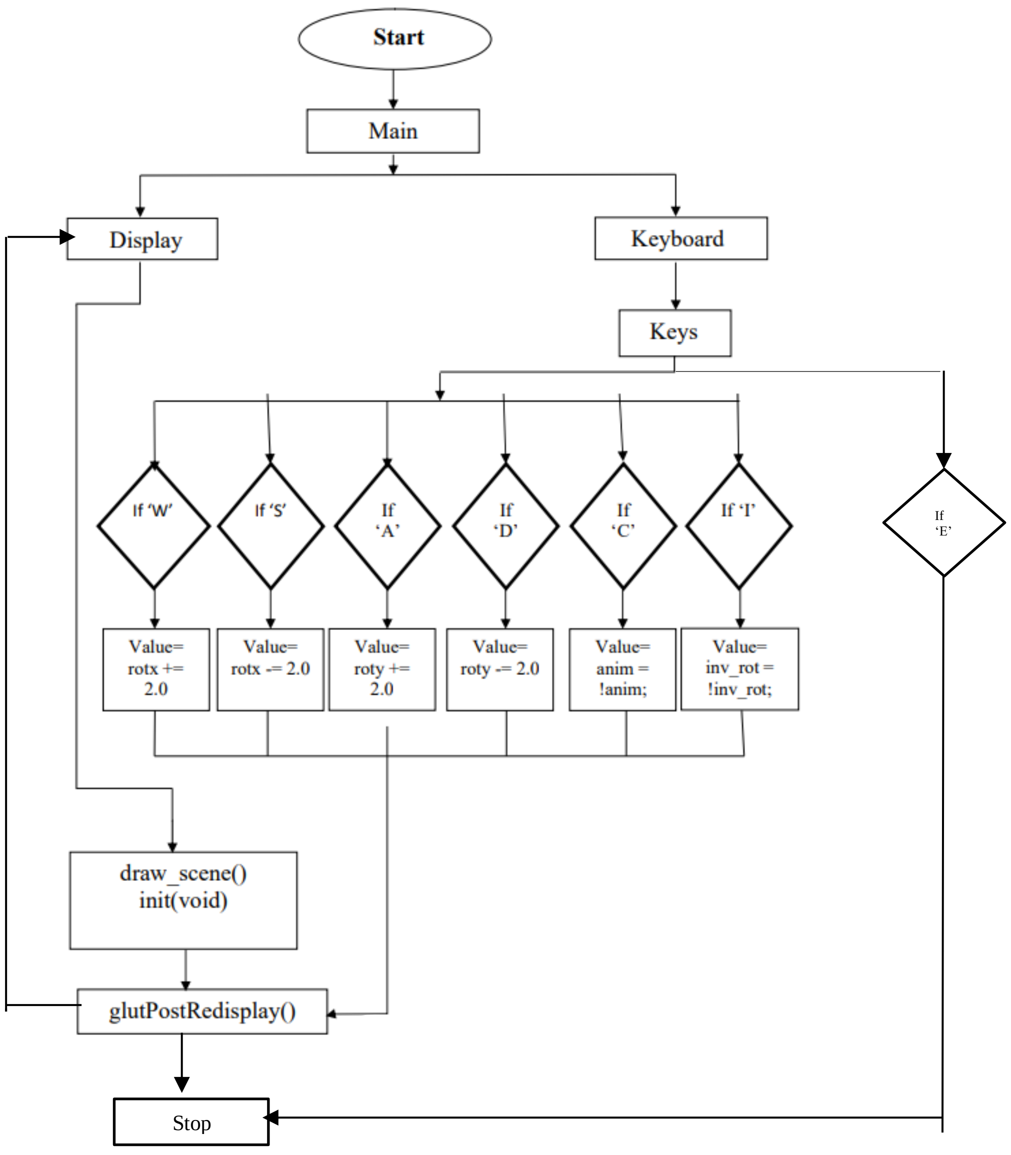


Fig.2.1 : System architecture of Ferris Wheel

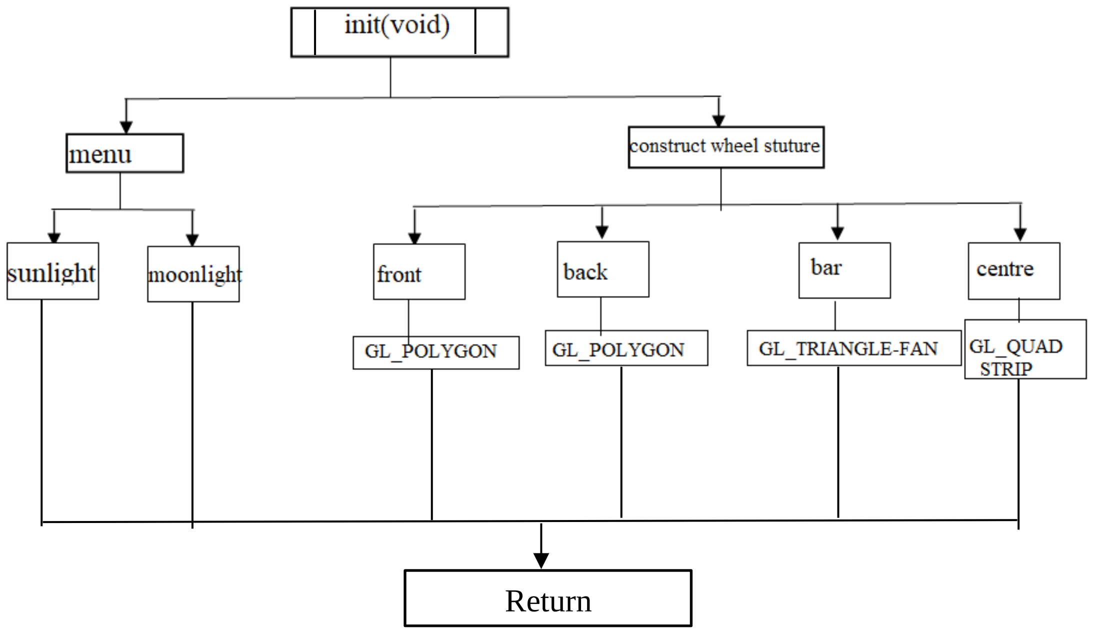


Fig.2.2 : Flow chart for init function

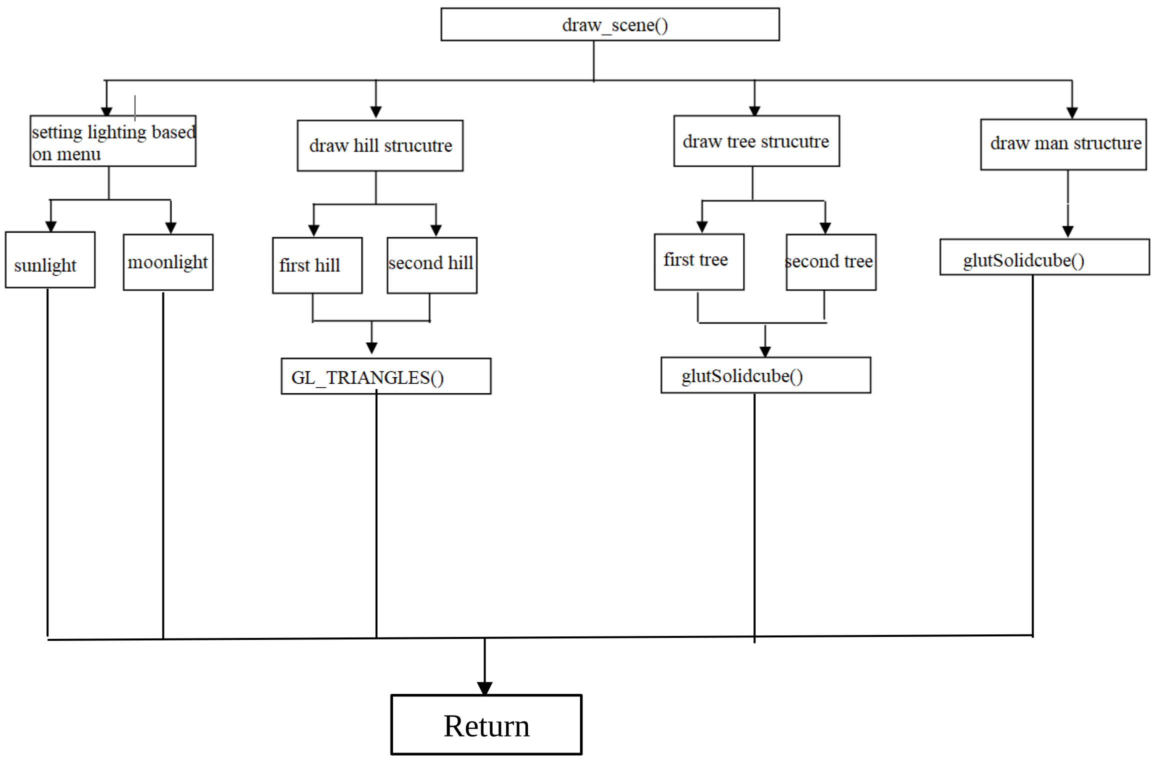


Fig.2.3 : Flow chart for draw_scene function

2.3 OpenGL API's Used with Description

OpenGL has become a widely accepted standard for developing graphics applications. Most of our applications will be designed to access OpenGL directly through functions in three libraries. Functions in main GL library have names that begin with the letters gl and are stored in a library usually referred to as GL.

Function Name	Description
glutInitDisplayMode	Sets the initial display mode . Declaration: void glutInitDisplayMode (unsigned int mode); Remarks: The initial display mode is used when creating top-level windows, sub windows, and overlays to determine the OpenGL display mode for the to-becreated window or overlay.
glutInitWindowposition	set the initial window position. Declaration: void glutInitWindowPosition(int x, int y); x: X location in pixels. y: Window Y location in pixels.
glutInitWindowSize	set the initial window size Declaration: void glutInitWindowSize(int width,int height) width: Width in pixels height: Height in pixels.

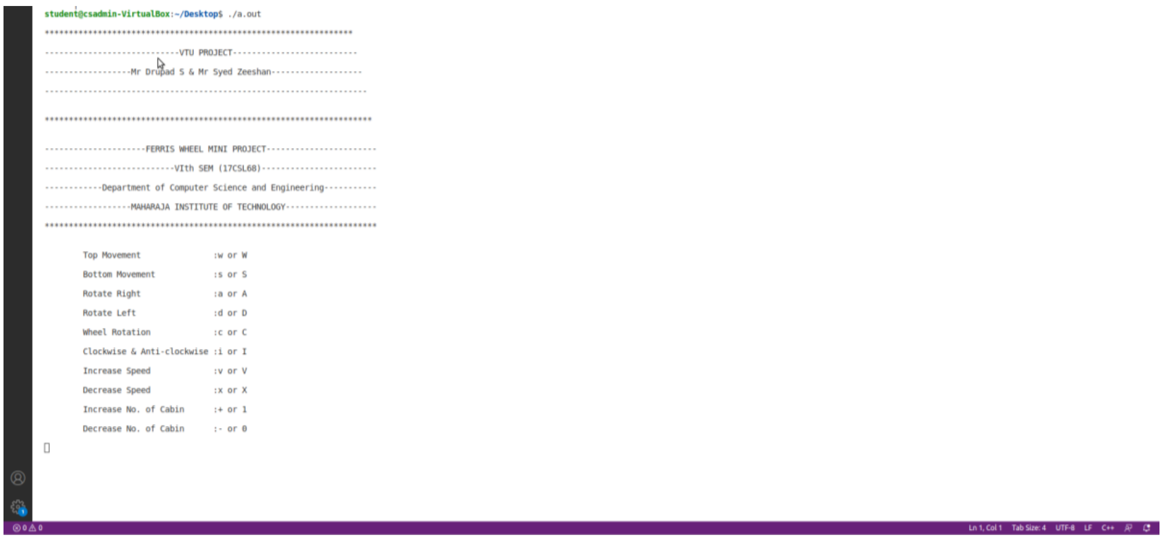
glutCreateWindow	set the title to graphics window. Declaration: Int glutCreateWindow(char *title); Remarks: This function creates a window on the display. The string title can be used to label the window. The integer value returned can be used to set the current window when multiple windows are created.
