RV COLLEGE OF ENGINEERING®
BENGALURU – 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

"Traffic Signal"

COMPUTER GRAPHICS LAB (16CS73)

OPEN ENDED EXPERIMENT REPORT

VII SEMESTER

2020-2021

Submitted by

S Meghna	**1RV17CS129**
Sai Swarna DR	**1RV17CS133**

Under the Guidance of

Prof Mamtha T

Department of CSE, R.V.C.E.,
Bengaluru - 560059

RV COLLEGE OF ENGINEERING®, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

Certified that the **Open-Ended Experiment** titled "**Traffic Signal**" has been carried out by **S Meghna(1RV17CS129) and Sai Swarna D R(1RV17CS133),** bonafide students of RV College of Engineering, Bengaluru, have submitted in partial fulfillment for the **Internal Assessment of Course: COMPUTER GRAPHICS LAB (16CS73)** during the year 2020-2021. It is certified that all corrections/suggestions indicated for the internal Assessment have been incorporated in the report.

Prof Mamtha T
Faculty Incharge,
Department of CSE,
R.V.C.E., Bengaluru –59

Dr. Ramakanth Kumar P
Head of Department,
Department of CSE,
R.V.C.E., Bengaluru–59

RV COLLEGE OF ENGINEERING® , BENGALURU - 560059
(Autonomous Institution Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **S Meghna(1RV17CS129) and Sai Swarna D R(1RV17CS133)** the students of Seventh Semester B.E., Computer Science and Engineering, R.V. College of Engineering, Bengaluru hereby declare that the mini-project titled **" Traffic Signal "** has been carried out by us and submitted in partial fulfillment for the **Internal Assessment of Course: COMPUTER GRAPHICS LAB (16CS73) - Open-Ended Experiment** during the year 2020-2021. We do declare that matter embodied in this report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bengaluru **S Meghna**
Date: 18th December 2021 **Sai Swarna DR**

Signature

ABSTRACT

The aim of the TRAFFIC SIGNAL Mini Project is to illustrate the concepts and usage of pre-built functions in OpenGL. Simulation of a traffic signal is being done using computer graphics.The car is built using cubes can be moved using arrow keys. Based on the traffic signal light the user can move the car so that the traffic rules are followed. If the car hits another car then a prompt stating an accident has occurred is shown. In this project, the movement of the car is controlled using a keyboard .

Table of Contents

1 . Introduction

1.1 Computer graphics

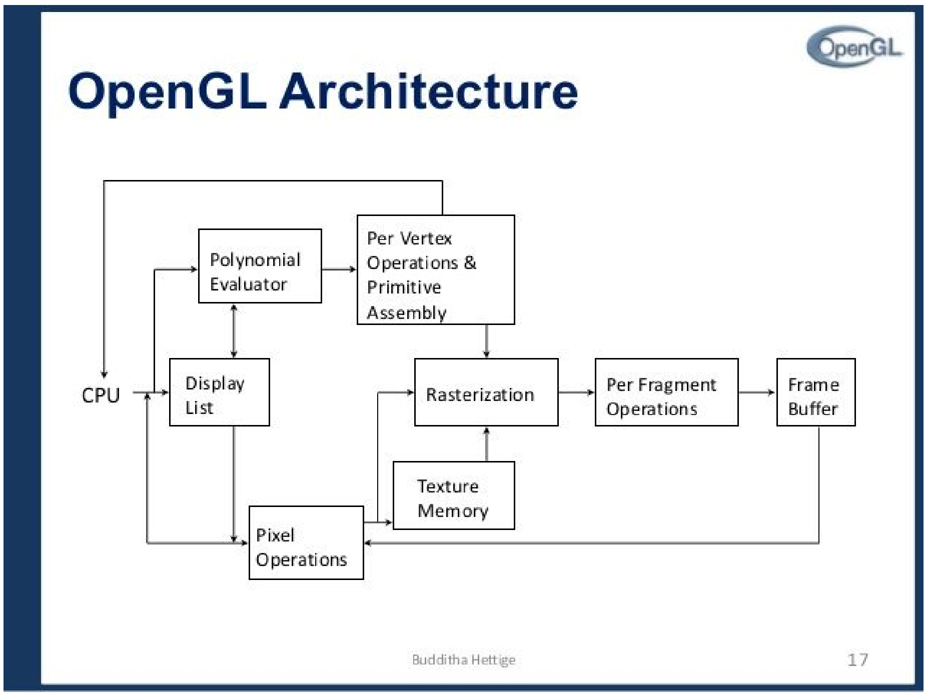
Computer Graphics is the creation of pictures with the help of a computer. The end product of computer graphics is a picture it may be a business graph, drawing, and engineering.In computer graphics, two or three-dimensional pictures can be created that are used for research.

1.2 OpenGl

As a software interface for graphics hardware, OpenGL's main purpose is to render two- and three-dimensional objects into a frame buffer.These objects are described as sequences of vertices or pixels.OpenGL performs several processing steps on this data to convert it to pixels to form the final desired image in the frame buffer.

1.2.1 OpenGL Graphics Architecture

In the given diagram, commands enter from the left and proceed through what can be thought of as a processing pipeline. Some commands specify geometric objects to be drawn, and others control how the objects are handled during the various processing stages.



OpenGL Architecture

As shown by the first block in the diagram, rather than having all commands proceed immediately through the pipeline, you can choose to accumulate some of them in a display list for processing at a later time.Rasterization produces a series of frame buffer addresses and associated values using a two-dimensional description of a point, line segment, or polygon.Each fragment so produced is fed into the last stage per-fragment operations, which performs the final operations on the data before it's stored as pixels in the frame buffer. These operations include conditional updates to the frame buffer based on incoming and previously stored z-value s (for z-buffering) and blending of incoming pixel colors with stored colors, as well as masking and other logical operations on pixel values.All elements of OpenGL state, including the contents of the texture memory and even of the frame buffer, can be obtained by an OpenGL application.

