

## CHAPTER 1

# INTRODUCTION

### 1.1 About Computer Graphics

Computer Graphics in today's world is one of the most widely used powerful and interesting features of the computer which gives us the power to handle the graphical data very efficiently and also process them rapidly and effectively. Computer graphics started with a display of data on hard copy plotters and cathode ray tube screens soon after the introduction of computers themselves. The development of computer graphics has been driven both by the needs of the user community and by advances in hardware and software. Computer graphics today is largely interactive. The user controls the contents, structures and appearances of the object and of their displayed images by using input devices, such as keyboard, mouse or touch-screen. Due to the close relationships between the input devices and the display, the handling of such devices is included in the study of computer graphics.

### 1.2 About OpenGL

OpenGL is a software interface to graphics hardware. OpenGL is designed to work efficiently even if the computer that displays the graphics you create isn't the computer that runs your graphics program [1]. OpenGL is designed as a streamlined, hardware-independent interface to be implemented on many different hardware platforms. OpenGL doesn't provide high-level commands for describing models of three-dimensional objects.

#### 1.2.1 OpenGL Architecture

This is a diagram representing the flow of graphical information, as it is processed from CPU to the frame buffer, in OpenGL.

The upper pipeline is for geometric, vertex-based primitives. The lower pipeline is for pixel-based, image primitives.

Texturing combines the two types of primitives together. There are several APIs related to

OpenGL:

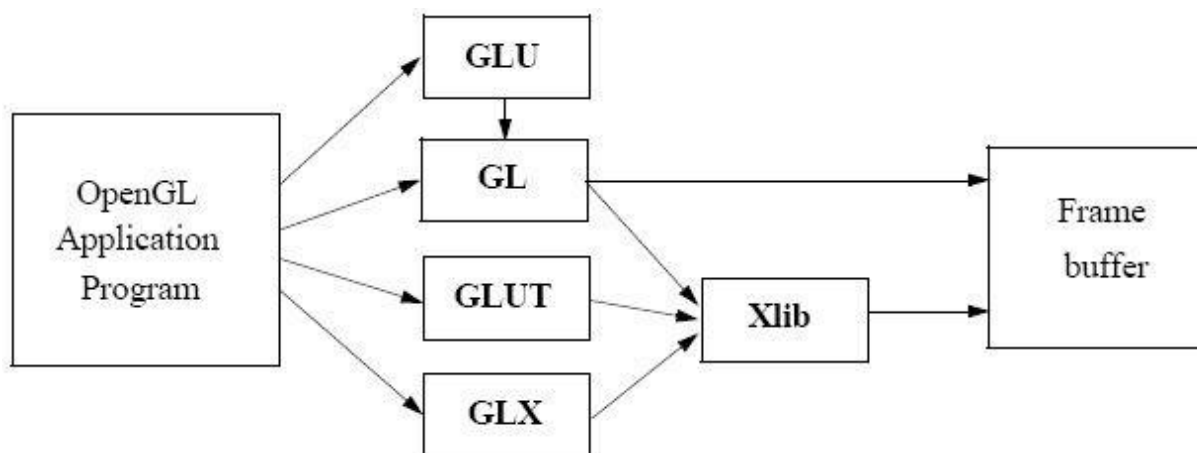
- 1.4.1.1 AGL, WGL, GLX: OpenGL & windowing systems interfaces
- 1.4.1.2 GLU: 2D & 3D Geometry, tessellations, etc.
- 1.4.1.3 GLUT: portable windowing API

### 1.3 OpenGL Library Organization

Support provided by a graphics API:

1. Primitive functions: points, line segments, polygons, text, curves, surfaces.
2. Attribute functions: colour, fill, type face.
3. Viewing functions: attributes of the synthetic camera.
4. Transformation functions: rotation, translation, scaling. Matrix computations.
5. Input functions: keyboard, pointing device.
6. System communication: window system, OS, other workstations, other users.

OpenGL Library Organization:



## 1.4 OpenGL in the Project

In the project, we use a particular graphics software system, OpenGL. The applications are designed to access OpenGL directly through functions in three libraries. The first is the GL library in which the function name begins with gl. The second is the OpenGL utility library (GLU) which uses only GL functions but contains code for creating common objects and simplify viewing and these functions that begin with glu. To interface with the windows system and to get input from the external devices into our programs we use the third library, the OpenGL Utility Toolkit (GLUT) whose functions are prefixed as glut.

## 1.5 OpenGL Functions

### 1.5.1 GLUT Callbacks

OpenGL has a wide variety of functions in GL, GLU, GLUT, and GLE. GLUT uses a callback mechanism to do its event processing. Callbacks simplify event processing for the application developer. Given below are some of the functions used in this project:

glutDisplayFunc()- called when pixels in the window need to be refreshed.

glutMouseFunc()- called when the user presses a mouse button on the mouse.

glutKeyboardFunc()- called when user enters a value (input) through Keyboard.

glutIdleFunc()- called when nothing else is going on.

glutReshapeFunc()- sets the reshape callback for the current window.

### 1.5.2 OpenGL Command

The OpenGL API calls are designed to accept almost any basic data type, which is reflected in the calls name. Knowing how the calls are structured makes it easy to determine which call should be used for a particular data format and size. For instance, vertices from most commercial models are stored as three component floating point vectors. As such, the appropriate OpenGL command to use is glVertex3fv (co-ordinates). For glVertex\*() calls Selective Repeat ARQ which don't specify all the coordinates i.e., glVertex2f().

## CHAPTER 2

### SYSTEM REQUIREMENT

The package is designed such that users with a computer having minimum configuration can also use it, which does not require complex graphics packages.

The package requires simple in-built functions found in the header file along with a few user-defined functions.

#### 2.1 Software Requirements

- Operating System - Ubuntu 14.04
- Language Tool - C Compiler - GNU C Compiler
- Libraries - freeglut3
- Documentation Tool - MS-Word

#### 2.2 Hardware Requirements

- Processor - Intel Pentium onwards Compatible Hardware
- RAM - 256Mb RAM (minimum)
- Hard Disk - 3 GB (minimum)
- Monitor – VGA Compatible
- Keyboard-Standard 101 keys keyboard

## CHAPTER 3

### DESIGN

The project “train arrival-departure” is designed in open GL using several standard header and library functions. When we run the program, it will display the introduction and the instructions and asks user to enter any key and hit enter , then projects gets opened. Creating a Train in computer graphics is not that much hard but logical. In this project we are going to implement an OpenGL Project on the running electric train. The best method for building the objects is by placing the drawing code for each object in separate function. Each object is defined using a coordinate system that makes modeling convenient. Pantograp and joints are done by GL\_LINE\_LOOP & GL\_LINE\_ mode. After we finish the drawing our next aim would be to give motion to the train. As we are going to draw a simple train so our track will be a straight not in zigzag manner. It is made of many boxes or rectangles while wheel is circular. The two parallel line have the electricity flow for the train running. We have electric engine here.

## CHAPTER 4

### IMPLEMENTATION

#### 4.1 Train Arrival-departure

In this mini project, there will be objects like train, signal, and a place called “railway station”. There will be a train in which it will start arriving at the railway station when the signal light “green” is turned on. It will stop at the railway station when the red light is turned on and the train will not move from the railway station till the user displays the green light in the traffic signal when the green light is turned on then the train will start to depart from the railway station. In this project, there will be night mode and day mode. To make it day mode and night mode and to display the signal light to train the controls will be provided from the keyboard and mouse.

- Press ‘g’ or ‘G’ to change the signal light to green.
- Press ‘r’ or ‘R’ to change the signal light to green.
- Press ‘d’ or ‘D’ to make it day.
- Press ‘n’ or ‘N’ to make night.
- Press ‘t’ or ‘T’ to train arrive at station.
- Press right mouse button to display menu.
- Press left mouse button to quit program.
- Press any key and hit enter.