glutDisplayFunc	Declaration: void glutDisplayFunc(void(*func)void)); Remarks: This function registers the display function that is executed when the window needs to be redrawn.
glClear	Declaration: void glClear(); Remarks: This function clears the particular buffe
glClearColor	Declaration: void glClearColor(GLfloat red, GLfloat green, Glfloatblue, Glfloat alpha); Remarks: This function sets the color value that is used whenclearing the color buffer
glEnd	Declaration: Void glEnd() ; Remarks: This function is used in conjunction with glBegin to delimit the vertices of an opengl primitive

glMatrixMode	Declaration: void glMatrixMode(GLenum mode); Remarks: This function specifies which matrix will be affected by subsequent transformations mode can be GL_MODELVIEW,GL_PROJECTION or GL_TEXTURE..
glIint	Declaration: Void glutInit(int *argc, char **argv). Remarks: To start through graphics system, we must first call glutInit (),glutInit will initialize the GLUT library and negotiate a session with the window system. During this process, glutInit may cause the termination of the GLUT program with an error message to the user if GLUT cannot be properly initialized
glutAddMenuEntry	Declaration: Void glutAddMenuEntry(char *name,int value); Remarks: Adds a menu entry to the bottom of the current menu.The String name will be displayed for the newly added menu entry.An identifier is used to represent each option.
glutCreateMenu	Declaration: int glutCreateMenu(void (*func)(int value)); Remarks: It creates a new pop-up menu and returns a unique small integer identifier.The range of the allocated identifiers starts at 1.