1.2.2 Primitives and Attributes

OpenGL draws primitives—points, line segments, or polygons—subject to several selectable modes.You can control modes independently of each other; that is, setting one mode doesn't affect whether other modes are set .Primitives are specified, modes are set, and other OpenGL operations are described by issuing commands in the form of function calls.

Primitives are defined by a group of one or more vertices. A vertex defines a point, an endpoint of a line, or a corner of a polygon where two edges meet. Data is associated with a vertex, and each vertex and its associated data are processed independently, in order, and in the same way. The type of clipping depends on which primitive the group of vertices represents.

Commands are always processed in the order in which they are received, although there may be an indeterminate delay before a command takes effect. This means that each primitive is drawn completely before any subsequent command takes effect. It also means that state-querying commands return data that's consistent with complete execution of all previously issued OpenGL commands.

1.2.3 Color, Viewing and Control Functions

Color Functions

We use glColor function to set the *foreground color*, and glClearColor function to set the *background* (or *clearing*) color.

- void **glColor3f**(GLfloat *red*, GLfloat *green*, GLfloat *blue*)
- void **glClearColor**(GLclampf *red*, GLclampf *green*, GLclampf *blue*, GLclampf *alpha*)

Viewing and Modeling Transformations

- glMatrixMode(GLenum mode) - sets the matrix to be worked on. Values for mode are:

  GL_MODELVIEW - use the modelview matrix
  GL_PROJECTION - use the projection matrix
  GL_TEXTURE - use the texture matrix

- glScalef(float x, float y, float z) - scale by (x, y, z). Also: glScaled( ).
- glTranslatef(float x, float y, float z) - translate by (x, y, z).
- glRotatef(float angle, float x, float y, float z) - rotate counterclockwise by angle degrees around the line through the origin with direction vector (x, y, z).
- glLoadIdentity(void) - set the current matrix to the identity.

Control Functions

- glutInit(int *argc, char **argv) - initializes GLUT, processes any command-line arguments for X functions.
- glutInitDisplayMode(unsigned int mode) - sets the basic display modes

  GLUT_SINGLE - single buffering
  GLUT_DOUBLE - double buffering
  GLUT_RGB - set to full color mode
  GLUT_RGBA - same as RGB
  GLUT_INDEX - set color index mode
  GLUT_DEPTH - set up a depth buffer

- glutInitWindowSize(int width, int size)- specifies the size, in pixels, of your window.
- glutInitWindowPosition(int x, int y) - specifies the screen location for the upper-left corner of your window.
- int glutCreateWindow(char *string)- creates a top-level window labeled with title.

1.3 Proposed System

1.3.1 Objective of the project

The main objective of this project is to use OpenGl functions to implement a traffic signal scenario. In this project, an accident alert feature has also been implemented.

1.3.2 Methodology

This program includes interaction through the keyboard.

- i -> Student Information
- c -> Keyboard interaction.
- P -> Start the project
- Car Controls using A,W,S,D(left, up, down, right)
- Q-> Quit

1.3.3 Scope

The player first sees a window which has student information and for keyboard interaction on how to play. Once the game is started, using the keyboard the car can be moved. The car has to move by following the traffic signal. If the user tries to move the car when there is Red signal and if it crashes with another car, it shows Accident and the game will end.

2. Requirement Specifications

2.1 Hardware Requirements

- Processor- 3GHz
- Operating System - Windows Operating System
- RAM - 4GB
- Hard Disk - minimum 18GB, recommended 40GB

2.2 Software Requirements

- Operating System- Windows 10/MacOS
- Tool - Visual Studio
- Language- C++
- Library- OpenGL(glut 3.7.6)

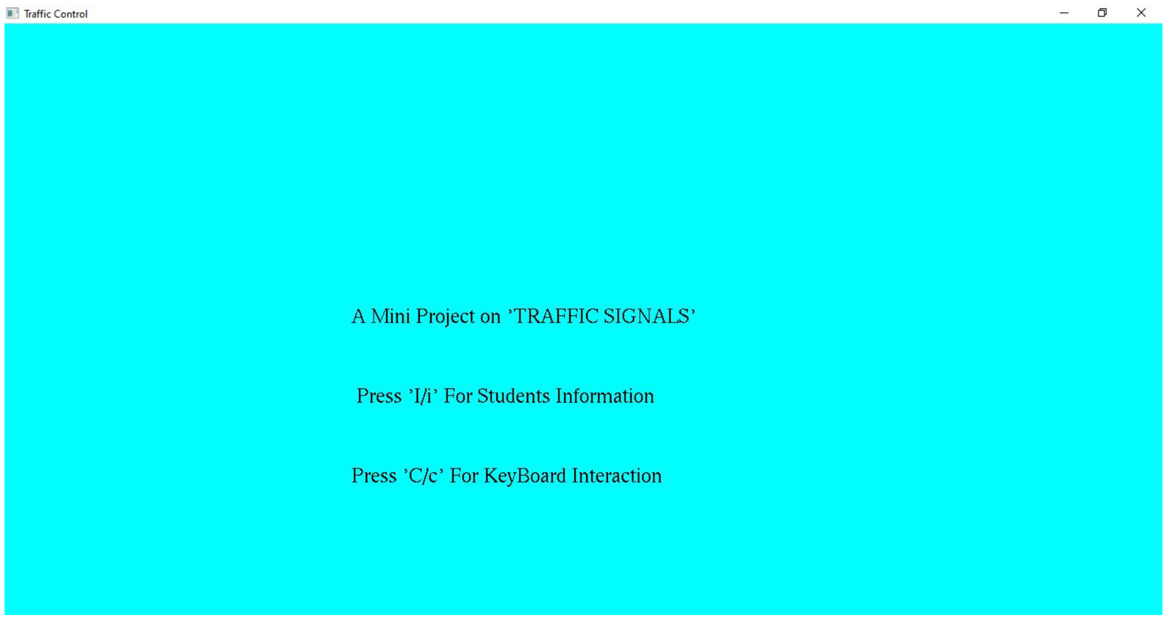
3. Implementation

Traffic signal project is implemented using the following gl functions:

