BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Bengaluru-560 004

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

COMPUTER GRAPHICS & VISULIZATION
MINI PROJECT (17CSL68)

"MOVING TRAIN"

Submitted By

USN	NAME
1BI17CS190	ANURAG KUMAR

for the academic year 2020-21

Under the guidance of

K J BHANUSHREE
(Assistant Professor)
Name of Lab In charge
Designation

VARSHITHA KC
(Assistant Professor)
Name of Lab in charge
Designation

VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi-590018, Karnataka

BANGALORE INSTITUTE OF TECHNOLOGY
K.R. Road, V.V. Puram, Be ngaluru-560 004

Department of Computer Science & Engineering

Certificate

This is to certify that the implementation of **Computer Graphics and Visualization MINI PROJECT (17CSL68)** entitled **"MOVING TRAIN"** has been successfully completed by

USN	NAME
1BI17CS190	**ANURAG KUMAR**

of VI semester B.E. for the partial fulfillment of the requirements for the Bachelor's degree in **Computer Science & Engineering** of the **Visvesvaraya Technological University** during the academic year **2020-2021**.

Lab In charges:

K J BHANUSHREE
Assistant Professor
Dept. of CS&E, BIT

VARSHITHA KC
Assistant Professor
Dept. of CS&E, BIT

Dr. ASHA T.
Professor and Head
Department of CS&E
Bangalore
Institute of Technology

Examiners: 1) 2)

ACKNOWLEDGEMENT

The knowledge & satisfaction that accompany the successful completion of any task would be incomplete without mention of people who made it possible, whose guidance and encouragement crowned my effort with success. I would like to thank all and acknowledge the help I have received to carry out this Mini Project.

I would like to convey my thanks to Head of Department **Dr. ASHA T.** for being kind enough to provide the necessary support to carry out the mini project.

I am most humbled to mention the enthusiastic influence provided by the lab in-charges **Prof. K J BHANUSHREE a**nd **Prof. VARSHITHA K C** , on the project for their ideas, time to time suggestions for being a constant guide and co-operation showed during the venture and making this project a great success.

I would also take this opportunity to thank my friends and family for their constant support and help. I'm very much pleasured to express my sincere gratitude to the friendly co-operation showed by all the **staff members** of Computer Science Department, BIT.

ANURAG KUMAR
1BI17CS190

Table of contents

Chapter -1

INTRODUCTION

1.1 Computer Graphics

Computer graphics is an art of drawing pictures, lines, charts, using computers with the help of programming. Computer graphics is made up of number of pixels. Pixel is the smallest graphical picture or unit represented on the computer screen. Basically, there are 2 types of computer graphics namely,

Interactive Computer Graphics involves a two-way communication between computer and user. The observer is given some control over the image by providing him with an input device. This helps him to signal his request to the computer.

Non-Interactive Computer Graphics otherwise known as passive computer graphics it is the computer graphics in which user does not have any kind of control over the image. Image is merely the product of static stored program and will work according to the instructions given in the program linearly. The image is totally under the control of program instructions not under the user. Example: screen savers.

1.2 Applications of Computer Graphics

Scientific Visualization

Scientific visualization is a branch of science, concerned with the visualization of three-dimensional phenomena, such as architectural, meteorological, medical, biological systems.

Graphic Design

The term graphic design can refer to a number of artistic and professional disciplines which focus on visual communication and presentation

Computer-aided Design

Computer-aided design (CAD) is the use of computer technology for the design of objects, real or virtual. The design of geometric models for object shapes, in particular, is often called computer-aided geometric design (CAGD).

The manufacturing process is tied in to the computer description of the designed objects so that the fabrication of a product can be automated using methods that are referred to as CAM, computer-aided manufacturing.

Web Design

Web design is the skill of designing presentations of content usually hypertext or hypermedia that is delivered to an end-user through the World Wide Web, by way of a Web browser.

Digital Art

Digital art most commonly refers to art created on a computer in digital form.

Video Games

A video game is an electronic game that involves interaction with a user interface to generate visual feedback on a raster display device.

Virtual Reality

Virtual reality (VR) is a technology which allows a user to interact with a computer simulated environment. The simulated environment can be similar to the real world. This allows the designer to explore various positions of an object. Animations in virtual reality environments are used to train heavy equipment operators or to analyse the effectiveness of various cabin configurations and control placements.