Table:2.1 :OpenGL API’s used with Description

CHAPTER – 3

RESULT ANALYSIS



```
student@csadmin-VirtualBox:~/Desktop$ ./a.out
****************************************************************
---------------------------VTU PROJECT-------------------------
-----------------Mr Drupad S & Mr Syed Zeeshan------------------
-------------------------------------------------------------------

********************************************************************

---------------------FERRIS WHEEL MINI PROJECT----------------------
--------------------------VIth SEM (17CSL68)-----------------------
-----------Department of Computer Science and Engineering----------
-----------------MAHARAJA INSTITUTE OF TECHNOLOGY------------------
*********************************************************************

        Top Movement              :w or W
        Bottom Movement           :s or S
        Rotate Right              :a or A
        Rotate Left               :d or D
        Wheel Rotation            :c or C
        Clockwise & Anti-clockwise :i or I
        Increase Speed            :v or V
        Decrease Speed            :x or X
        Increase No. of Cabin     :+ or 1
        Decrease No. of Cabin     :- or 0
```

Fig.3.1:Console Window

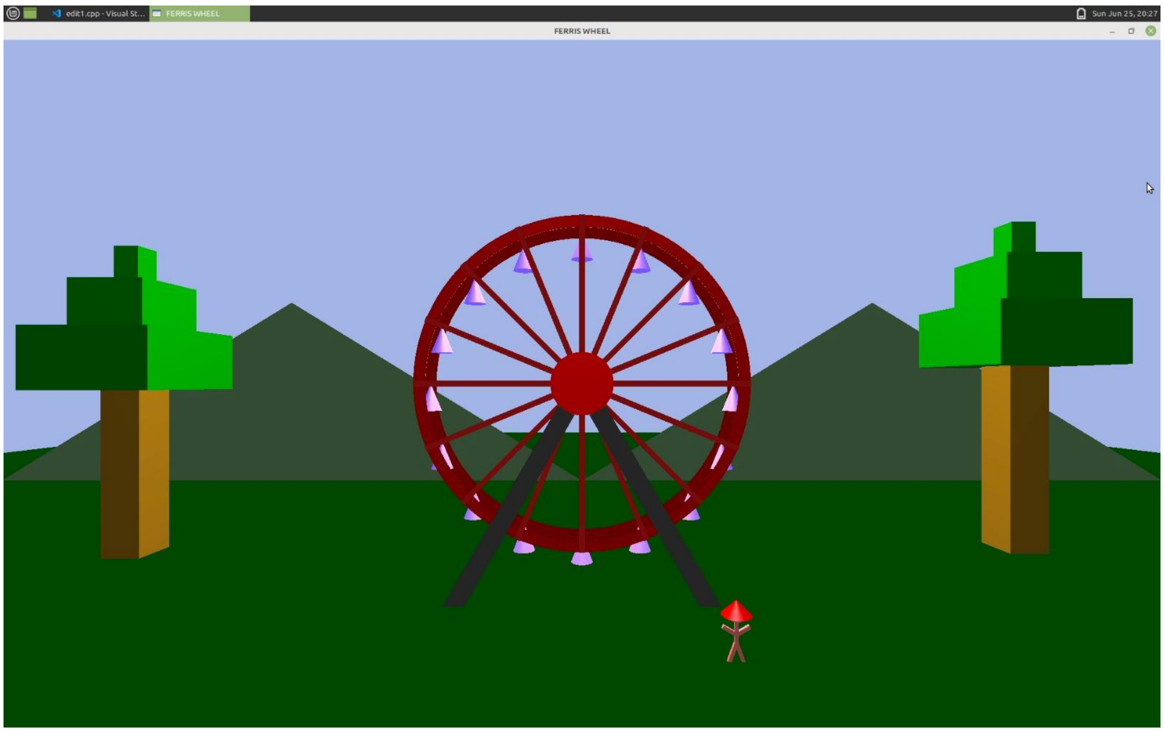


Fig.3.2:Ferrish wheel front view

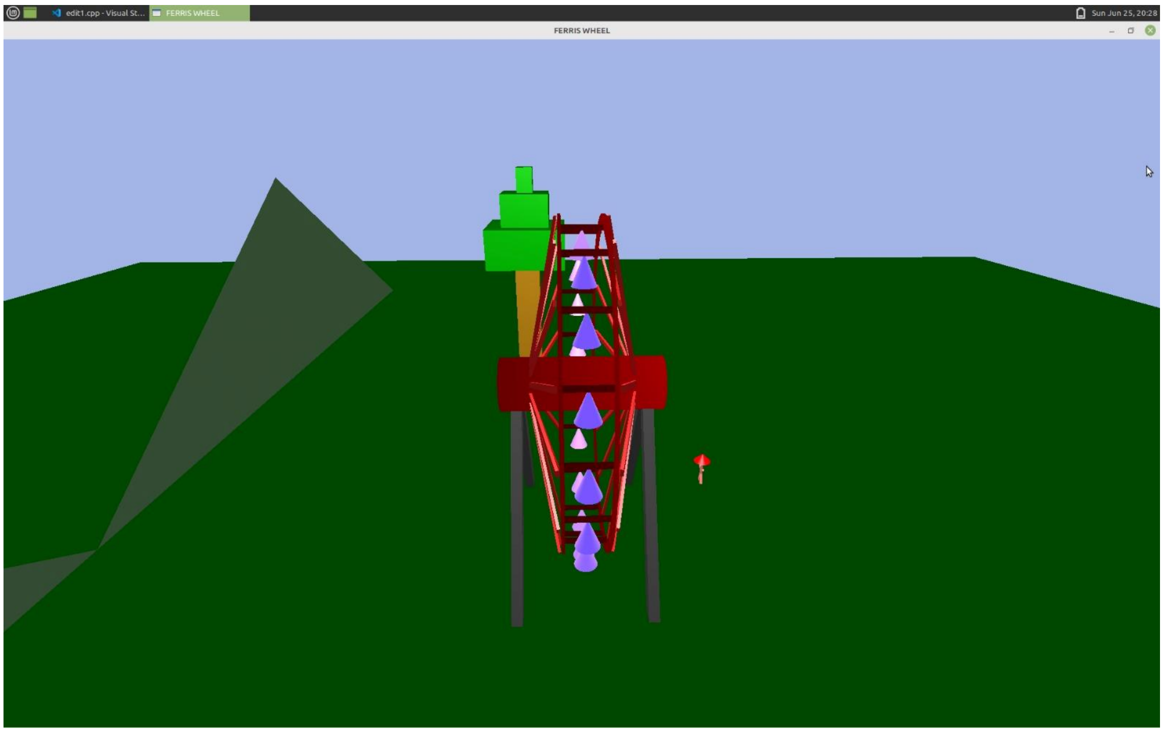


Fig.3.3:Ferrish wheel side view

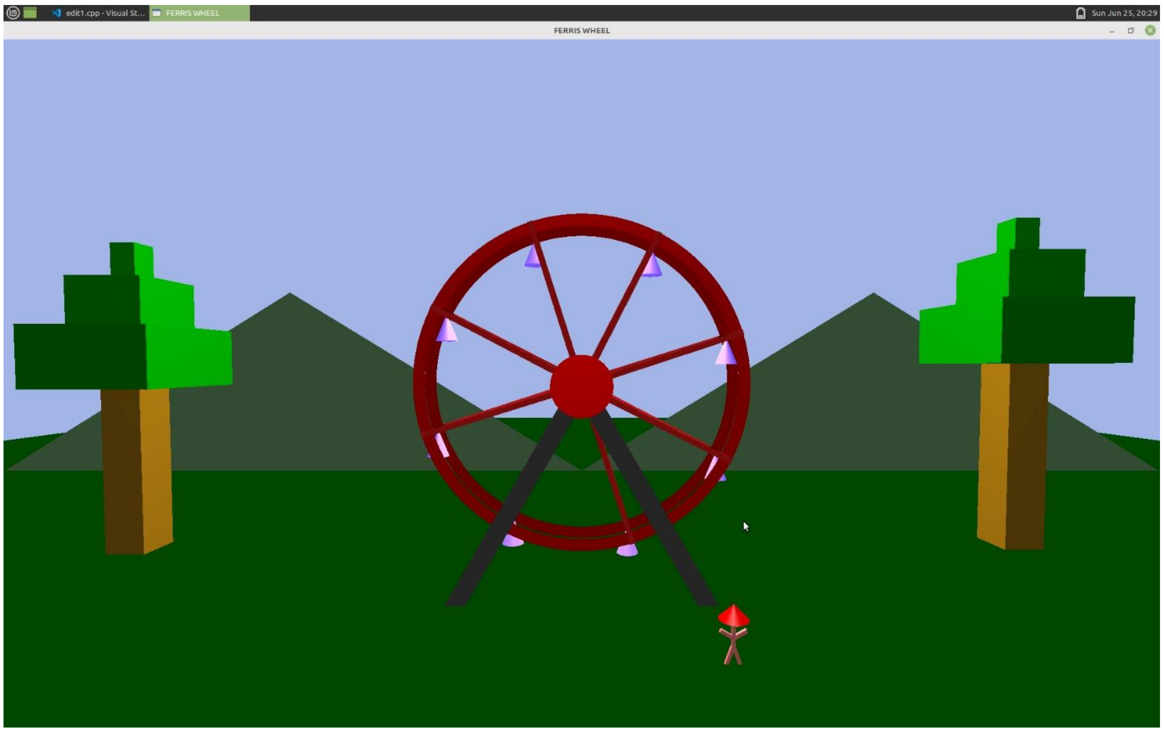


Fig.3.4:Ferrish wheel with lowest number of cabin