- ➢ glutInit() : interaction between the windowing system and OPENGL is initiated.
- ➢ glutInitDisplayMode() : used when double buffering is required and depth information is required.
- ➢ glutCreateWindow() : this opens the OPENGL window and displays the title at top of the window.
- ➢ glutInitWindowSize() : specifies the size of the window.
- ➢ glutInitWindowPosition() : specifies the position of the window in screen coordinates.
- ➢ glutKeyboardFunc() : handles normal ascii symbol.
- ➢ glutSpecialFunc() : handles special keyboard keys.
- ➢ glutReshapeFunc() : sets up the callback function for reshaping the window.
- ➢ glutIdleFunc() : this handles the processing of the background.
- ➢ glutDisplayFunc() : this handles redrawing of the window.
- ➢ glutMainLoop() : this starts the main loop, it never returns.
- ➢ glViewport() : used to set up the viewport.
- ➢ glVertex3fv() : used to set up the points or vertices in three dimensions.
- ➢ glColor3fv() : used to render color to faces.
- ➢ glFlush() : used to flush the pipeline.
- ➢ glutPostRedisplay() : used to trigger an automatic redrawal of the object.
- ➢ glMatrixMode() : used to set up the required mode of the matrix.
- ➢ glLoadIdentity() : used to load or initialize to the identity matrix.
- ➢ glTranslatef() : used to translate or move the rotation centre from one point to another in three dimensions.
- ➢ glRotatef() : used to rotate an object through a specified rotation angle.

4. Results and Snapshots

4.1 Start Window Screen

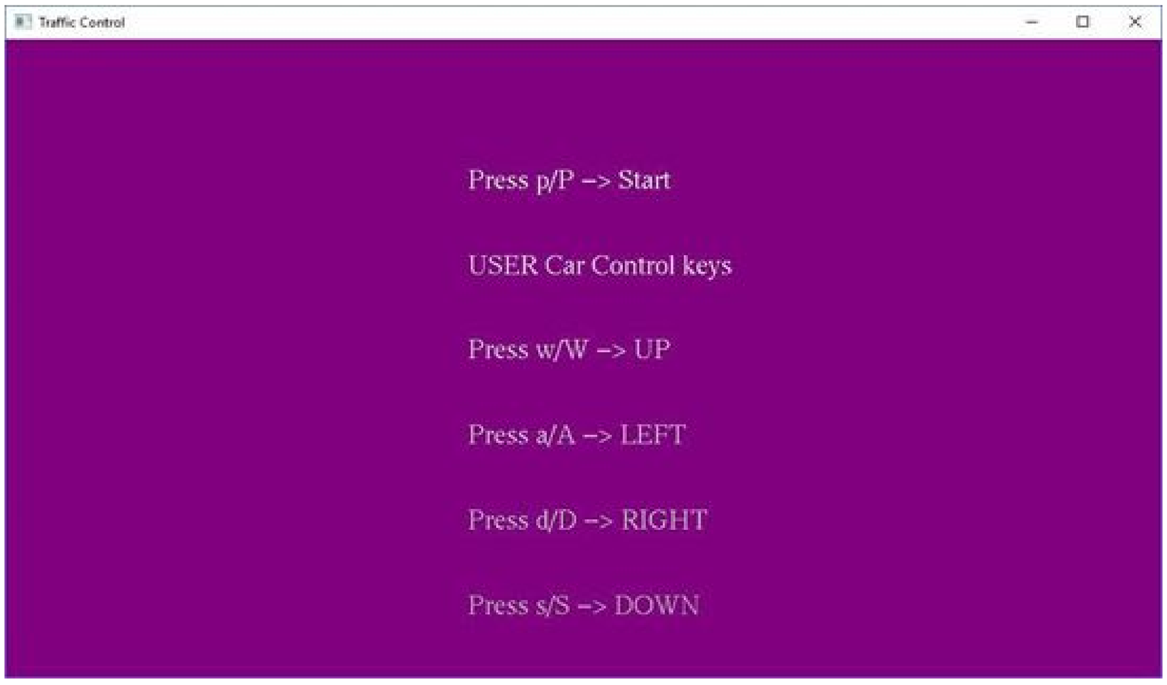

Traffic Control

A Mini Project on ’TRAFFIC SIGNALS’