Computer Simulation

A computer simulation, a computer model or a computational model is a computer program, or network of computers, that attempts to simulate an abstract model of a particular system.

Education and Training

Computer simulations have become a useful part of mathematical modelling of many natural systems in physics, chemistry and biology, human systems in economics, psychology, and social science and in the process of engineering new technology, to gain insight into the operation of those systems, or to observe their behaviour. Most simulators provide screens for visual display of the external environment with multiple panels is mounted in front of the simulator.

Image Processing

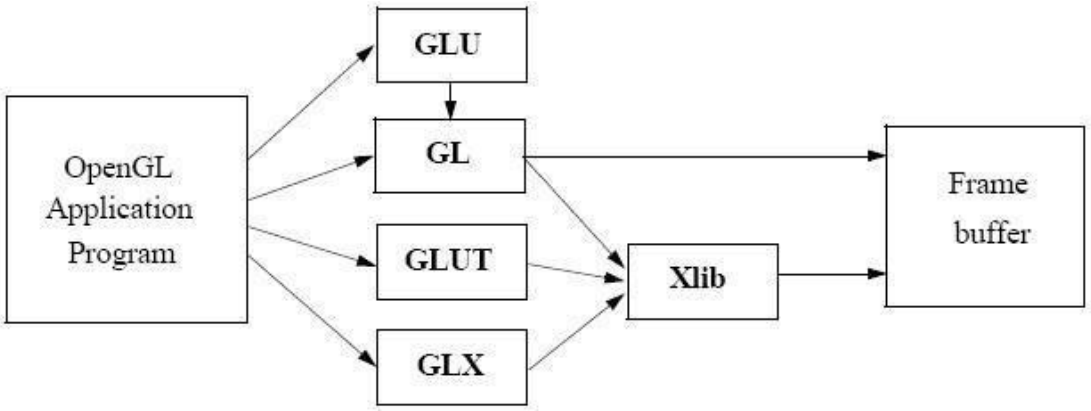
The modification or interpretation of existing pictures such as photographs and TV scans, is called image processing. In computer graphics, a computer is used to create a picture. Image processing techniques, on the other hand, are used to improve picture quality, analyse images, or recognize visual patterns for robotics applications.

1.3 OpenGL

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 the three libraries. Functions in main GL libraries have names that begin with the letters gl and are stored in a library usually referred to as GL.

The second is the OpenGL Utility Library (GLU). This library uses only GL functions but contains code for creating common objects and simplifying viewing. All functions in GLU can be created from the core GL library. The GLU library is available in all OpenGL implementations. Functions in the GLU library starts with the letters glu.

The third is the OpenGL Utility Toolkit (GLUT). It provides the minimum functionality that should be formulated in modern windowing systems.



1.4 Objectives of the Project

• To show the working of the gluPerspective in appearance of the objects used ingame.

- To show the implementation of Textures for a better appearance of the real-world objects.
- To show the implementation of the OpenGL transformation functions.
- To show the working of lighting and shading.
- To show the user and programme interaction using input device.

1.5 Problem Statement

To demonstrate a computer graphics mini project that is developed using OpenGL(c++) where a train will be arriving at the railway station along with many other visual effects.

1.6 Organisation of the Project

The project was organised in a systematic way. First we analysed what are the basic features to be included in the project to make it acceptable. As it is a graphics oriented project, we made the sketches prior, so as to have an idea like how our output must look like. After all these, the source code was formulated as a paper work. All the required software were downloaded. Finally, the successful implementation of the project.