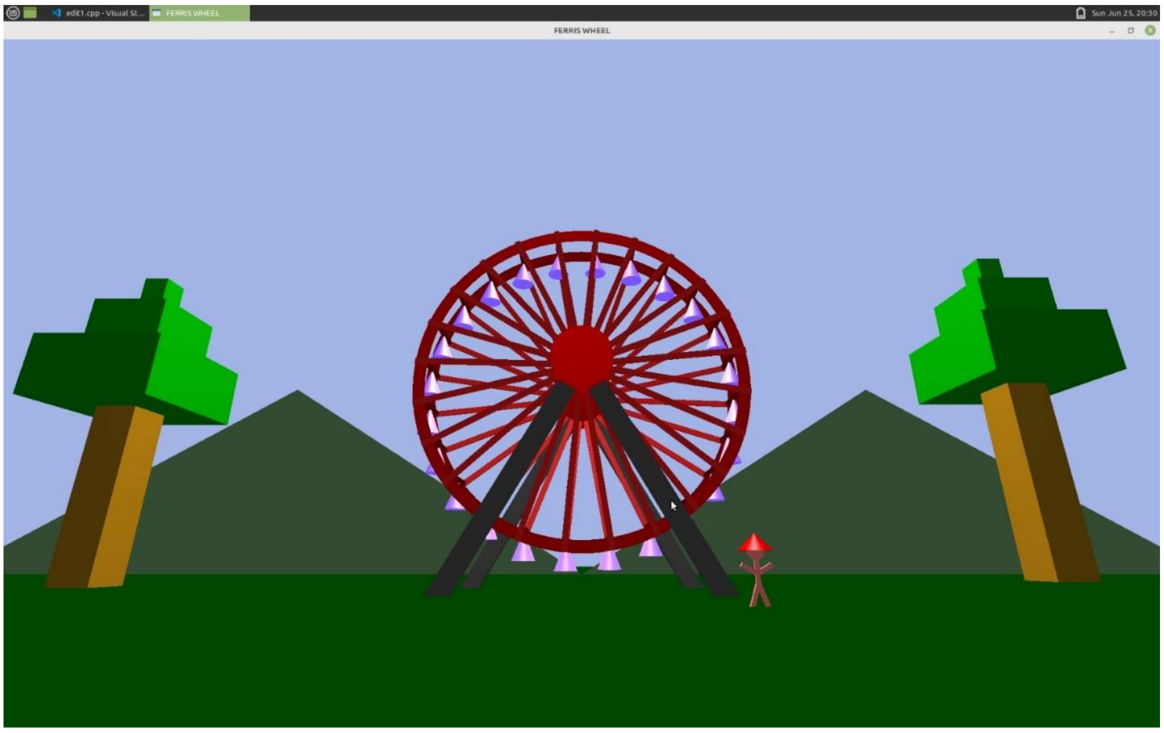


Fig.3.5:Ferrish wheel another view

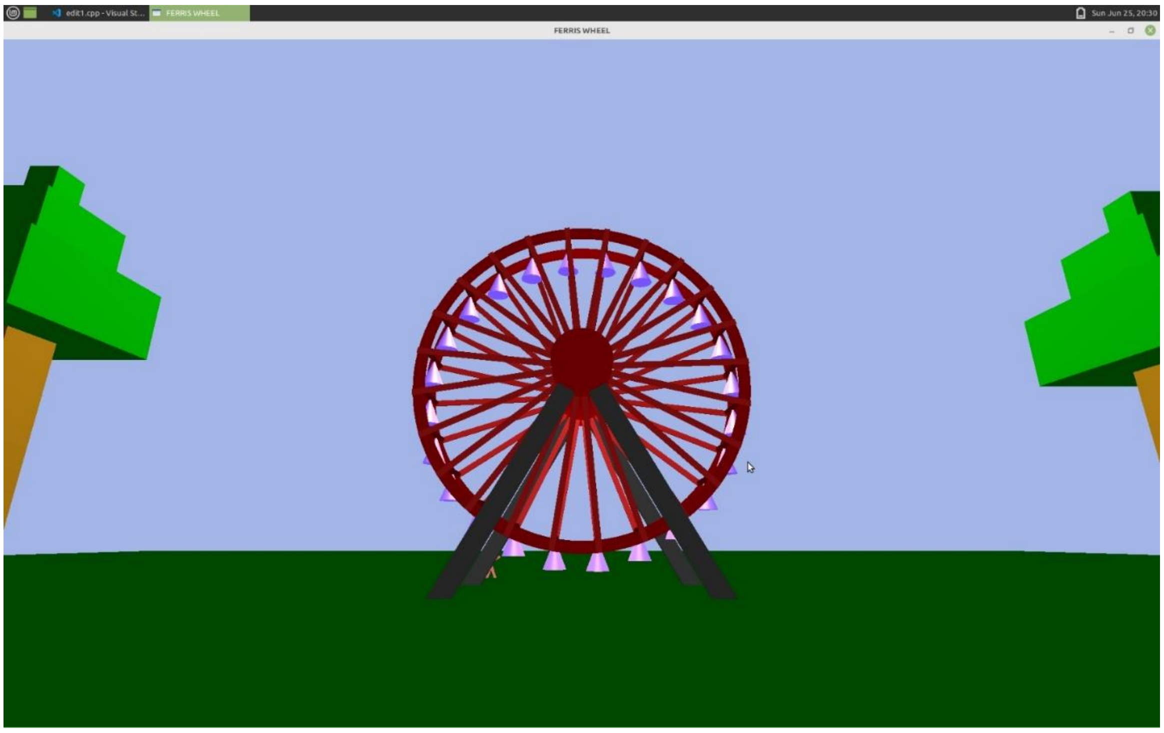


Fig.3.6s:Ferrish wheel back view

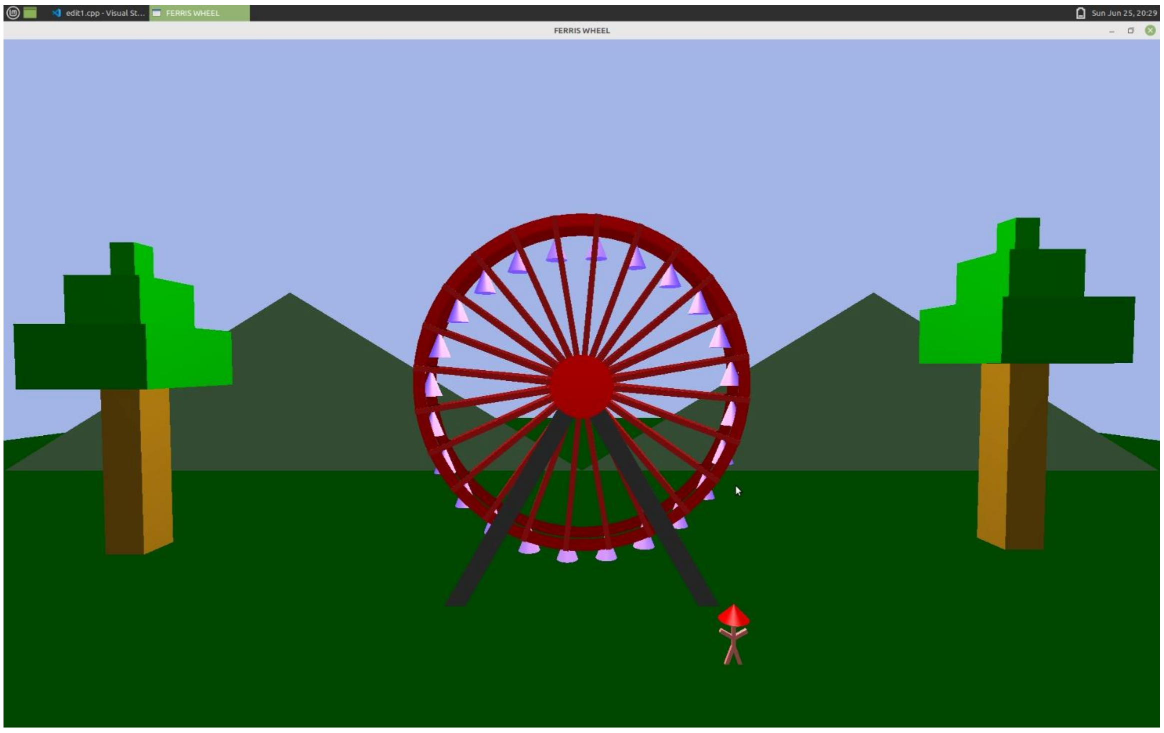


Fig.3.7:Ferrish wheel with maximum number of cabin

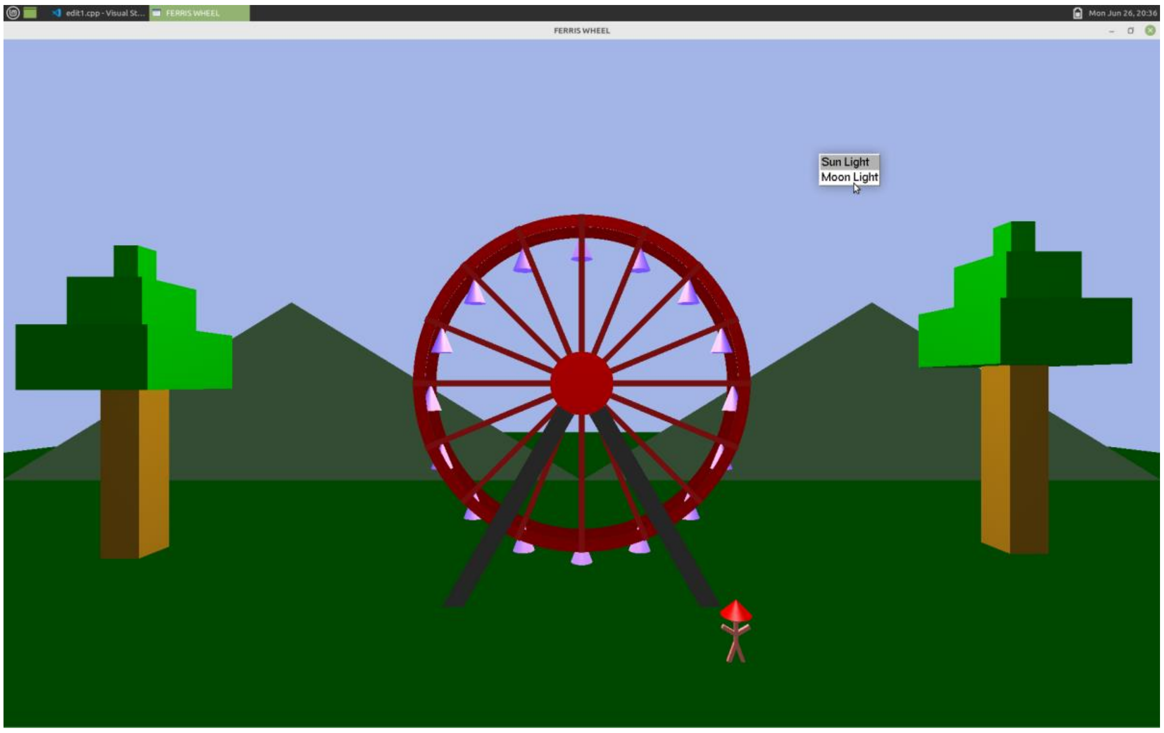


Fig.3.8:Menu drive for choosing sun or moon light

3.2 Discussion

The Ferris wheel consists of a central structure with rotating wheels and a cabin attached to it. The code also includes options to switch between sun and moon lighting modes using a right-click menu. The project demonstrates the use of OpenGL for creating 3D graphics and animation. It utilizes various OpenGL functions and concepts such as lighting, shading, material properties, transformations, and primitive shapes