Press ’I/i’ For Students Information

Press ’C/c’ For KeyBoard Interaction

4.2 Keyboard interface Window


Traffic Control

Press p/P –> Start

USER Car Control keys

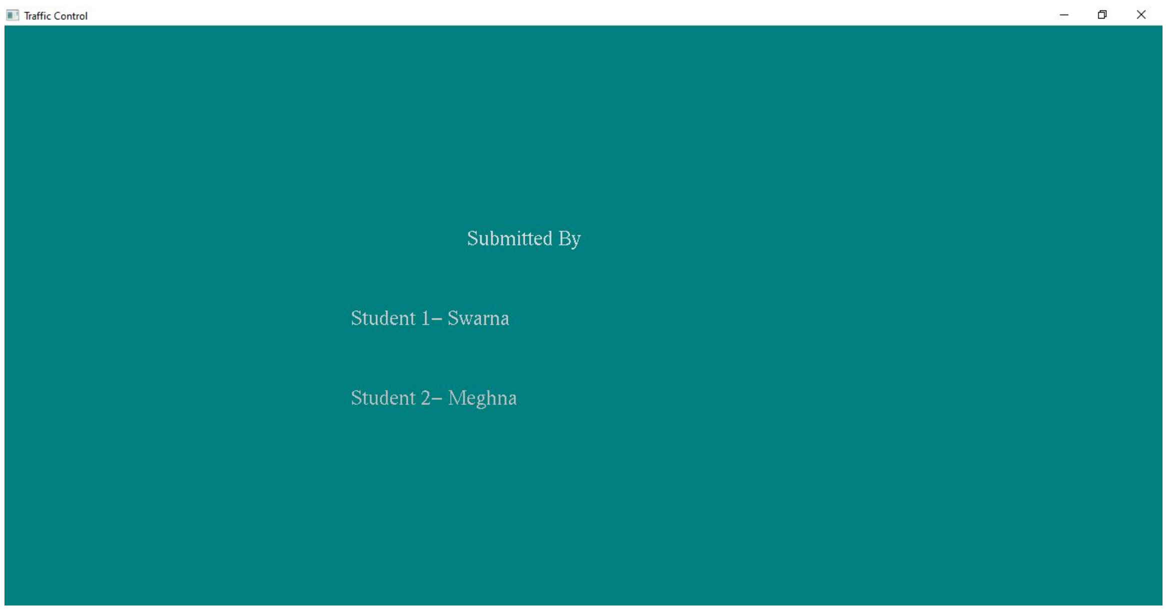
Press w/W –> UP

Press a/A –> LEFT

Press d/D –> RIGHT

Press s/S –> DOWN

4.3 Students information Window

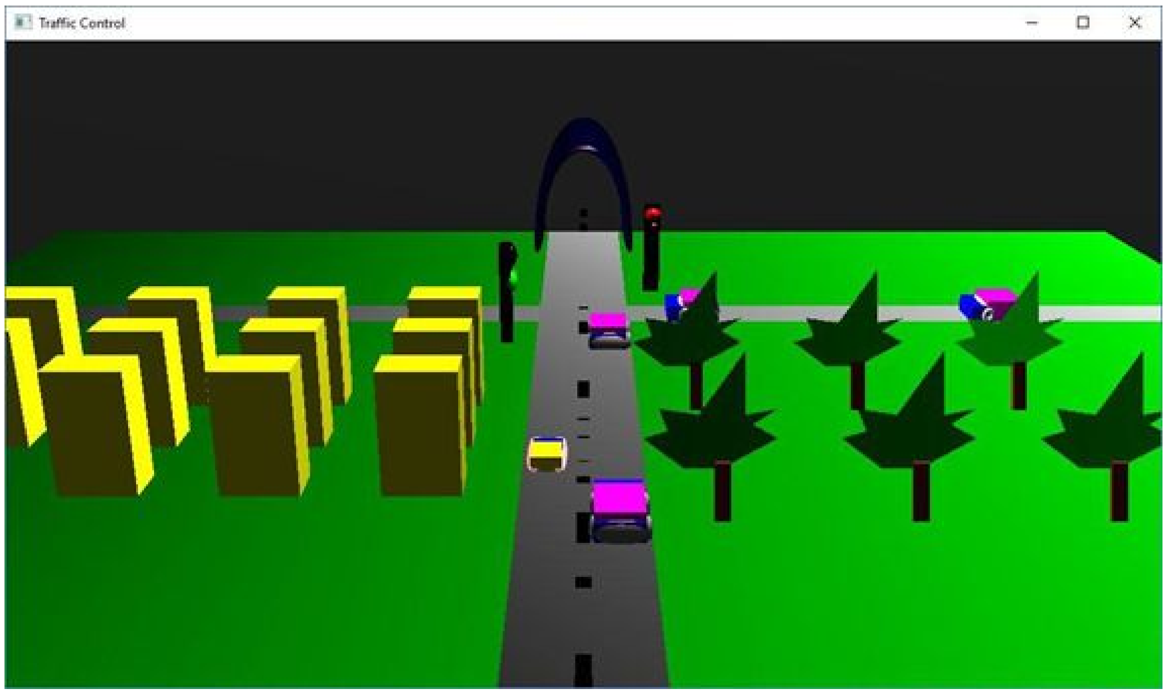


Submitted By

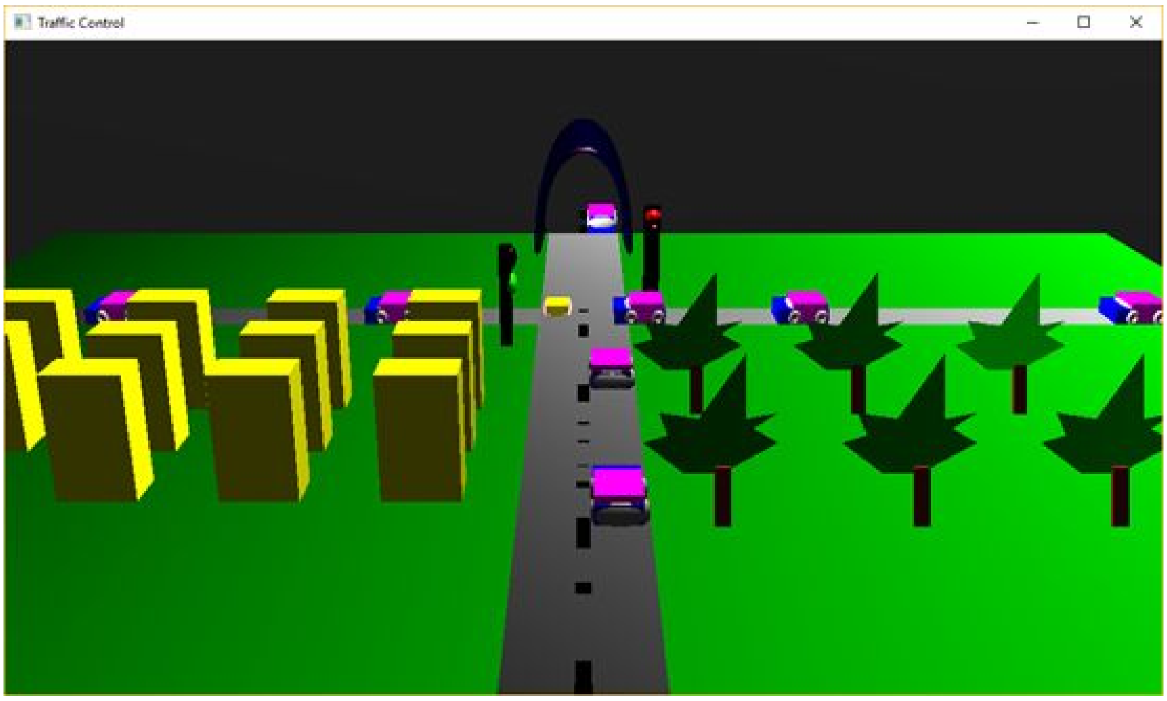
Student 1– Swarna

Student 2– Meghna

4.4 Game Running window



4.4 Game Running window(car passing the signal)



4.5 Accident Alert

Accident

Press r –> Reset

5. Conclusion

The project “Traffic Signal” clearly demonstrates the simulation of traffic signal using OpenGL. OpenGL supports enormous flexibility in the design and the use of Opengl graphics program. The built in functions and methods covers many functionality and reduce the job of coding as well as makes the implementation simpler.

Finally we conclude that this program clearly illustrates the traffic signal using openGL and has been completed successfully and demonstrated.

6. Bibliography

1. http://jerome.jouvie.free.fr/OpenGl/Lessons/Lesson3.php
2. https://www3.ntu.edu.sg/home/ehchua/programming/opengl/cg_introduction.html
3. Interactive Computer Graphics: A Top-Down Approach with Shader-Based OpenGL, 6th Edition
4. Computer Graphics: Principles and Practice by James D. Foley, Andries van Dam, Steven K. Feiner, John Hughes, Morgan McGuire, David F. Sklar, and Kurt Akeley.
5. https://www.glprogramming.com/red/chapter01.html