Chapter -2

SYSTEM REQUIREMENTS

Hardware Requirements

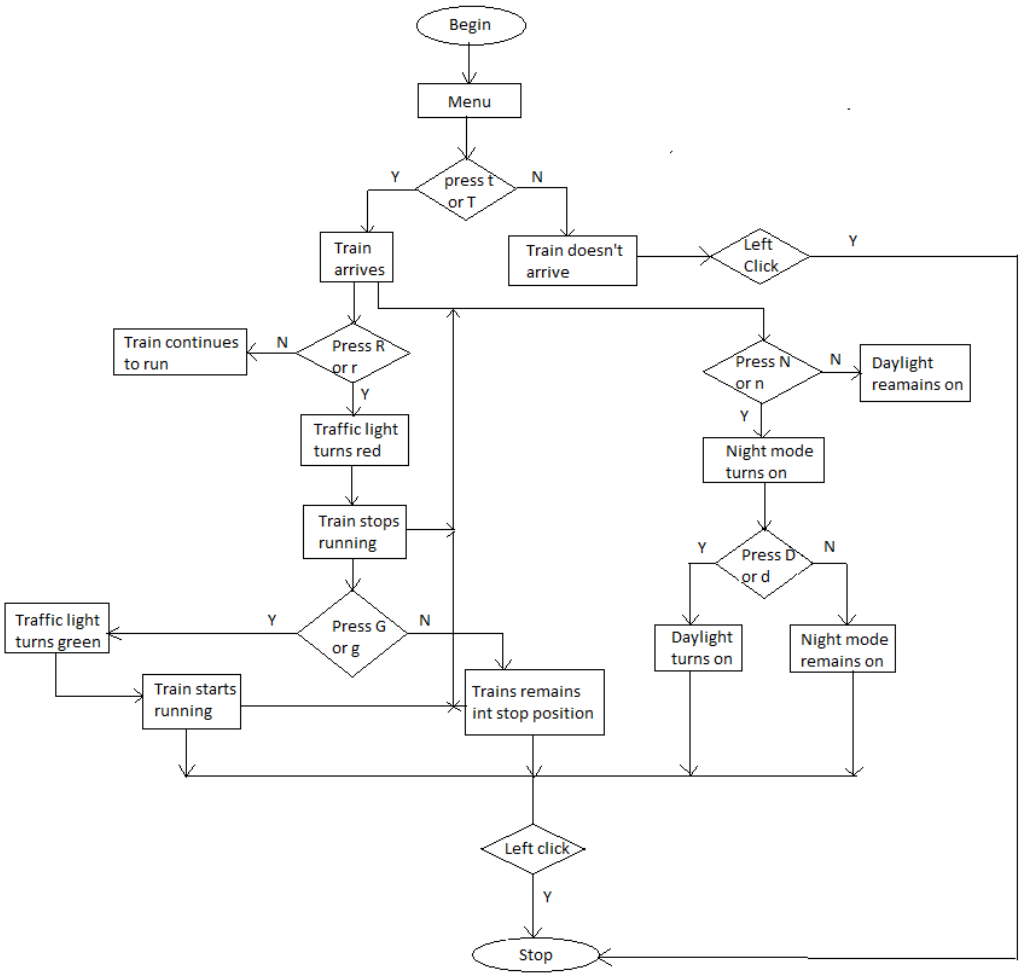
- Main Processor : PENTIUM III
- Processor Speed: 800 MHz
- RAM Size : 128 MB DDR
- Keyboard : Standard qwerty serial or PS/2 keyboard
- Mouse : Standard serial or PS/2 mouse
- Compatibility : AT/T Compatible
- Cache memory : 256 KB
- Diskette drive : 1,44MB,3.5 inches

Software Requirements

- Operating System: Windows XP and Linux (Fedora)
- Hypervisor used : VMware workstation
- Compiler used : g++
- Language used : C++ language
- Editor : gedit Text Editor
- Toolkit : GLUT Toolkit

Chapter -3

DESIGN



3.1 Description of Flow Diagram

The description of the flow diagram is as follows:

Step1 : Start

Srep2 :Menu is displayed

Step3 : User will press t or T for arrival of train and if user doesn't press t or T and presses left click then quits the screen.

Step4 : Either the user can stop the train by pressing r or R or can switch between day and night mode by pressing n or N for night mode and d or D for day mode.

Step5 : If user presses r or R then the traffic light turns red and train stops.
If doesn't press anything the train will keep running.

Step6 : If the train is stopped by r or R button then to again run the train we press g or G.

Step7 : At any point if we press left key we quit the program

Chapter -4

IMPLEMENTATION

Built in Functions

1. glutInit()

glutInit is used to initialize the GLUT library.

Usage: void glutInit (int *argc, char **argv);

Description: glutInit will initialize the GLUT library and negotiate a session with the window system.

2. glutInitDisplayMode()

glutInitDisplayMode sets the initial display mode.