By pressing the right button in the mouse you can see a menu on the screen. Which containing some options and if you click the left button of the mouse we can quit the program. In the night mode, a comet movement is also designed by the controls provided. You can use the option so that it would look great to see.

---

## 4.2 Complete source code

```
#include<stdio.h>
#include<GL/glut.h>
#include <GL/gl.h>
#include <stdlib.h>
#define SPEED 30.0

float i=0.0,m=0.0,n=0.0,o=0.0,c=0.0;

int light=1,day=1,plane=0,comet=0,xm=900,train=0;
char ch;

void declare(char *string)
{
while(*string)
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, *string++);
}

void draw_pixel(GLint cx, GLint cy)
{

glBegin(GL_POINTS);
glVertex2i(cx,cy);
glEnd();
}

void plotpixels(GLint h,GLint k, GLint x,GLint y)
{
draw_pixel(x+h,y+k);
draw_pixel(-x+h,y+k);
draw_pixel(x+h,-y+k);
draw_pixel(-x+h,-y+k);
draw_pixel(y+h,x+k);
draw_pixel(-y+h,x+k);
draw_pixel(y+h,-x+k);
draw_pixel(-y+h,-x+k);
}

void draw_circle(GLint h, GLint k, GLint r)
{
{
GLint d=1-r, x=0, y=r;
while(y>x)
{
plotpixels(h,k,x,y);
if(d<0) d+=2*x+3;
else
{
```

---

```
d+=2*(x-y)+5;
--y;
}
++x;
}
plotpixels(h,k,x,y);
}

void draw_object()
{
int l;
if(day==1)
{
//sky
glColor3f(0.0,0.9,0.9);
glBegin(GL_POLYGON);
glVertex2f(0,380);
glVertex2f(0,700);
glVertex2f(1100,700);
glVertex2f(1100,380);
glEnd();

//sun

for(l=0;l<=35;l++)
{
glColor3f(1.0,0.9,0.0);
draw_circle(100,625,l);
}

//plane
if(plane==1)
{
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(925+n,625+o);
glVertex2f(950+n,640+o);
glVertex2f(1015+n,640+o);
glVertex2f(1030+n,650+o);
glVertex2f(1050+n,650+o);
glVertex2f(1010+n,625+o);
glEnd();

glColor3f(0.8,0.8,0.8);
glBegin(GL_LINE_LOOP);
glVertex2f(925+n,625+o);
glVertex2f(950+n,640+o);
```

---



---

```
glVertex2f(1015+n,640+o);
glVertex2f(1030+n,650+o);
glVertex2f(1050+n,650+o);
glVertex2f(1010+n,625+o);
glEnd();
```

```
}
```

```
//cloud1
```

```
for(l=0;l<=20;l++)
{
glColor3f(1.0,1.0,1.0);
draw_circle(160+m,625,l);
```

```
}
```

```
for(l=0;l<=35;l++)
{
glColor3f(1.0,1.0,1.0);
draw_circle(200+m,625,l);
draw_circle(225+m,625,l);
}
```

```
for(l=0;l<=20;l++)
{
glColor3f(1.0,1.0,1.0);
draw_circle(265+m,625,l);
}
```

```
//cloud2
```

```
for(l=0;l<=20;l++)
{
glColor3f(1.0,1.0,1.0);
draw_circle(370+m,615,l);
}
```

```
for(l=0;l<=35;l++)
{

glColor3f(1.0,1.0,1.0);
draw_circle(410+m,615,l);
draw_circle(435+m,615,l)
```

---

```
draw_circle(470+m,615,l);  
}
```

```
for(l=0;l<=20;l++)  
{  
glColor3f(1.0,1.0,1.0);  
draw_circle(500+m,615,l);  
}
```

```
//grass  
glColor3f(0.0,0.9,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(0,160);  
glVertex2f(0,380);  
glVertex2f(1100,380);  
glVertex2f(1100,160);  
glEnd();  
  
}
```

```
else  
{
```

```
//sky  
glColor3f(0.0,0.0,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(0,380);  
glVertex2f(0,700);  
glVertex2f(1100,700);  
glVertex2f(1100,380);  
glEnd();
```

```
//moon  
int l;
```

```
for(l=0;l<=35;l++)  
{  
glColor3f(1.0,1.0,1.0);  
draw_circle(100,625,l);  
}
```

```
//star1
```

```
glColor3f(1.0,1.0,1.0);
```

---

```
glBegin(GL_TRIANGLES);  
glVertex2f(575,653);  
glVertex2f(570,645);  
glVertex2f(580,645);  
glVertex2f(575,642);  
glVertex2f(570,650);  
glVertex2f(580,650);  
glEnd();
```

```
//star2  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(975,643);  
glVertex2f(970,635);  
glVertex2f(980,635);  
glVertex2f(975,632);  
glVertex2f(970,640);  
glVertex2f(980,640);  
glEnd();
```

```
//star3  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(875,543);  
glVertex2f(870,535);  
glVertex2f(880,535);  
glVertex2f(875,532);  
glVertex2f(870,540);  
glVertex2f(880,540);  
glEnd();
```

```
//star4  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(375,598);  
glVertex2f(370,590);  
glVertex2f(380,590);  
glVertex2f(375,587);  
glVertex2f(370,595);  
glVertex2f(380,595);  
glEnd();
```

```
//star5  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(750,628);  
glVertex2f(745,620);  
glVertex2f(755,620);  
glVertex2f(750,618);  
glVertex2f(745,625);
```

```
glVertex2f(755,625);  
glEnd();
```

```
//star6  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(200,628);  
glVertex2f(195,620);  
glVertex2f(205,620);  
glVertex2f(200,618);  
glVertex2f(195,625);  
glVertex2f(205,625);  
glEnd();
```

```
//star7  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(100,528);  
glVertex2f(95,520);  
glVertex2f(105,520);  
glVertex2f(100,518);  
glVertex2f(95,525);  
glVertex2f(105,525);  
glEnd();
```

```
//star8  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(300,468);  
glVertex2f(295,460);  
glVertex2f(305,460);  
glVertex2f(300,458);  
glVertex2f(295,465);  
glVertex2f(305,465);  
glEnd();
```

```
//star9  
glColor3f(1.0,1.0,1.0);  
glBegin(GL_TRIANGLES);  
glVertex2f(500,543);  
glVertex2f(495,535);  
glVertex2f(505,535);  
glVertex2f(500,532);  
glVertex2f(495,540);  
glVertex2f(505,540);  
glEnd();
```

```
//comet  
if(comet==1)  
{
```

---

---

```
for(l=0;l<=7;l++)
{
glColor3f(1.0,1.0,1.0);
draw_circle(300+c,675,l);
}

glColor3f(1.0,1.0,1.0);
glBegin(GL_TRIANGLES);
glVertex2f(200+c,675);
glVertex2f(300+c,682);
glVertex2f(300+c,668);
glEnd();
}

//Plane
if(plane==1)
{

for(l=0;l<=1;l++)
{
glColor3f(1.0,0.0,0.0);
draw_circle(950+n,625+o,l);
glColor3f(1.0,1.0,0.0);
draw_circle(954+n,623+o,l);

}

}

//grass
glColor3f(0.0,0.3,0.0);
glBegin(GL_POLYGON);
glVertex2f(0,160);
glVertex2f(0,380);
glVertex2f(1100,380);
glVertex2f(1100,160);
glEnd();

}

//track boundary
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(0,150);
glVertex2f(0,160);
glVertex2f(1100,160);
glVertex2f(1100,150);
glEnd();
```