CONCLUSION AND FUTURE ENHANCEMENT

Conclusion

In conclusion, this project presents a basic implementation of a Ferris wheel animation using OpenGL. It demonstrates fundamental concepts and techniques of 3D graphics, including rendering objects, applying lighting effects, handling user input, and performing animation. The code successfully creates a Ferris wheel structure with rotating wheels and a cabin, accompanied by tree structures and a simple background. It incorporates lighting effects by offering the option to switch between sun and moon lighting modes through a right-click menu.

Future Enchancement

- **Refactoring and Code Optimization:** Enhance the code structure and organization to improve readability, maintainability, and performance. Consider implementing design patterns and modularizing the code into classes and functions to promote reusability and extensibility
- **Camera Control and Interaction:** Implement camera controls to allow users to manipulate the viewing angle, zoom level, and position. This would enhance the user experience and provide more flexibility in exploring the Ferris wheel and its surroundings
- **Texture Mapping and Materials:** Introduce texture mapping techniques to apply realistic textures to the objects in the scene, such as the Ferris wheel structure, cabin, and trees. Additionally, incorporate materials to simulate different surface properties, such as reflectivity, transparency, and shininess

REFERENCES

[1] Interactive Computer Graphics A Top-Down Approach with OpenGL, Edward Angel Addison-Wesely, 5th edition, 2008

[2] Introduction to Computer Graphics 2018 Edition.

[3] "Computer Graphics", Addison-Wesley 1997 James D Foley, Andries Van Dam,Steven.

[4] Engineering & Computer Graphics Workbook using SOLIDWORKS 2019.