Usage: void glutInitDisplayMode (unsigned int mode);

Mode-Display mode, normally the bitwise OR-ing GLUT display mode bit masks. Description: The initial display mode is used when creating top-level windows, sub-windows, and overlays to determine the OpenGL display mode for the to-be created window oroverlay.

3. glutCreateWindow()

glutCreateWindow creates a top-level window.

Usage: intglutCreateWindow (char *name);

Name-ASCII character string for use as window name

Description: glutCreateWindow creates a top-level window. The name will be provided to the window system as the window's name. The intent is that the window system will label the window with the name. Implicitly, the current window is set to the newly created window.

Each created window has a unique associated OpenGL context.

4.glutDisplayFunc()

glutDisplayFunc sets the display callback for the current window.

Usage: void glutDisplayFunc (void(*func)(void));

Func: The new display callback function.

Description: glutDisplayFunc sets the display callback for the current window. When GLUT determines that the normal plane for the window needs to be redisplayed, the display callback for the window is called. Before the callback, the current window is set to the window needing to be redisplayed and the layer in use is set to the normal plane. The display callback is called with no parameters. The entire normal plane region should be redisplayed in response to the callback.

4. glutMainLoop()

glutMainLoop enters the GLUT event processing loop.

Usage: void glutMainLoop(void);

Description: glutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never return. It will call as necessary any callbacks that have been registered.

5. glMatrixMode()

The two most important matrices are the model-view and projection matrix. At many times, the state includes values for both of these matrices, which are initially set to identity matrices. There is only a single set of functions that can be applied to any type of matrix. Select the matrix to which the operations apply by first set in the matrix mode, a variable that is set to one type of matrix and is also part of the state.

6. glTranslate(GLfloat X, GLfloat Y, GLfloat Z)

glTranslate produces a translation by x y z. If the matrix mode is either

GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after a call to glTranslate are translated.

7. glRotatef(GLdouble angle, GLdouble X, GLdouble Y, GLdouble Z)

glRotatef produces a rotation of angle degrees around the vector x y z. If the matrix mode is either GL_MODEL_VIEW or GL_PROJECTION, all objects drawn after glRotatef is called are rotated. Use glPushMatrix() and glPopmatrix() to save and restore the unrotated coordinate system.

8. glPushMatrix()

There is a stack of matrices for each of the matrix mode. In GL_MODELVIEW mode, the stack depth is atleast 32. In other modes, GL_COLOR, GL_PROJECTION, and GL_TEXTURE, the depth is atleast 2. The current matrix in any mode is the matrix on the top of the stack for that mode.

9. glPopMatrix()

glPopMatrix pops the current matrix stack, replacing the current matrix with the one below it on the stack. Initially, each of the stack contains one matrix, an identity matrix. It is an error to push a full matrix stack or pop a matrix stack that contains only a single matrix. In either case, the error flag is set and no other change is made to GL state.

10. glutSwapBuffers()

Usage: void glutSwapBuffers(void);

Description: Performs a buffer swap on the layer in use for the current window. Specifically, glutSwapBuffers promotes the contents of the front buffer. The contents of the back buffer then become undefined.

11. glPointSize(GLfloat size)

glPointSize specifies the rasterized diameter of points. This value will be used rasterize points. Otherwise, the value written to the shading language built-in

variable gl-PointSize will be used. The point size specified by glPointSize is always returned when GL_POINT_SIZE is queried.

12. glutKeyboardFunc()

Usage: void glutKeyboardFunc(void(*func)(unsigned char key, int x, int y)

Func: The new keyboard callback function

Description: glutKeyboardFunc sets the keyboard callback for the current window. When a user types into the window, each key press generating an ASCII character will generate a keyboard callback. The key callback parameter is the generated ASCII character.

13. glLine Width(GLfloat width)

Parameters: width- Specifies the width of rasterized lines. The initial value is 1. Description: glLineWidth specifies the rasterized width of lines. The actual width is determined by rounding the supplied width to the nearest integer. i pixels are filled in each column that is rasterized, where I is the rounded value of width.

14. glLoadIdentity(void)

glLoadIdentity replaces the current matrix with the identity matrix. It is semantically equivalent to calling glLoadMatrix with the identity matrix.