---

```
//platform
```

```
glColor3f(0.560784,0.560784,0.737255);  
glBegin(GL_POLYGON);  
glVertex2f(0,160);  
glVertex2f(0,250);  
glVertex2f(1100,250);  
glVertex2f(1100,160);  
glEnd();
```

```
//table 1
```

```
glColor3f(1.0,0.498039,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(140,190);  
glVertex2f(140,210);  
glVertex2f(155,210);  
glVertex2f(155,190);  
glEnd();
```

```
glColor3f(0.2,0.2,0.2);  
glBegin(GL_POLYGON);  
glVertex2f(130,210);  
glVertex2f(130,215);  
glVertex2f(225,215);  
glVertex2f(225,210);  
glEnd();
```

```
glColor3f(1.0,0.498039,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(200,190);  
glVertex2f(200,210);  
glVertex2f(215,210);  
glVertex2f(215,190);  
glEnd();
```

```
//table 2
```

```
glColor3f(1.0,0.498039,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(390,190);  
glVertex2f(390,210);  
glVertex2f(405,210);  
glVertex2f(405,190);  
glEnd();
```

```
glColor3f(0.2,0.2,0.2);  
glBegin(GL_POLYGON);  
glVertex2f(380,210);
```

```
glVertex2f(380,215);  
glVertex2f(475,215);  
glVertex2f(475,210);  
glEnd();
```

```
glColor3f(1.0,0.498039,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(450,190);  
glVertex2f(450,210);  
glVertex2f(465,210);  
glVertex2f(465,190);  
glEnd();
```

```
//table 3
```

```
glColor3f(1.0,0.498039,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(840,190);  
glVertex2f(840,210);  
glVertex2f(855,210);  
glVertex2f(855,190);  
glEnd();
```

```
glColor3f(0.2,0.2,0.2);  
glBegin(GL_POLYGON);  
glVertex2f(830,210);  
glVertex2f(830,215);  
glVertex2f(925,215);  
glVertex2f(925,210);  
glEnd();
```

```
glColor3f(1.0,0.498039,0.0);  
glBegin(GL_POLYGON);  
glVertex2f(900,190);  
glVertex2f(900,210);  
glVertex2f(915,210);  
glVertex2f(915,190);  
glEnd();
```

```
//below track
```

```
glColor3f(0.623529,0.623529,0.372549);  
glBegin(GL_POLYGON);  
glVertex2f(0,0);  
glVertex2f(0,150);  
glVertex2f(1100,150);  
glVertex2f(1100,0);  
glEnd();
```

```
//Railway track
```

```
glColor3f(0.0,0.0,0.0);
```

---

---

```
glBegin(GL_POLYGON);  
glVertex2f(-100,0);  
glVertex2f(-100,20);  
glVertex2f(1100,20);  
glVertex2f(1100,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(-100,80);  
glVertex2f(-100,100);  
glVertex2f(1100,100);  
glVertex2f(1100,80);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(0,0);  
glVertex2f(0,80);  
glVertex2f(10,80);  
glVertex2f(10,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(80,0);  
glVertex2f(80,80);  
glVertex2f(90,80);  
glVertex2f(90,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(150,0);  
glVertex2f(150,80);  
glVertex2f(160,80);  
glVertex2f(160,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(220,0);  
glVertex2f(220,80);  
glVertex2f(230,80);  
glVertex2f(230,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(290,0);  
glVertex2f(290,80);  
glVertex2f(300,80);  
glVertex2f(300,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(360,0);
```

---



---

```
glVertex2f(360,80);  
glVertex2f(370,80);  
glVertex2f(370,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(430,0);  
glVertex2f(430,80);  
glVertex2f(440,80);  
glVertex2f(440,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(500,0);  
glVertex2f(500,80);  
glVertex2f(510,80);  
glVertex2f(510,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(570,0);  
glVertex2f(570,80);  
glVertex2f(580,80);  
glVertex2f(580,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(640,0);  
glVertex2f(640,80);  
glVertex2f(650,80);  
glVertex2f(650,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(710,0);  
glVertex2f(710,80);  
glVertex2f(720,80);  
glVertex2f(720,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(770,0);  
glVertex2f(770,80);  
glVertex2f(780,80);  
glVertex2f(780,0);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(840,0);  
glVertex2f(840,80);  
glVertex2f(850,80);
```

---

---

```
glVertex2f(850,0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(900,0);
glVertex2f(900,80);
glVertex2f(910,80);
glVertex2f(910,0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(970,0);
glVertex2f(970,80);
glVertex2f(980,80);
glVertex2f(980,0);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(1040,0);
glVertex2f(1040,80);
glVertex2f(1050,80);
glVertex2f(1050,0);
glEnd();

//track boundary
glColor3f(0.647059,0.164706,0.164706);
glBegin(GL_POLYGON);
glVertex2f(-100,100);
glVertex2f(-100,150);
glVertex2f(1100,150);
glVertex2f(1100,100);
glEnd();

//railway station boundary (fence)
glColor3f(0.647059,0.164706,0.164706);
glBegin(GL_POLYGON);
glVertex2f(0,250);
glVertex2f(0,310);
glVertex2f(5,320);
glVertex2f(10,310);
glVertex2f(10,250);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(90,250);
glVertex2f(90,310);
glVertex2f(95,320);
glVertex2f(100,310);
glVertex2f(100,250);
glEnd();
```

---

---

```
glBegin(GL_POLYGON);  
glVertex2f(140,250);  
glVertex2f(140,310);  
glVertex2f(145,320);  
glVertex2f(150,310);  
glVertex2f(150,250);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(190,250);  
glVertex2f(190,310);  
glVertex2f(195,320);  
glVertex2f(200,310);  
glVertex2f(200,250);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(240,250);  
glVertex2f(240,310);  
glVertex2f(245,320);  
glVertex2f(250,310);  
glVertex2f(250,250);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(340,250);  
glVertex2f(340,310);  
glVertex2f(345,320);  
glVertex2f(350,310);  
glVertex2f(350,250);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(390,250);  
glVertex2f(390,310);  
glVertex2f(395,320);  
glVertex2f(400,310);  
glVertex2f(400,250);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(950,250);  
glVertex2f(950,310);  
glVertex2f(955,320);  
glVertex2f(960,310);  
glVertex2f(960,250);  
glEnd();
```

```
glBegin(GL_POLYGON);  
glVertex2f(1000,250);
```

---

```
glVertex2f(1000,310);
glVertex2f(1005,320);
glVertex2f(1010,310);
glVertex2f(1010,250);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(1050,250);
glVertex2f(1050,310);
glVertex2f(1055,320);
glVertex2f(1060,310);
glVertex2f(1060,250);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(950,295);
glVertex2f(950,305);
glVertex2f(1100,305);
glVertex2f(1100,295);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(950,265);
glVertex2f(950,275);
glVertex2f(1100,275);
glVertex2f(1100,265);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(0,295);
glVertex2f(0,305);
glVertex2f(400,305);
glVertex2f(400,295);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(0,265);
glVertex2f(0,275);
glVertex2f(400,275);
glVertex2f(400,265);
glEnd();

//tree 1
glColor3f(0.9,0.2,0.0);
glBegin(GL_POLYGON);
glVertex2f(50,185);
glVertex2f(50,255);
glVertex2f(65,255);
glVertex2f(65,185);
glEnd();
```

---

---

```
for(l=0;l<=30;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(40,250,l);
draw_circle(80,250,l);
}

for(l=0;l<=25;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(50,290,l);
draw_circle(70,290,l);
}

for(l=0;l<=20;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(60,315,l);
}

//tree 2
glColor3f(0.9,0.2,0.0);
glBegin(GL_POLYGON);
glVertex2f(300,185);
glVertex2f(300,255);
glVertex2f(315,255);
glVertex2f(315,185);
glEnd();

for(l=0;l<=30;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(290,250,l);
draw_circle(330,250,l);
}

for(l=0;l<=25;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(300,290,l);
draw_circle(320,290,l);
}

for(l=0;l<=20;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(310,315,l);
}
```