APPENDIX A- SOURCE CODE

```
#include <windows.h>
#include<string.h>
#include<stdarg.h>
#include<stdio.h>
#include <GL/glut.h>
static double carx = -1.2, cary = -4;
static double cx[5] = { 6,15,22,30,40}, cy[5] = { -1,-10,-24,-30,-40 }, x[10] = { 0 };
static double s1 = 0;
static bool stop = false, red = true, green = false, crash = false;
void* font;
void* currentfont;
void setFont(void* font)
{
currentfont = font;
}
void drawstring(float x, float y, float z, const char* string)
{
        const char* c;
          glRasterPos3f(x, y, z);
          for (c = string;*c != '\0';c++)
          {
                  glColor3f(0.0, 1.0, 1.0);
                  glutBitmapCharacter(currentfont, *c);
          }
```

```
}
void tree()
{
        glPushMatrix();
        glColor3f(0, 0.2, 0);
        glBegin(GL_POLYGON);
        glVertex2f(0, 0);
        glVertex2f(-.5, 0.5);
        glVertex2f(0, 0.7);
        glVertex2f(-0.3, 1);
        glVertex2f(0.2, 0.9);
        glVertex2f(0.8, 2);
        glVertex2f(0.8, 0.9);
        glVertex2f(1.2, 1);
        glVertex2f(0.8, 0.8);
        glVertex2f(1.2, 0.5);
        glVertex2f(0.5, 0);
        glEnd();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(0.5, -0.4, 0);
        glScaled(0.2, 1, 0.1);
        glColor3f(0.5, 0.1, 0.1);
        glutSolidCube(1.0);
        glPopMatrix();

}
void road()
{
        glColor3f(0, 1, 0);
        glBegin(GL_POLYGON);
        glVertex2f(-6, 6);
        glVertex2f(6, 6);
        glVertex2f(6, -6);
        glVertex2f(-6, -6);
        glEnd();
        glColor3f(0.4, 0.4, 0.4);
        glBegin(GL_POLYGON);
        glVertex2f(-6, 2.5);
        glVertex2f(6, 2.5);
        glVertex2f(6, 1.5);
        glVertex2f(-6, 1.5);
        glEnd();
        glColor3f(0, 0, 0);
        for (float x1 = 0;x1 <= 100;x1 += 0.4) {
                glBegin(GL_POLYGON);
                glVertex2f(-6 + x1, 2);
                glVertex2f(-5.8 + x1, 2);
                glVertex2f(-5.8 + x1, 1.9);
                glVertex2f(-6 + x1, 1.9);
                glEnd();
        }
        glColor3f(0.4, 0.4, 0.4);
        glBegin(GL_POLYGON);
        glVertex2f(0.4, 6);
        glVertex2f(-0.4, 6);
        glVertex2f(-0.4, -6);
        glVertex2f(0.4, -6);
        glEnd();
        glColor3f(0, 0, 0);
```

```
	for (float y1 = 0;y1 <= 100;y1 += 1) {
		glBegin(GL_QUADS);
		glVertex2f(0.04, -6 + y1);
		glVertex2f(0.04, -5.5 + y1);
		glVertex2f(-0.04, -5.5 + y1);
		glVertex2f(-0.04, -6 + y1);
		glEnd();
	}
}
void bdg(float x1, float y1)
{
	glPushMatrix();
	glTranslatef(x1, y1, 0);
	glScaled(0.5, 0.4, 2.2);
	glutSolidCube(1);
	glPopMatrix();
}
void cars(float x1, float y1)
{
	glPushMatrix();
	glScaled(0.3, 0.5, 0.5);
	glTranslatef(x1, y1, 0);
	glPushMatrix();
	glColor3f(1, 0, 1);
	glutSolidCube(1);
	glPopMatrix();
	glPushMatrix();
	glRotatef(90, 1.0f, 0.0f, 0.0f);
	glTranslatef(-0.45, 0.15, -0.02);
	glScaled(0.15, 0.15, 0.6);
	glColor3f(.5, .5, .5);
	glutSolidTorus(1, 0.8, 20, 20);
	glPopMatrix();
	glPushMatrix();
	glRotatef(90, 1.0f, 0.0f, 0.0f);
	glTranslatef(0.35, 0.15, -0.02);
	glScaled(0.15, 0.15, 0.6);
	glColor3f(.5, .5, .5);
	glutSolidTorus(1, 0.8, 20, 20);
	glPopMatrix();
	glPushMatrix();
	glTranslatef(-0.2, 0, 0);
	glScaled(1.5, 1, .7);
	glColor3f(0, 0, 1);
	glutSolidCube(1);
	glPopMatrix();
	glPopMatrix();
}
void carsv(float x1, float y1)
{
	glPushMatrix();
	glScaled(0.3, 0.5, 0.5);
	glTranslatef(x1, y1, 0);
	glRotatef(-90, 0, 0, 1);
	glPushMatrix();
	glColor3f(1, 0, 1);
	glutSolidCube(1);
	glPopMatrix();
	glPushMatrix();
	glRotatef(90, 1.0f, 0.0f, 0.0f);
```

```
        glTranslatef(-0.45, 0.15, -0.02);
        glScaled(0.15, 0.15, 0.6);
        glColor3f(.5, .5, .5);
        glutSolidTorus(1, 0.8, 20, 20);
        glPopMatrix();
        glPushMatrix();
        glRotatef(90, 1.0f, 0.0f, 0.0f);
        glTranslatef(0.35, 0.15, -0.02);
        glScaled(0.15, 0.15, 0.6);
        glColor3f(.5, .5, .5);
        glutSolidTorus(1, 0.8, 20, 20);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(-0.2, 0, 0);
        glScaled(1.5, 1, .7);
        glColor3f(0, 0, 1);
        glutSolidCube(1);
        glPopMatrix();
        glPopMatrix();
}

void myCar(float x1, float y1)
{
        glPushMatrix();
        glScaled(0.2, 0.6, 0.1);
        glTranslatef(x1, y1, 0);
        glPushMatrix();
        glColor3f(0, 0, 1);
        glutSolidCube(1);
        glPopMatrix();
        glPushMatrix();
        glRotatef(90, 1.0f, 0.0f, 0.0f);
        glTranslatef(-0.4, 0.15, -0.02);
        glScaled(0.15, 0.15, 0.6);
        glColor3f(1.5, .5, .5);
        glutSolidTorus(1, 0.8, 20, 20);
        glPopMatrix();
        glPushMatrix();
        glRotatef(90, 1.0f, 0.0f, 0.0f);
        glTranslatef(0.4, 0.15, -0.02);
        glScaled(0.15, 0.15, 0.6);
        glColor3f(1.5, .5, .5);
        glutSolidTorus(1, 0.8, 20, 20);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(0, -.25, 0);
        glScaled(1, 0.6, 2.7);
        glColor3f(1, 1, 0);
        glutSolidCube(1);
        glPopMatrix();
        glPopMatrix();
}
void trafic(double ang)
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glClearColor(1.0, 1.0, 1.0, 0);
        glLoadIdentity();
        glTranslatef(0.0f, 0.0f, -13.0f);
        glRotatef(-70, 1, 0, 0);
        glPushMatrix();
```

```
road();
glPopMatrix();