Pseudocode

```
#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);
```

```
        }
  ….
  …..
  …..
vglColor3f(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);
```

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

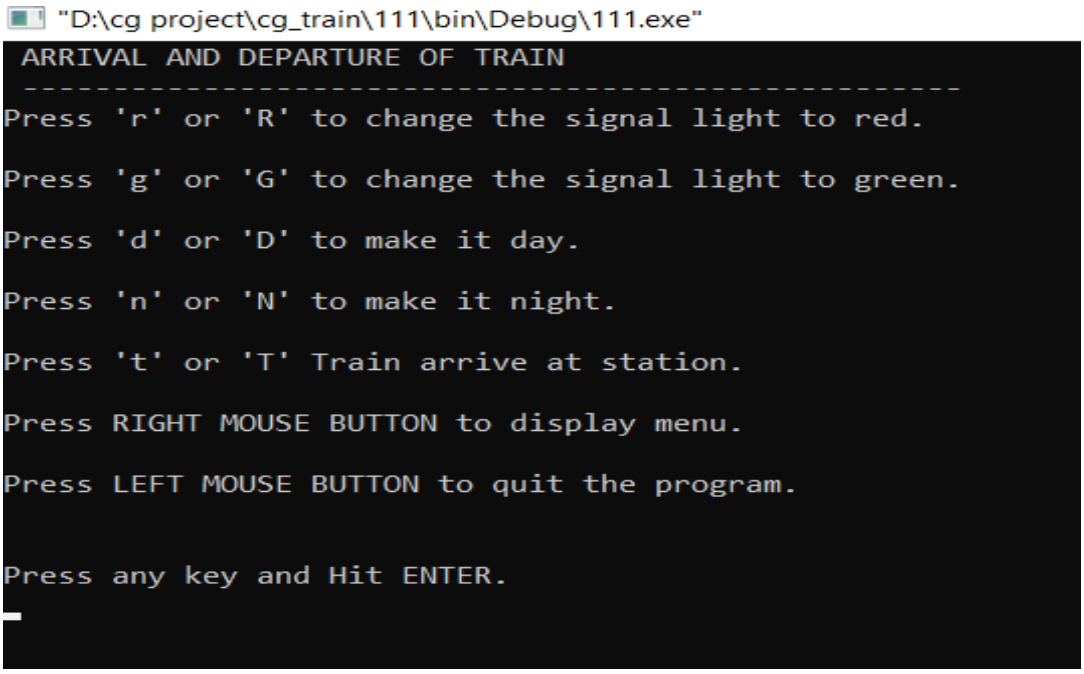
```
traffic_light();
glFlush();
}
int main(int argc,char** argv)
{
int c_menu;
printf(" ARRIVAL AND DEPARTURE OF TRAIN\n ");
printf("..........\n");
printf("Press 'r' or 'R' to change the signal light to red. \n\n");
printf("Press 'g' or 'G' to change the signal light to green. \n\n");
printf("Press '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

SCREENSHOTS

1. Menu Options



2. This is a railway station scene

3. Train stops at the red signal

4. Train moving at night

5. Train can start and stop both in day and night following the traffic signals

6. We can fly an aeroplane too while the train is moving

Chapter -6

CONCLUSION

'Moving Train' mini project is just a theme through this project we have learnt a lot about OpenGL library and its practical implementation .

This project has enabled us to have a deep understanding of computer graphics and through this project we have acquired a deep knowledge about OpenGL library built in functions and their usage in the project.

We thus would like to emphasize the importance of this project to many other perspectives of technical, mathematical,graphical and software concepts which we were unaware of.

Future Enhancements

1.In future the same project can be enhanced in such a way that we can interact more with the project. As of now, we can only change the scene and bring up the hand movement and motion of parents.

2.A vast amount of future work can be possible by following investigations and strategies.

3.More features can be included and can be modified in a more versatile way.

BIBLIOGRAPHY

Reference Books

[1] Donald Hearn & Pauline Baker: Computer Graphics with OpenGL Version, 3rd/4th Edition, Pearson Education, 2011

[2] Edward Angel: Interactive Computer Graphics- A Top Down approach with OpenGL, 5th Edition, Pearson Education

[3] James D Foley, Andries Van Dam, Steven K Feiner, John F Huges Computer Graphics with OpenGL, Pearson Education

[4] Macro Cantu: Mastering Delphi

Websites

[1] https://www.opengl.org/

[2] http://stackoverflow.com/questions/ opengl

[3] https://geeksforgeeks.org/