APPENDIX – A

SOURCE CODE

```
#include <GL/glut.h>
#include <GL/glu.h>
#include <GL/gl.h>
#include <stdlib.h>
#include <cmath>
#include<stdio.h>
#define M_PI 3.1414
#define WHEEL 1
#define CABIN 2
bool isSunLight = true;
const int MENU_SUNLIGHT = 1;
const int MENU_MOONLIGHT = 2;


void menu(int option) {
switch (option) {
case MENU_SUNLIGHT:
isSunLight = true;
break;
case MENU_MOONLIGHT:
isSunLight = false;
break;
default:
break;
}

glutPostRedisplay();
}

GLfloat global_ambient[] = { 0.3f,0.3f,0.3f,1.0f };

GLfloat dir_ambient[] = { 0.6f,0.6f,0.6f,1.0f };
GLfloat dir_diffuse[] = { 0.8f,0.8f,0.8f,1.0f };
GLfloat dir_specular[] = { 1.0f,1.0f,1.0f,1.0f };

GLfloat dir_position[] = { 0.0f,1.0f,1.0f,0.0f };
GLfloat mat_specular[] = { 0.6,0.6,0.6,1.0 };

GLfloat rotx = 0;
GLfloat roty = 0;

GLint n_bars = 16;
GLfloat c_radius;

GLfloat w_rot;
GLfloat w_speed = 0.25;

bool anim = false;
bool inv_rot = false;

float toRad(float deg)
{
return (deg * M_PI) / 180;
}

void init(void)
{
glutCreateMenu(menu);
glutAddMenuEntry("Sun Light", MENU_SUNLIGHT);
glutAddMenuEntry("Moon Light", MENU_MOONLIGHT);
glutAttachMenu(GLUT_RIGHT_BUTTON);

glLightModelfv(GL_LIGHT_MODEL_AMBIENT, global_ambient);

glLightfv(GL_LIGHT0, GL_AMBIENT, dir_ambient);
glLightfv(GL_LIGHT0, GL_DIFFUSE, dir_diffuse);
glLightfv(GL_LIGHT0, GL_SPECULAR, dir_specular);
glLightfv(GL_LIGHT0, GL_POSITION, dir_position);
glEnable(GL_LIGHT0);

glEnable(GL_COLOR_MATERIAL);
glColorMaterial(GL_FRONT, GL_AMBIENT_AND_DIFFUSE);
glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
glMateriali(GL_FRONT, GL_SHININESS, 30);

glEnable(GL_LIGHTING);
glEnable(GL_LIGHT0);
glEnable(GL_DEPTH_TEST);
glEnable(GL_NORMALIZE);

glShadeModel(GL_SMOOTH);

glNewList(WHEEL, GL_COMPILE);
glColor3f(0.8, 0, 0.0);

glBegin(GL_TRIANGLE_FAN);
glVertex3f(0, 0, 15);
for (float angle = 0; angle <= 360; angle += 0.1)
{
glVertex3f(5 * cos(toRad(angle)), 5 * sin(toRad(angle)), 15);
}
glEnd();
glBegin(GL_QUAD_STRIP);
for (float angle = 0; angle <=360; angle += 0.1) {
glVertex3f(5 * cos(toRad(angle)), 5 * sin(toRad(angle)), 15);
glVertex3f(5 * cos(toRad(angle)), 5 * sin(toRad(angle)), -15);
}
glEnd();
glBegin(GL_TRIANGLE_FAN);
glVertex3f(0, 0, -15);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(5 * cos(toRad(angle)), 5 * sin(toRad(angle)), -15);
}
glEnd();
for (float angle = 0; angle <= 360; angle += 0.1) {
glBegin(GL_POLYGON);
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), 4);
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), 4);
glVertex3f(28 * cos(toRad(angle + 0.2)), 28 * sin(toRad(angle + 0.2)),
4);
glVertex3f(30 * cos(toRad(angle + 0.2)), 30 * sin(toRad(angle + 0.2)),
4);
glEnd();
}

glBegin(GL_QUAD_STRIP);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), 4);
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), 3.5);
}
glEnd();

glBegin(GL_QUAD_STRIP);
```

```
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), 4);
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), 3.5);
}
glEnd();

for (float angle = 0; angle <= 360; angle += 0.1) {
glBegin(GL_POLYGON);
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), 3.5);
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), 3.5);
glVertex3f(28 * cos(toRad(angle + 0.2)), 28 * sin(toRad(angle + 0.2)),
3.5);
glVertex3f(30 * cos(toRad(angle + 0.2)), 30 * sin(toRad(angle + 0.2)),
3.5);
glEnd();
}



for (float angle = 0; angle <= 360; angle += 0.1) {
glBegin(GL_POLYGON);
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), -4);
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), -4);
glVertex3f(28 * cos(toRad(angle + 0.2)), 28 * sin(toRad(angle + 0.2)), -
4);
glVertex3f(30 * cos(toRad(angle + 0.2)), 30 * sin(toRad(angle + 0.2)), -
4);
glEnd();
}

glBegin(GL_QUAD_STRIP);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), -4);
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), -3.5);
}
glEnd();

glBegin(GL_QUAD_STRIP);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), -4);
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), -3.5);
}
glEnd();

for (float angle = 0; angle <= 360; angle += 0.1) {
glBegin(GL_POLYGON);
glVertex3f(30 * cos(toRad(angle)), 30 * sin(toRad(angle)), -3.5);
glVertex3f(28 * cos(toRad(angle)), 28 * sin(toRad(angle)), -3.5);
glVertex3f(28 * cos(toRad(angle + 0.2)), 28 * sin(toRad(angle + 0.2)), -
3.5);
glVertex3f(30 * cos(toRad(angle + 0.2)), 30 * sin(toRad(angle + 0.2)), -
3.5);
glEnd();
}

glEndList();

glNewList(CABIN, GL_COMPILE);
glColor3f(0.5, 0, 0);
glBegin(GL_TRIANGLE_FAN);
glVertex3f(0, 0, -4);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(0.5 * cos(toRad(angle)), 0.5 * sin(toRad(angle)), -4);
}
```

```
glEnd();

glBegin(GL_QUAD_STRIP);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(0.5 * cos(toRad(angle)), 0.5 * sin(toRad(angle)), -3.5);
glVertex3f(0.5 * cos(toRad(angle)), 0.5 * sin(toRad(angle)), 3.5);
}
glEnd();

glBegin(GL_TRIANGLE_FAN);
glVertex3f(0, 0, 4);
for (float angle = 0; angle <= 360; angle += 0.1) {
glVertex3f(0.5 * cos(toRad(angle)), 0.5 * sin(toRad(angle)), 4);
}
glEnd();


glPushMatrix();
glColor3f(0.95, 0.67, 2.06);
glTranslatef(0, -5, 0);
glRotatef(-90, 1, 0, 0);
glutSolidCone(2, 5, 50, 50);
glPopMatrix();

glEndList();
}

void draw_scene() {
if (isSunLight) {
s Set sun light properties
GLfloat sun_light_position[] = {0.0f, 50.0f, 0.0f, 1.0f};
GLfloat sun_light_diffuse[] = {1.0f, 1.0f, 1.0f, 1.0f};
GLfloat sun_light_ambient[] = {0.2f, 0.2f, 0.2f, 1.0f};

glLightfv(GL_LIGHT0, GL_POSITION, sun_light_position);
glLightfv(GL_LIGHT0, GL_DIFFUSE, sun_light_diffuse);
glLightfv(GL_LIGHT0, GL_AMBIENT, sun_light_ambient);

glEnable(GL_LIGHT0);
glDisable(GL_LIGHT1);
} else {
GLfloat moon_light_position[] = {0.0f, -50.0f, 0.0f, 1.0f};
GLfloat moon_light_diffuse[] = {0.7f, 0.7f, 0.7f, 1.0f};
GLfloat moon_light_ambient[] = {0.1f, 0.1f, 0.1f, 1.0f};

glLightfv(GL_LIGHT1, GL_POSITION, moon_light_position);
glLightfv(GL_LIGHT1, GL_DIFFUSE, moon_light_diffuse);
glLightfv(GL_LIGHT1, GL_AMBIENT, moon_light_ambient);

glEnable(GL_LIGHT1);
glDisable(GL_LIGHT0);
}

glPushMatrix();
glTranslatef(0.0, -20.0, -100.0);
glColor3f(0.4f, 0.6f, 0.4f);
glBegin(GL_TRIANGLES);
glVertex3f(-215.0f, -16.0f, 0.0f);
glVertex3f(0.0f, -16.0f, 0.0f);
glVertex3f(-107.5f, 50.0f, 0.0f);
glEnd();
glPopMatrix();

glPushMatrix();
```

```
glTranslatef(0.0, -20.0, -100.0);
glColor3f(0.4f, 0.6f, 0.4f);
glBegin(GL_TRIANGLES);
glVertex3f(0.0f, -16.0f, 0.0f);
glVertex3f(215.0f, -16.0f, 0.0f);
glVertex3f(107.5f, 50.0f, 0.0f);
glEnd();
glPopMatrix();

glPushMatrix();
glColor3f(0.5f, 0.35f, 0.05f); // Brown color for tree trunk
glTranslatef(-95.0f, -20.0f, -15.0f); // Position the tree trunk
glScalef(1.0f, 5.0f, 1.0f); // Scale the tree trunk
glutSolidCube(8.0f); // Draw the tree trunk