---

---

```
//tree 5
glColor3f(0.9,0.2,0.0);
glBegin(GL_POLYGON);
glVertex2f(1050,185);
glVertex2f(1050,255);
glVertex2f(1065,255);
glVertex2f(1065,185);
glEnd();

for(l=0;l<=30;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(1040,250,l);
draw_circle(1080,250,l);
}

for(l=0;l<=25;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(1050,290,l);
draw_circle(1070,290,l);
}

for(l=0;l<=20;l++)
{
glColor3f(0.0,0.5,0.0);
draw_circle(1060,315,l);
}

//railway station
glColor3f(0.647059,0.164706,0.164706);
glBegin(GL_POLYGON);
glVertex2f(400,250);
glVertex2f(400,450);
glVertex2f(950,450);
glVertex2f(950,250);
glEnd();

//roof
glColor3f(0.556863,0.419608,0.137255);
glBegin(GL_POLYGON);
glVertex2f(350,450);
glVertex2f(450,500);
glVertex2f(900,500);
glVertex2f(1000,450);
glEnd();
//side window
glColor3f(0.556863,0.41
```

---

---

```
9608,0.137255);
glBegin(GL_POLYGON);
glVertex2f(400,400);
glVertex2f(350,350);
glVertex2f(400,350);
glEnd();

//side window
glColor3f(0.556863,0.419608,0.137255);
glBegin(GL_POLYGON);
glVertex2f(950,400);
glVertex2f(1000,350);
glVertex2f(950,350);
glEnd();

glColor3f(0.847059,0.847059,0.74902);
glBegin(GL_POLYGON);
glVertex2f(425,250);
glVertex2f(425,250);
glVertex2f(425,400);
glVertex2f(450,425);
glVertex2f(550,425);
glVertex2f(575,400);
glVertex2f(575,250);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(600,250);
glVertex2f(600,400);
glVertex2f(625,425);
glVertex2f(725,425);
glVertex2f(750,400);
glVertex2f(750,250);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(775,250);
glVertex2f(775,400);
glVertex2f(800,425);
glVertex2f(900,425);
glVertex2f(925,400);
glVertex2f(925,250);
glEnd();

//window 1
glColor3f(0.196078,0.6,0.8);
glBegin(GL_POLYGON);
glVertex2f(450,300);
glVertex2f(450,375);
glVertex2f(550,375);
```

---

---

```
glVertex2f(550,300);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2f(450,337.5);
glVertex2f(550,337.5);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2f(500,375);
glVertex2f(500,300);
glEnd();

//window 2
glColor3f(0.196078,0.6,0.8);
glBegin(GL_POLYGON);
glVertex2f(800,300);
glVertex2f(800,375);
glVertex2f(900,375);
glVertex2f(900,300);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2f(800,337.5);
glVertex2f(900,337.5);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_LINES);
glVertex2f(850,375);
glVertex2f(850,300);
glEnd();

//door
glColor3f(0.329412,0.329412,0.329412);
glBegin(GL_POLYGON);
glVertex2f(625,250);
glVertex2f(625,375);
glVertex2f(725,375);
glVertex2f(725,250);
glEnd();

//signal
glColor3f(1.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(1060,160);
```

---



---

```
glVertex2f(1060,350);
glVertex2f(1070,350);
glVertex2f(1070,160);
glEnd();

glColor3f(0.7,0.7,0.7);
glBegin(GL_POLYGON);
glVertex2f(1040,350);
glVertex2f(1040,500);
glVertex2f(1090,500);
glVertex2f(1090,350);
glEnd();

for(l=0;l<=20;l++)
{
glColor3f(0.0,0.0,0.0);
draw_circle(1065,475,l);
glColor3f(1.0,1.0,0.0);
draw_circle(1065,425,l);
glColor3f(0.0,0.0,0.0);
draw_circle(1065,375,l);
}
if(train==1)
{
//train carrier 3

glColor3f(0.258824,0.435294,0.258824);
glBegin(GL_POLYGON);
glVertex2f(-600+i-xm,50);
glVertex2f(-600+i-xm,300);
glVertex2f(-1000+i-xm,300);
glVertex2f(-1000+i-xm,50);
glEnd();

//top

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(-590+i-xm,300);
glVertex2f(-590+i-xm,310);
glVertex2f(-1010+i-xm,310);
glVertex2f(-1010+i-xm,300);
glEnd();

// Windows

glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(-825+i-xm,175);
glVertex2f(-825+i-xm,285);
```

---

---

```
glVertex2f(-985+i-xm,285);
glVertex2f(-985+i-xm,175);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(-615+i-xm,175);
glVertex2f(-615+i-xm,285);
glVertex2f(-775+i-xm,285);
glVertex2f(-775+i-xm,175);
glEnd();

// carrier 3 Wheels

for(l=0;l<50;l++)
{
glColor3f(0.35,0.16,0.14);
draw_circle(-675+i-xm,50,l);
draw_circle(-925+i-xm,50,l);
}

//train carrier 2

glColor3f(0.258824,0.435294,0.258824);
glBegin(GL_POLYGON);
glVertex2f(-150+i-xm,50);
glVertex2f(-150+i-xm,300);
glVertex2f(-550+i-xm,300);
glVertex2f(-550+i-xm,50);
glEnd();

//top

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(-140+i-xm,300);
glVertex2f(-140+i-xm,310);
glVertex2f(-560+i-xm,310);
glVertex2f(-560+i-xm,300);
glEnd();

// Windows

glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(-375+i-xm,175);
glVertex2f(-375+i-xm,285);
glVertex2f(-535+i-xm,285);
glVertex2f(-535+i-xm,175);
glEnd();

glBegin(GL_POLYGON);
```

---

---

```
glVertex2f(-165+i-xm,175);
glVertex2f(-165+i-xm,285);
glVertex2f(-325+i-xm,285);
glVertex2f(-325+i-xm,175);
glEnd();

//connector

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(-550+i-xm,75);
glVertex2f(-550+i-xm,95);
glVertex2f(-600+i-xm,95);
glVertex2f(-600+i-xm,75);
glEnd();

// carrier 2 Wheels

for(l=0;l<50;l++)
{
glColor3f(0.35,0.16,0.14);
draw_circle(-225+i-xm,50,l);
draw_circle(-475+i-xm,50,l);
}

// train carrier 1

glColor3f(0.258824,0.435294,0.258824);
glBegin(GL_POLYGON);
glVertex2f(300+i-xm,50);
glVertex2f(300+i-xm,300);
glVertex2f(-100+i-xm,300);
glVertex2f(-100+i-xm,50);
glEnd();

//top

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(310+i-xm,300);
glVertex2f(310+i-xm,310);
glVertex2f(-110+i-xm,310);
glVertex2f(-110+i-xm,300);
glEnd();

// Windows

glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(75+i-xm,175);
```

---

---

```
glVertex2f(75+i-xm,285);
glVertex2f(-85+i-xm,285);
glVertex2f(-85+i-xm,175);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(285+i-xm,175);
glVertex2f(285+i-xm,285);
glVertex2f(125+i-xm,285);
glVertex2f(125+i-xm,175);
glEnd();

//connector

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(-100+i-xm,75);
glVertex2f(-100+i-xm,95);
glVertex2f(-150+i-xm,95);
glVertex2f(-150+i-xm,75);
glEnd();

// carrier 1 Wheels

for(l=0;l<50;l++)
{
glColor3f(0.35,0.16,0.14);
draw_circle(-25+i-xm,50,l);
draw_circle(225+i-xm,50,l);
}

//train base

glColor3f(0.196078,0.6,0.8);
glBegin(GL_POLYGON);
glVertex2f(350+i-xm,50);
glVertex2f(350+i-xm,125);
glVertex2f(755+i-xm,125);
glVertex2f(820+i-xm,50);
glEnd();

//train control chamber

glColor3f(1.0,0.25,0.0);
glBegin(GL_POLYGON);
glVertex2f(360+i-xm,125);
glVertex2f(360+i-xm,325);
glVertex2f(560+i-xm,325);
glVertex2f(560+i-xm,125);
glEnd();
```