float tempy[5] = { cy[0] + x[5] / 15,cy[1] + x[6] / 15,cy[2] + x[7] / 15,cy[3] + x[8] / 15,cy[4] + x[9] / 15 };
for (int ty = 0;ty <= 4;ty++) {
        if (tempy[4] >= 8) {
                for (int i = 0;i <= 9;i++)
                        x[i] = 0;
        }

        if (tempy[ty] >= 1.5 && tempy[ty] <= 2 && red)
        {
                //do nothing
        }
        else if (ty == 0)
        {
                x[ty + 5] += 0.07;
        }
        else if (tempy[ty] >= tempy[ty - 1] - 2)
        {
        }
        else
                x[ty + 5] += 0.07;
        carsv(0.7, tempy[ty]);
}
float tempx[5] = { cx[0] - x[0] / 15,cx[1] - x[1] / 15,cx[2] - x[2] / 15,cx[3] - x[3] / 15,cx[4] - x[4] / 15 };
for (int tx = 0;tx <= 4;tx++)
{
        if (tempx[tx] >= 1.8 && tempx[tx] <= 3 && green)
        {

        }
        else if (tx == 0)
        {
                x[tx] += 0.07;
        }
        else if (tempx[tx] <= tempx[tx - 1] + 2)
        {
        }
        else
                x[tx] += 0.07;

        cars(tempx[tx], 3.5);
}
for (int ty1 = 0;ty1 <= 4;ty1++)
{
        if (carx >= 0 && carx <= 1 && cary >= tempy[ty1] - 0.4 && cary <= tempy[ty1] + 0.4)
        {
                crash = true;
        }
}
for (int ty2 = 0;ty2 <= 4;ty2++)
{
        if (cary >= 3.0 && cary <= 3.8 && carx >= tempx[ty2] - 0.4 && carx <= tempx[ty2] + 0.4)
        {
                crash = true;
        }
}
myCar(carx, cary);
for (int i = -4; i < 0; i++)
        for (int j = -3; j < 0; j++)
```

```
		{
			bdg(i, j);
		}
	glPushMatrix();
	glTranslatef(0, 8, 0);
	glColor3f(0.1, 0.1, 0.1);
	glScaled(30, 0.5, 13);
	glutSolidCube(1);
	glPopMatrix();
	glPushMatrix();
	glTranslatef(0, 6, 0);
	glRotatef(90, 1, 0, 0);
	glScaled(0.5, 1.5, 13);
	glColor3f(0, 0, 0.1);
	glutSolidTorus(0.1, 1, 30, 30);
	glPopMatrix();
	glPushMatrix();
	glTranslatef(0.65, 2.8, 0);
	glScaled(0.3, 0.2, 4.5);
	glColor3f(0, 0, 0);
	glutSolidCube(0.5);
	glPopMatrix();
	glPushMatrix();
	if (red)
		glColor3f(1, 0, 0);
	else
		glColor3f(0, 0, 0);
	glTranslatef(0.65, 2.6, 1.05);
	glScaled(0.2, 0.2, 0.2);
	glutSolidSphere(0.35, 30, 30);
	glPopMatrix();
	glPushMatrix();
	if (green)
		glColor3f(0, 1, 0);
	else
		glColor3f(0, 0, 0);
	glTranslatef(0.65, 2.5, 0.9);
	glScaled(0.2, 0.2, 0.2);
	glutSolidSphere(0.35, 30, 30);
	glPopMatrix();
	glPushMatrix();
	glTranslatef(-0.65, 0.8, 0);
	glScaled(0.2, 0.4, 4.5);
	glutSolidCube(0.5);
	glPopMatrix();

	//red signal
	glPushMatrix();
	if (red)
		glColor3f(0, 1, 0);
	else
		glColor3f(0, 0, 0);
	glTranslatef(-0.6, 0.8, 0.7);
	glScaled(0.2, 0.5, 0.5);
	glutSolidSphere(0.25, 30, 30);
	glPopMatrix();
	glPushMatrix();
	if (green)
		glColor3f(1, 0, 0);
	else
```

```
                glColor3f(0, 0, 0);
        glTranslatef(-0.6, 0.8, 1);
        glScaled(0.2, 0.5, 0.5);
        glutSolidSphere(0.25, 30, 30);
        glPopMatrix();
        glPushMatrix();
        glTranslatef(2.5, -2.8, 1);
        glRotatef(70, 1, 0, 0);
        glScaled(0.4, 0.4, 0.4);
        tree();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(1.5, -2.8, 1);
        glRotatef(70, 1, 0, 0);
        glScaled(0.4, 0.4, 0.4);
        tree();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(0.5, -2.8, 1);
        glRotatef(70, 1, 0, 0);
        glScaled(0.4, 0.4, 0.4);
        tree();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(2.5, -4.8, 1);
        glRotatef(70, 1, 0, 0);
        glScaled(0.4, 0.4, 0.4);
        tree();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(1.5, -4.8, 1);
        glRotatef(70, 1, 0, 0);
        glScaled(0.4, 0.4, 0.4);
        tree();
        glPopMatrix();
        glPushMatrix();
        glTranslatef(0.5, -4.8, 1);
        glRotatef(70, 1, 0, 0);
        glScaled(0.4, 0.4, 0.4);
        tree();
        glPopMatrix();
        glFlush();
        glutSwapBuffers();
}
void i()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glClearColor(0.0, 0.5, 0.5, 0);
        glLoadIdentity();
        glTranslatef(0.0f, 0.0f, -13.0f);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, 1, -1.0, "Submitted By");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-2, 0, -1.0, "Student 1- Swarna");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-2, -1, -1.0, "Student 2- Meghna");
        glEnable(GL_COLOR_MATERIAL);
        glFlush();
        glutSwapBuffers();
}
void c()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glClearColor(0.5, 0.0, 0.5, 0);
        glLoadIdentity();
        glTranslatef(0.0f, 0.0f, -13.0f);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, 2, -1.0, "Press p/P -> Start");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, 1, -1.0, "USER Car Control keys");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, 0, -1.0, "Press w/W -> UP");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, -1, -1.0, "Press a/A -> LEFT");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, -2, -1.0, "Press d/D -> RIGHT");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, -3, -1.0, "Press s/S -> DOWN");
        glEnable(GL_COLOR_MATERIAL);
        glFlush();
        glutSwapBuffers();
}
void gameOver() {
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glClearColor(0, 0, 0, 0);
        glLoadIdentity();
        glTranslatef(0.0f, 0.0f, -13.0f);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, 0, -1.0, "Accident");