glColor3f(0.0f, 0.5f, 0.0f); // Green color for tree branches
glPushMatrix();
glTranslatef(0.0f, 5.0f, 0.0f);
glScalef(5.0f, 0.5f, 5.0f);
glutSolidCube(5.0f);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f, 7.0f, 0.0f);
glScalef(3.0f, 0.5f, 3.0f);
glutSolidCube(5.0f);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f, 8.5f, 0.0f);
glScalef(1.0f, 0.5f, 1.0f);
glutSolidCube(5.0f);
glPopMatrix();

glPopMatrix();

glPushMatrix();
glColor3f(0.5f, 0.35f, 0.05f); // Brown color for tree trunk

glTranslatef(92.0f, -15.0f, -15.0f); // Position the tree trunk
glScalef(1.0f, 5.0f, 1.0f); // Scale the tree trunk
glutSolidCube(8.0f); // Draw the tree trunk
glColor3f(0.0f, 0.5f, 0.0f); // Green color for tree branches

glPushMatrix();
glTranslatef(0.0f, 5.0f, 0.0f);
glScalef(5.0f, 0.5f, 5.0f);
glutSolidCube(5.0f);
glPopMatrix();

glPushMatrix();
glTranslatef(0.0f, 7.0f, 0.0f);
glScalef(3.0f, 0.5f, 3.0f);
glutSolidCube(5.0f);
glPopMatrix();
glPushMatrix();
glTranslatef(0.0f, 8.5f, 0.0f);
glScalef(1.0f, 0.5f, 1.0f);
glutSolidCube(5.0f);
glPopMatrix();

glPopMatrix();


glPushMatrix();
```

```
glTranslatef(0.0, -20.0, -100.0);
glColor3f(0.4f, 0.7f, 0.4f);
glBegin(GL_TRIANGLES);
glVertex3f(-215.0f, -16.0f, 0.0f);
glVertex3f(0.0f, -16.0f, 0.0f);
glVertex3f(-107.5f, 50.0f, 0.0f);
glEnd();

glPopMatrix();
glPushMatrix();
glTranslatef(0.0, -20.0, -100.0);
glColor3f(0.4f, 0.7f, 0.4f);
glBegin(GL_TRIANGLES);
glVertex3f(0.0f, -16.0f, 0.0f);
glVertex3f(215.0f, -16.0f, 0.0f);
glVertex3f(107.5f, 50.0f, 0.0f);
glEnd();
glPopMatrix();

glPushMatrix();
glColor3f(0.0, 0.4, 0.0);
glTranslatef(0, -40, 0);
glScalef(150, 2, 150);
glutSolidCube(4);
glPopMatrix();

glPushMatrix();
glCallList(WHEEL);
glPopMatrix();

glPushMatrix();
glRotatef(w_rot, 0, 0, 1);
for (int i = 0; i < n_bars; i++) {
c_radius = (360.0 / n_bars) * i;
glPushMatrix();
glPushMatrix();
glColor3f(0.92, 0.12, 0.12
glRotatef(c_radius, 0, 0, 1);
glRotatef(-11, 1, 0, 0);
glTranslatef(0, 14.5, 10);
glScalef(1, 28, 0.5);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glColor3f(0.92, 0.12, 0.12);
glRotatef(c_radius, 0, 0, 1);
glRotatef(11, 1, 0, 0);
glTranslatef(0, 14.5, -10);
glScalef(1, 28, 0.5);
glutSolidCube(1);
glPopMatrix();

glTranslatef(28 * cos(toRad(c_radius)), 28 * sin(toRad(c_radius)), 0);
glRotatef(-w_rot, 0, 0, 1);
glCallList(CABIN);
glPopMatrix();
}
glPopMatrix();

glPushMatrix();
glColor3f(0.3, 0.3, 0.3);
glRotatef(-30, 0, 0, 1);
glTranslatef(0, -25, 12);
glScalef(3, 40, 2);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glColor3f(0.3, 0.3, 0.3); wheel)
glRotatef(30, 0, 0, 1);
glTranslatef(0, -25, 12);
glScalef(3, 40, 2);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glColor3f(0.3, 0.3, 0.3);
glRotatef(-30, 0, 0, 1);
glTranslatef(0, -25, -12);
glScalef(3, 40, 2);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glColor3f(0.3, 0.3, 0.3);
glRotatef(30, 0, 0, 1);
glTranslatef(0, -25, -12);
glScalef(3, 40, 2);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glTranslatef(20, -30, 30);
glColor3f(1, 0, 0);
glPushMatrix();
glRotatef(-90, 1, 0, 0);
glutSolidCone(2, 2, 50, 50);
glPopMatrix();
glColor3f(1, 0.5, 0.5);
glutSolidSphere(1, 50, 50);//
glPushMatrix();
glTranslatef(0, -2, 0);
glScalef(0.5, 4, 0.5);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glTranslatef(-0.9, -6, 0);
glRotatef(-20, 0, 0, 1);
glScalef(0.5, 5, 0.5);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glTranslatef(0.9, -6, 0);
glRotatef(20, 0, 0, 1);
glScalef(0.5, 5, 0.5);
glutSolidCube(1);
glPopMatrix();

glPushMatrix();
glTranslatef(-0.9, -2, 0);
glRotatef(-120, 0, 0, 1);
glScalef(0.5, 2, 0.5);
glutSolidCube(1);
glPopMatrix();
```

```
glPushMatrix();
glTranslatef(0.9, -2, 0);
glRotatef(120, 0, 0, 1);
glScalef(0.5, 2, 0.5);
glutSolidCube(1);
glPopMatrix();
glPopMatrix();



}


void display(void) {
glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
glClearColor(0.64, 0.71, 0.9, 1);
glPushMatrix();
glTranslatef(0, 0, -100);
glRotatef(rotx, 1, 0, 0);
glRotatef(roty, 0, 1, 0);
draw_scene();
glPopMatrix();
glutSwapBuffers();
}

void idle() {
if (anim) {
if (inv_rot)
w_rot -= w_speed;
else
w_rot += w_speed;
glutPostRedisplay();
}
}

void reshape(int w, int h) {
if (h == 0)
h = 1;


glViewport(0, 0, w, h);

float fAspect = (GLfloat)w / (GLfloat)h;

glMatrixMode(GL_PROJECTION);
glLoadIdentity();

gluPerspective(65.0, (GLfloat)w / (GLfloat)h, 1.0, 800.0);

glMatrixMode(GL_MODELVIEW);
glLoadIdentity();
}

void keyboard(unsigned char key, int x, int y) {
switch (key) {
case 'w':
case 'W':
rotx += 2.0;
break;
case 's':
case 'S':
rotx -= 2.0;
break;
case 'a':
```

```
case 'A':
roty += 2.0;
break;
case 'd':
case 'D':
roty -= 2.0;
break;
case '+':
case '1':
if (n_bars < 24)
n_bars++;
break;
case '-':
case '0':
if (n_bars > 8)
n_bars--;
break;
case 'x':
case 'X':
w_speed = w_speed - 1.0;
break;
case 'v':
case 'V':
w_speed = w_speed +1.0;
break;
case 'c':
case 'C':
anim = !anim;
break;
case 'i':
case 'I':
inv_rot = !inv_rot;
}
glutPostRedisplay();
}

int main(int argc, char** argv) {
printf("\n**************************************************************
**\n");
printf("\n----------------------------VTU PROJECT-----------------------
---\n");
printf("\n------------------Mr Drupad S & Mr Syed Zeeshan---------------
----\n");
printf("\n--------------------------------------------------------------
-----\n\n");
printf("\n**************************************************************
******\n\n");
printf("\n---------------------FERRIS WHEEL MINI PROJECT----------------
-------\n");
printf("\n---------------------------VIth SEM (17CSL68)-----------------
-------\n");
printf("\n------------Department of Computer Science and Engineering----
-------\n");
printf("\n------------------MAHARAJA INSTITUTE OF TECHNOLOGY------------
-------\n");
printf("\n**************************************************************
*******\n\n");
printf("\n  Top Movement               :w or W\n");
printf("\n  Bottom Movement            :s or S\n");
printf("\n  Rotate Right               :a or A\n");
printf("\n  Rotate Left                :d or D\n");
printf("\n  Wheel Rotation             :c or C\n");
printf("\n  Clockwise & Anti-clockwise :i or I\n");
printf("\n  Increase Speed             :v or V\n");
printf("\n  Decrease Speed              :x or X\n");
printf("\n  Increase No. of Cabin       :+ or 1\n");
printf("\n  Decrease No. of Cabin       :- or 0\n");
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB | GLUT_DEPTH);
glutInitWindowSize(1000, 800);
glutInitWindowPosition(500, 100);
glutCreateWindow("FERRIS WHEEL");
init();
glutReshapeFunc(reshape);
glutIdleFunc(idle);
glutDisplayFunc(display);
glutKeyboardFunc(keyboard);
glutMainLoop();
return 0;
}
```