---

---

```
//window
```

```
glColor3f(1.0,1.0,1.0);
glBegin(GL_POLYGON);
glVertex2f(375+i-xm,175);
glVertex2f(375+i-xm,315);
glVertex2f(545+i-xm,315);
glVertex2f(545+i-xm,175);
glEnd();
```

```
//train top
```

```
glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(350+i-xm,325);
glVertex2f(350+i-xm,350);
glVertex2f(570+i-xm,350);
glVertex2f(570+i-xm,325);
glEnd();
```

```
//tain engine
```

```
glColor3f(1.0,0.5,0.0);
glBegin(GL_POLYGON);
glVertex2f(560+i-xm,125);
glVertex2f(560+i-xm,225);
glVertex2f(755+i-xm,225);
glVertex2f(755+i-xm,125);
glEnd();
```

```
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(580+i-xm,125);
glVertex2f(580+i-xm,225);
glVertex2f(590+i-xm,225);
glVertex2f(590+i-xm,125);
glEnd();
```

```
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(600+i-xm,125);
glVertex2f(600+i-xm,225);
glVertex2f(610+i-xm,225);
glVertex2f(610+i-xm,125);
glEnd();
```

```
glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(735+i-xm,125);
glVertex2f(735+i-xm,225);
glVertex2f(745+i-xm,225);
```

```
glVertex2f(745+i-xm,125);
glEnd();

//tain smoke

glColor3f(0.196078,0.6,0.9);
glBegin(GL_POLYGON);
glVertex2f(650+i-xm,225);
glVertex2f(650+i-xm,275);
glVertex2f(700+i-xm,275);
glVertex2f(700+i-xm,225);
glEnd();

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(640+i-xm,275);
glVertex2f(640+i-xm,300);
glVertex2f(710+i-xm,300);
glVertex2f(710+i-xm,275);
glEnd();

//train head-light

glColor3f(1.0,1.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(755+i-xm,225);
glVertex2f(765+i-xm,225);
glVertex2f(765+i-xm,185);
glVertex2f(755+i-xm,185);
glEnd();

glColor3f(0.0,0.0,0.0);
glBegin(GL_POLYGON);
glVertex2f(755+i-xm,225);
glVertex2f(785+i-xm,225);
glVertex2f(755+i-xm,205);

glEnd();

// train connector

glColor3f(0.309804,0.184314,0.184314);
glBegin(GL_POLYGON);
glVertex2f(350+i-xm,75);
glVertex2f(350+i-xm,95);
glVertex2f(300+i-xm,95);
glVertex2f(300+i-xm,75);
glEnd();

//train wheels
```

```
for(l=0;l<50;l++)
{
    glColor3f(0.35,0.16,0.14);
    draw_circle(425+i-xm,50,l);
    draw_circle(700+i-xm,50,l);
}
}
//Railway Station Board

glColor3f(0.5,0.5,0.5);
glBegin(GL_POLYGON);
glVertex2f(580,500);
glVertex2f(580,520);
glVertex2f(590,520);
glVertex2f(590,500);
glEnd();

glColor3f(0.5,0.5,0.5);
glBegin(GL_POLYGON);
glVertex2f(770,500);
glVertex2f(770,520);
glVertex2f(780,520);
glVertex2f(780,500);
glEnd();

glColor3f(0.435294,0.258824,0.258824);
glBegin(GL_POLYGON);
glVertex2f(560,510);
glVertex2f(560,540);
glVertex2f(580,550);
glVertex2f(780,550);
glVertex2f(800,540);
glVertex2f(800,510);
glEnd();

glColor3f(1.0,1.0,1.0);
glRasterPos2f(570,520);
declare("RAILWAY STATION");

glFlush();
}

void traffic_light()
{
    int l;
```

---

```
if(light==1)
{
for(l=0;l<=20;l++)
{

glColor3f(0.0,0.0,0.0);
draw_circle(1065,475,l);

glColor3f(0.0,0.7,0.0);
draw_circle(1065,375,l);
}
}

else
{

for(l=0;l<=20;l++)
{
glColor3f(1.0,0.0,0.0);
draw_circle(1065,475,l);

glColor3f(0.0,0.0,0.0);
draw_circle(1065,375,l);
}
}

void idle()
{
glClearColor(1.0,1.0,1.0,1.0);

if(light==0 && (i>=0 && i<=1150))
{

i+=SPEED/10;
m+=SPEED/150;
n-=2;
o+=0.2;
c+=2;

}

if(light==0 && (i>=2600 && i<=3000))

i+=SPEED/10;
m+=SPEED/150;
n-=2;
o+=0.2;
c+=2;
```

---



---

```
    }

    if(light==0)
    {
        i=i;
        m+=SPEED/150;
        n-=2;
        o+=0.2;
        c+=2;
    }

    else
    {

        i+=SPEED/10;
        m+=SPEED/150;
        n-=2;
        o+=0.2;
        c+=2;
    }
    if(i>3500)
    i=0.0;
    if(m>1100)
    m=0.0;
    if( o>75)
    {
        plane=0;
    }
    if(c>500)
    {
        comet=0;
    }
    glutPostRedisplay();

}

void mouse(int btn,int state,int x,int y)
{

    if(btn==GLUT_LEFT_BUTTON && state==GLUT_UP)
    exit(0);
}
void keyboardFunc( unsigned char key, int x, int y )
{
    switch( key )
    {
        case 'g':
        case 'G':
            light=1;
            break;
```

---

```
    case 'r':
    case 'R':
    light=0;
    break;

    case 'd':
    case 'D':
    day=1;
    break;

    case 'n':
    case 'N':
    day=0;
    break;

    case 't':
    case 'T':
    train=1;
    i=0;
    break;

};

}

void main_menu(int index)
{
    switch(index)
    {
        case 1:
        if(index==1)

        {
            plane=1;
            o=n=0.0;
        }
        break;

        case 2:
        if(index==2)
        {
            comet=1;
            c=0.0;
        }
        break;
    }
}

void myinit()
```

---

```

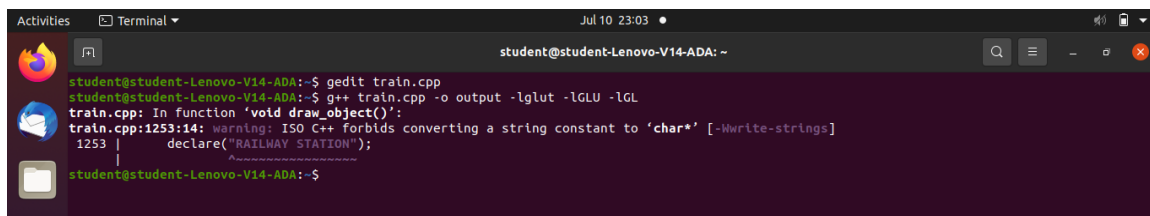
{
glClearColor(1.0,1.0,1.0,1.0);
glColor3f(0.0,0.0,1.0);
glPointSize(2.0);
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0.0,1100.0,0.0,700.0);
}
void display()
{
glClear(GL_COLOR_BUFFER_BIT);
draw_object();
traffic_light();
glFlush();
}
int main(int argc,char** argv)
{
int c_menu;
printf("Project by CSEMiniProjects.com\n");
printf("-----");
printf("          ARRIVAL AND DEPARTURE OF TRAIN          ");
printf("-----");
printf("Press 'r' or 'R' to change the signal light to red. \n\n");
printf("Press 'g' or 'G' to change the signal light to green. \n\n");
printf("Press 'd' or 'D' to make it day. \n\n");
printf("Press 'n' or 'N' to make it night. \n\n");
printf("Press 't' or 'T' Train arrive at station.\n\n");
printf("Press RIGHT MOUSE BUTTON to display menu. \n\n");
printf("Press LEFT MOUSE BUTTON to quit the program. \n\n\n");
printf("Press any key and Hit ENTER.\n");
scanf("%s",&ch);
glutInit(&argc,argv);
glutInitDisplayMode(GLUT_SINGLE|GLUT_RGB);
glutInitWindowSize(1100.0,700.0);
glutInitWindowPosition(0,0);
glutCreateWindow("Traffic Control");
glutDisplayFunc(display);
glutIdleFunc(idle);
glutKeyboardFunc(keyboardFunc);
glutMouseFunc(mouse);
myinit();
c_menu=glutCreateMenu(main_menu);
        glutAddMenuEntry("Aeroplane",1);
glutAddMenuEntry("Comet",2);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMainLoop();
return 0;
}