        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(1, 1, 1);
        drawstring(-1, -1, -1.0, "Press r -> Reset");
        glFlush();
        glutSwapBuffers();
}
void p()
{
        if (s1 <= 10) {
                if (s1 < 5) {
                        red = true;
                        green = false;
                        s1 += 0.006;
                }
                if (s1 > 5) {
                        red = false;
                        green = true;
                        s1 += 0.006;
                }
                if (s1 >= 9.5) {
                    s1 = 0;
                }
        }
        if (crash)
        {
                gameOver();
        }
        else
        {
                trafic(x[0]);
        }
}
void P()
{
        trafic(x[0]);
}
void doInit()
{       glClearColor(1.0, 1.0, 1.0, 0);
        glViewport(0, 0, 640, 480);
        glMatrixMode(GL_PROJECTION);
        glLoadIdentity();
        gluPerspective(30.0f, (GLfloat)640 / (GLfloat)480, 0.1f, 200.0f);
        glMatrixMode(GL_MODELVIEW);
        glLoadIdentity();
        glClearDepth(2.0f);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_COLOR_MATERIAL);
        glDepthFunc(GL_LEQUAL);
}
void display()
{
        glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
        glClearColor(0, 1, 1, 1);
        glLoadIdentity();
        glTranslatef(0.0f, 0.0f, -13.0f);
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(0, 0, 0);
        drawstring(-2, 0, -1.0, "A Mini Project on 'TRAFFIC SIGNALS'");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(0, 0, 0);
        drawstring(-2, -1, -1.0, " Press 'I/i' For Students Information");
        setFont(GLUT_BITMAP_TIMES_ROMAN_24);
        glColor3f(0, 0, 0);
        drawstring(-2, -2, -1.0, "Press 'C/c' For KeyBoard Interaction");
        GLfloat mat_ambient[] = { 0.0f,1.0f,2.0f,1.0f };
        GLfloat mat_diffuse[] = { 0.0f,1.5f,.5f,1.0f };
        GLfloat mat_specular[] = { 5.0f,1.0f,1.0f,1.0f };
        GLfloat mat_shininess[] = { 50.0f };
        glMaterialfv(GL_FRONT, GL_AMBIENT, mat_ambient);
        glMaterialfv(GL_FRONT, GL_DIFFUSE, mat_diffuse);
        glMaterialfv(GL_FRONT, GL_SPECULAR, mat_specular);
        glMaterialfv(GL_FRONT, GL_SHININESS, mat_shininess);
        GLfloat lightIntensity[] = { 1.7f,1.7f,1.7f,1.0f };
        GLfloat light_position3[] = { 2.0f,10.0f,3.0f,1.0f };
        glLightfv(GL_LIGHT0, GL_POSITION, light_position3);
        glLightfv(GL_LIGHT0, GL_DIFFUSE, lightIntensity);
        GLfloat lightIntensity1[] = { 1.7f,1.7f,1.7f,1.0f };
        GLfloat light_position1[] = { -2.0f,1.0f,-5.0f,1.0f };
        glLightfv(GL_LIGHT1, GL_POSITION, light_position1);
        glLightfv(GL_LIGHT1, GL_DIFFUSE, lightIntensity1);
        glEnable(GL_COLOR_MATERIAL);
        glFlush();
        glutSwapBuffers();
}
void menu(int id)
{
        switch (id)
        {
        case 1:glutIdleFunc(P);
                break;
        case 2:glutIdleFunc(p);
                break;
        case 5:exit(0);
                break;
        }
        glFlush();
        glutSwapBuffers();
        glutPostRedisplay();
}
void mykey(unsigned char key, int x, int y)
{
        if (key == 'p')
        {
                glutIdleFunc(p);
        }
        if (key == 'P')
        {
                glutIdleFunc(P);
        }
        if (key == 'w' || key == 'W')
        {
                cary += 0.08;
                glutIdleFunc(p);
        }
        if (key == 'a' || key == 'A')
        {
                if (carx >= -1.5)
                        carx -= 0.1;
                glutIdleFunc(p);
        }
        if (key == 'd' || key == 'D')
        {
                if (carx <= 1.5)
                        carx += 0.1;
                glutIdleFunc(p);
        }
        if (key == 's' || key == 'S')
        {
                cary -= 0.08;
                glutIdleFunc(P);
        }
        if (key == '1')
        {
                red = true;
                green = false;
                glutIdleFunc(p);
        }
        if (key == '2')
        {
                red = false;
```

```
            green = true;
            glutIdleFunc(p);
        }
        if (key == 'r')
        {
            red = false;
            green = true;
            crash = false;
            glClearColor(1, 1, 1, 1);
            glutIdleFunc(p);
        }
        if (key == 'i')
        {
            glutIdleFunc(i);
        }
        if (key == 'c')
        {
            glutIdleFunc(c);
        }
        if (key == 'q' || key == 'Q')
        {
            exit(0);
        }
}
int main(int argc, char* argv[])
{
        glutInit(&argc, argv);
        glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGB);
        glutInitWindowSize(1000, 480);
        glutInitWindowPosition(0, 0);
        glutCreateWindow("Traffic Control");
        glutDisplayFunc(display);
        glEnable(GL_LIGHTING);
        glEnable(GL_LIGHT0);
        glEnable(GL_LIGHT1);
        glShadeModel(GL_SMOOTH);
        glEnable(GL_DEPTH_TEST);
        glEnable(GL_NORMALIZE);
        glutKeyboardFunc(mykey);
        glutCreateMenu(menu);
        glutAddMenuEntry("Start       'p'", 1);
        glutAddMenuEntry("Quit           'q'", 5);
        glutAttachMenu(GLUT_RIGHT_BUTTON);
        doInit();
        glutMainLoop();
        return 0;
}
```