```

## CHAPTER 5

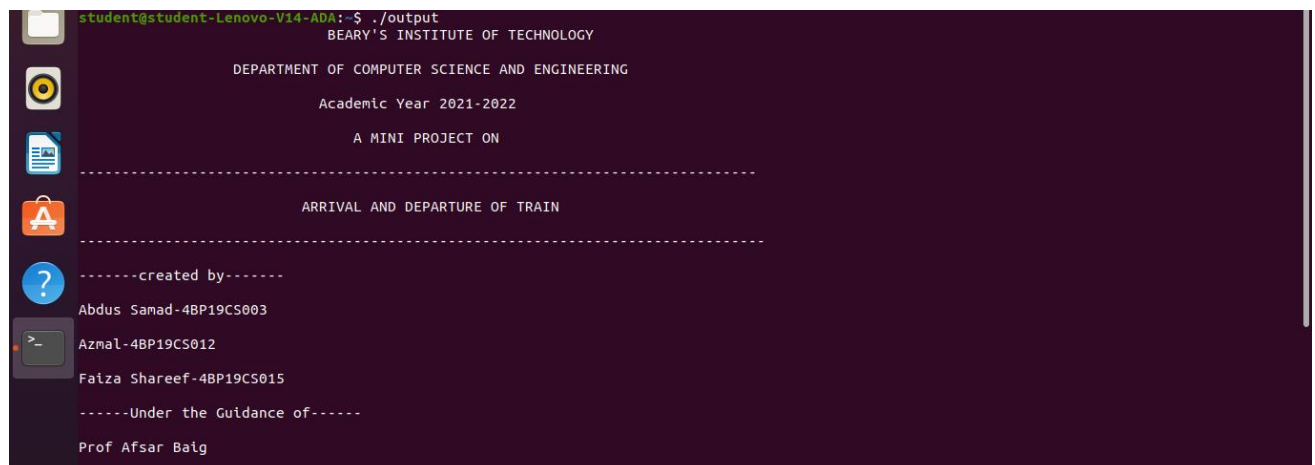
### RESULTS

#### SCREENSHOTS



```
student@student-Lenovo-V14-ADA: ~  
student@student-Lenovo-V14-ADA:~$ gedit train.cpp  
student@student-Lenovo-V14-ADA:~$ g++ train.cpp -o output -lglut -lGLU -lGL  
train.cpp: In function 'void draw_object()':  
train.cpp:1253:14: warning: ISO C++ forbids converting a string constant to 'char*' [-Wwrite-strings]  
1253 |         declare("RAILWAY STATION");  
      |         ^~~~~~  
student@student-Lenovo-V14-ADA:~$
```

Figure 5.1 Reading the Input



```
student@student-Lenovo-V14-ADA:~$ ./output  
BEARY'S INSTITUTE OF TECHNOLOGY  
  
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING  
  
Academic Year 2021-2022  
  
A MINI PROJECT ON  
-----  
ARRIVAL AND DEPARTURE OF TRAIN  
-----  
-----created by-----  
Abdus Samad-4BP19CS003  
Azmal-4BP19CS012  
Faiza Shareef-4BP19CS015  
-----Under the Guidance of-----  
Prof Afsar Baig
```

Figure 5.2 Introduction screen

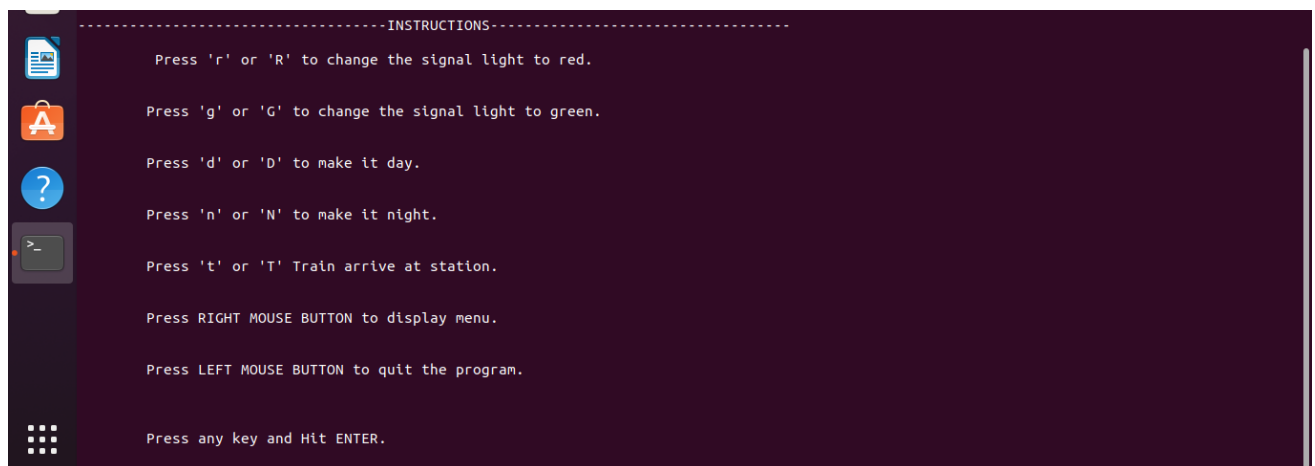


Figure 5.3 Instructions

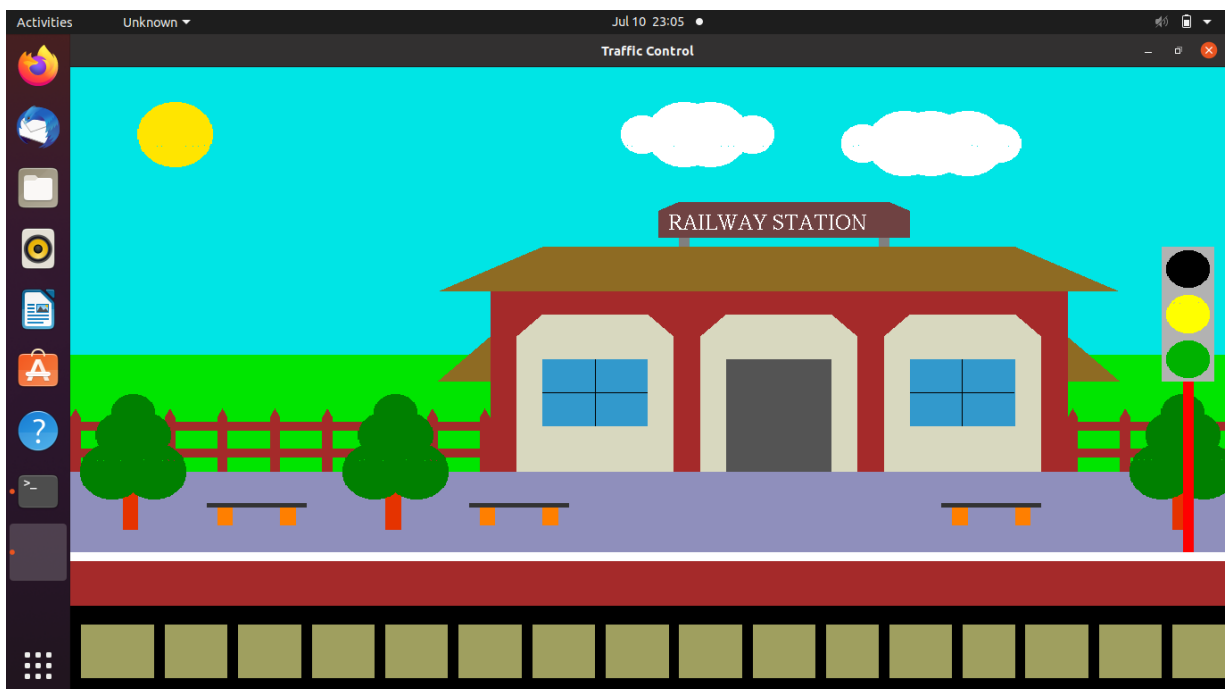


Figure 5.4 Railway Station

Figure 5.5 Day Mode

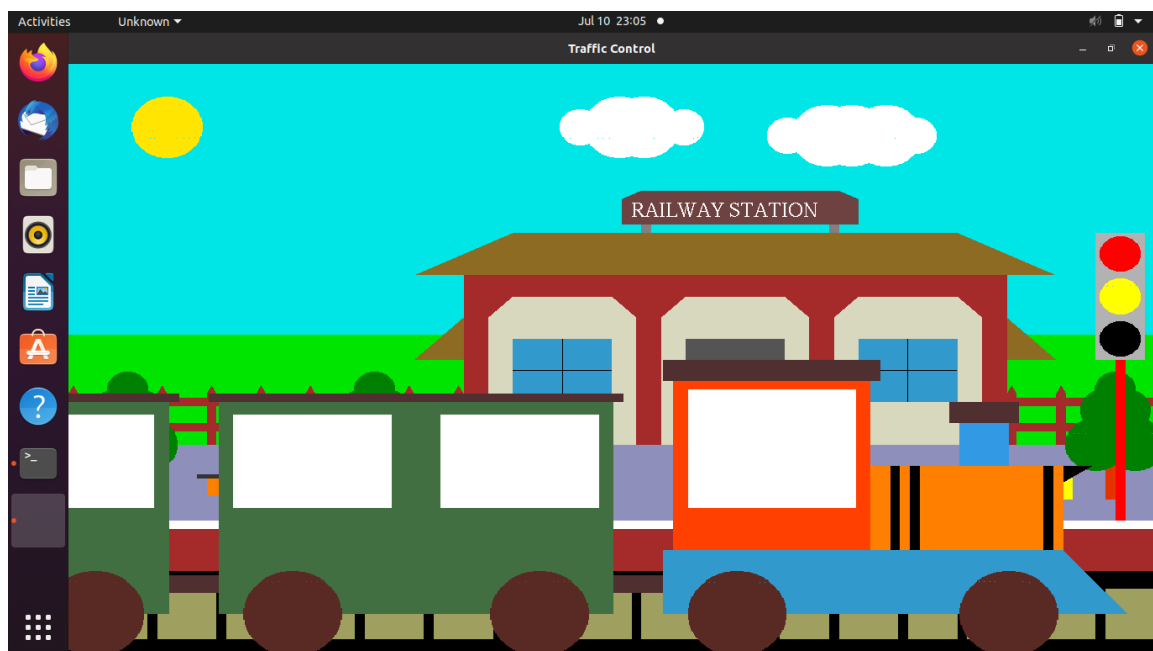
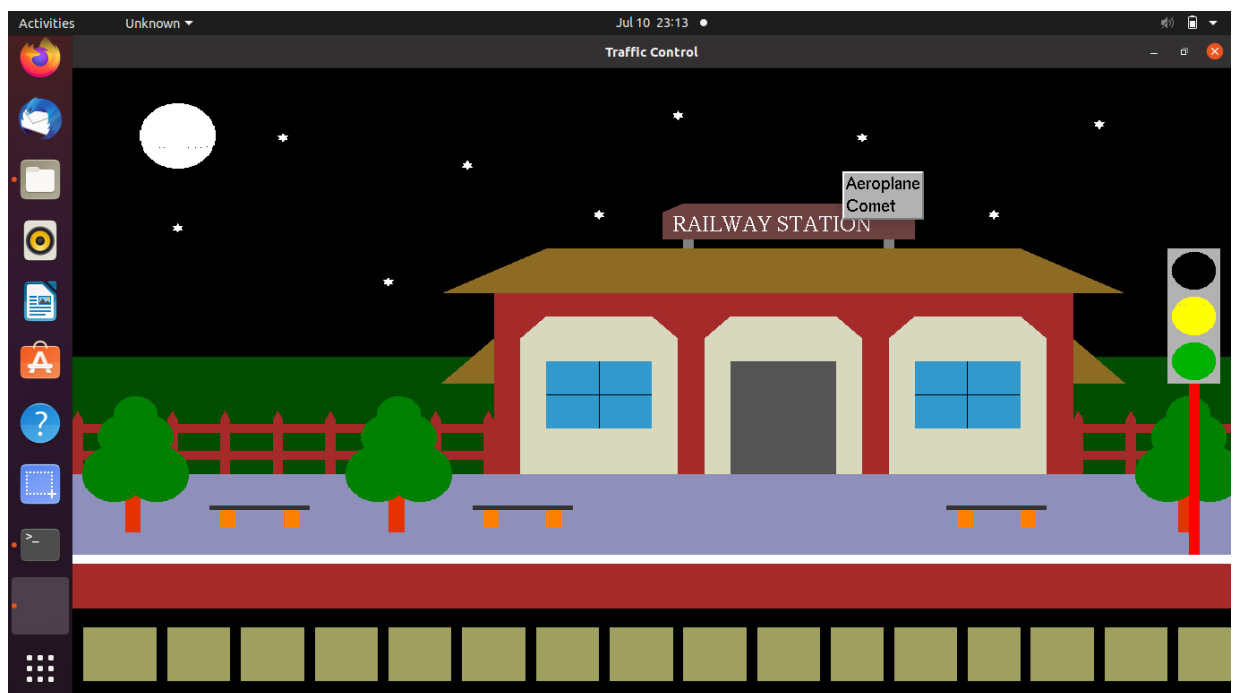
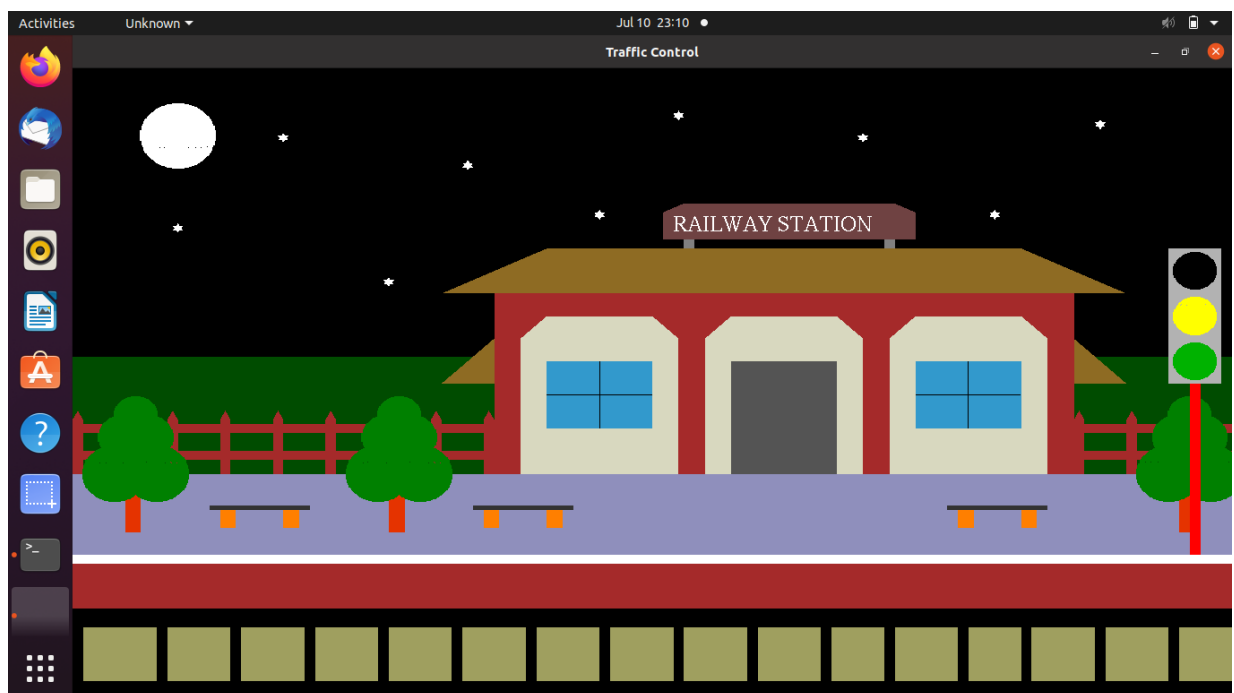
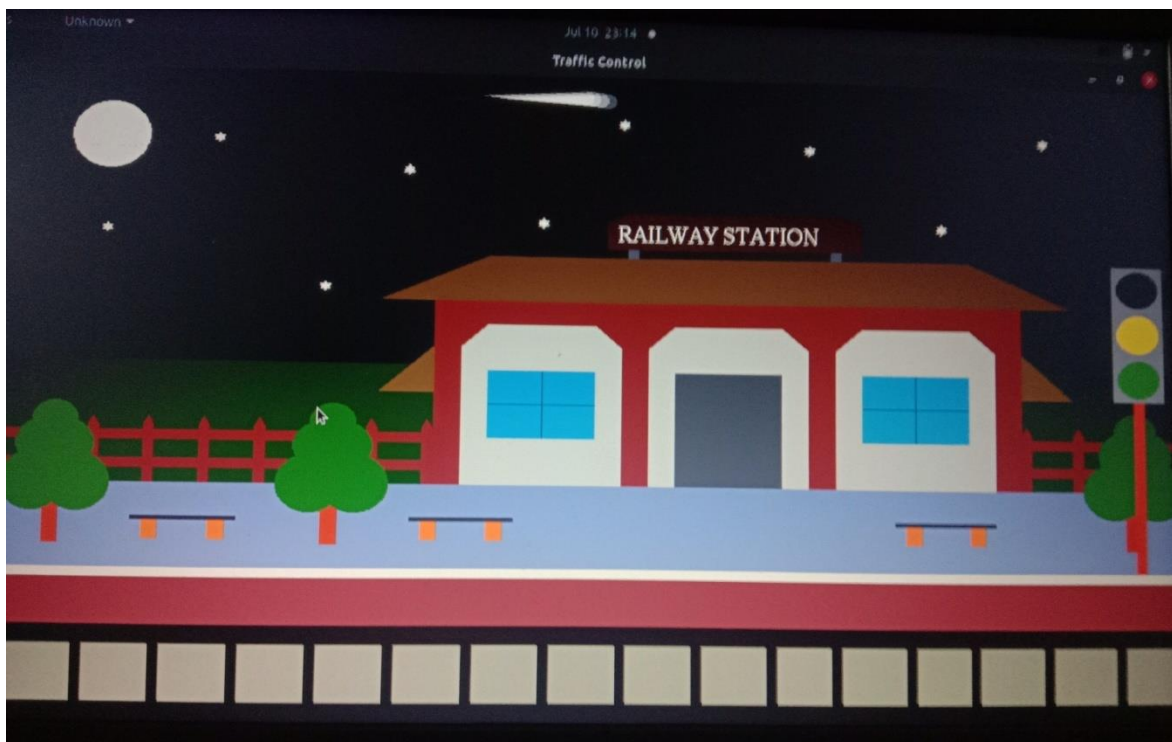
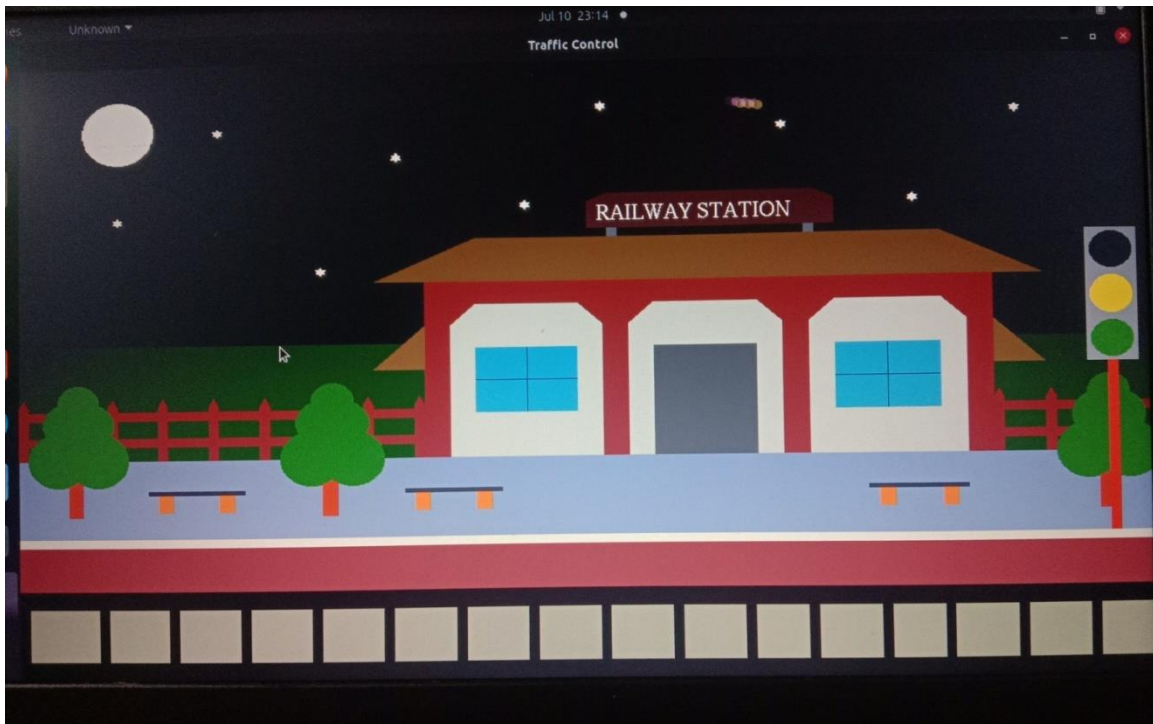




Figure 5.6 Night Mode







## CONCLUSION

The wonderful experience developing this project. By working on the project, we gained hands-on experience of using the OpenGL software using which we experimented this project. We have made use of hardware devices such as mouse and keyboard driven interface, thus to reduce the complexity and make it user friendly. The project helped in understanding the working of computer graphics using OpenGL and various concepts, functions and methodologies for the development of a graphics packages. The proposed system will serve its purpose without any hassles.

---

## REFERENCES

- [1]. Edward Angel, Interactive Computer Graphics a Top-Down Approach Using OpenGL, 5th edition, Dorling Kindersley India Pvt. Ltd., UP, India, 2009.
- [2]. Computer Graphics Using OpenGL – F.S. Hill, Jr. 2nd Edition, Pearson Education, 2001.
- [4]. Computer Graphics – James D Foley, Andres Van Dam, Steven K Feigner, John F Hughes, Addison-Wesley 1997.
- [5]. Computer Graphics - OpenGL Version – Donald Hearn and Pauline Baker, 2nd Edition, Pearson Education.

