Computer Graphics (UCS505)

Traffic Light Simulation

Submitted By

AAYUSHI PURI 102103676

NISHIL CHAUDHARY 102103680

SANSKAR SUREKA 102103688

3CO24

B.E. Third Year – COE

Submitted To:

MS. RIYA SHARMA

THAPAR INSTITUTE
OF ENGINEERING & TECHNOLOGY
(Deemed to be University)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

S. No.	Description	Page No.
1.	Introduction to Project	3
2.	Computer Graphics concepts used	4
3.	User Defined Functions used	5
4.	Code	7
5.	Screenshots	28

Introduction to Project

Graphics provides one of the most natural means of communicating with a computer, since our highly developed 2D and 3D pattern recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. It has the added advantage that, with the computer, we can make pictures not only of concrete realworld objects but also of abstract, synthetic objects, such as mathematical surfaces and of data that have no inherent geometry, such as survey results.

OpenGL (Open Graphics Library) is a standard specification defining a cross language cross platform API for writing applications that produce 2D and 3D computer graphics. OpenGL is widely used in CAD, virtual reality, scientific visualization, information visualization and flight simulation.

The project, Traffic Light Simulation, is essentially a manifestation of the computer graphics technology and uses the Open Graphics library. It is a simulation of traffic control, an event which is common in the day to day lives of people traveling through the roads. The project demonstrates the regulation of traffic through the different traffic signals.

There are three traffic lights or signals, **Red**, **Green** and **Yellow**. **Red** implies stop, **Green** means go, **Yellow** means ready to go.
The same application has been showcased in the project. When its a green signal it means the vehicle can move or run on the road but if this green signal is turned off and the red signal is turned on then it indicates the driver to stop his vehicle. The simulation can be performed both in the night and day themes.

All components in the project have been built using the OpenGL functionalities including all kinds of shapes and backgrounds. The aim of this project was to seek the opportunity to learn the different concepts of the subject and apply them practically using relevant technology.

Computer Graphics concepts used

Computer graphics is one of the most exciting and rapidly growing computer fields. It is also an extremely effective medium for communication between man and computer;a humanbeing can understand the information content of a displayed diagram or perspective view much faster than he can understand a table of numbers or text containing the same information.

OpenGL is an application program interface (API) offering various functions to implement primitives, models, and images. This offers functions to create and manipulate render lighting, coloring, viewing the models. OpenGL offers different coordinate systems and frames. OpenGL offers translation, rotation, and scaling of objects.

1. **glVertex2f(x,y,z):** It specifies the x and y coordinates of the vertex, and the z coordinate is set to zero.

2. **glBegin(Polygon_Type):** The glBegin subroutine accepts a single argument that specifies which of 10 ways the vertices will be interpreted.

3. **glEnd():** This is a inbuilt function in opengl library to end the polygon structure start with glBegin().

4. **glMatrixMode(GL_PROJECTION):** Specifies which matrix stack is the target for subsequent matrix operations. Three values are accepted: GL_MODELVIEW , GL_PROJECTION.

5. **glClearColor():** It sets the color value to the buffer.

6. **drawstring("string"):** This command is use to write string on screen.

7. **glutInitWindowSize(x_size,y_size):** This command is use to intialize window Screen.

8. **glutInitWindowPosition(x,y):** This Command is used to intialize window at specified position.

User-Defined Functions

1. **void draw_pixel(GLint, GLint):** This is a helper function of the function for drawing a circle. This is used to draw a point/pixel on the screen whose coordinates it receives as its arguments.

2. **void plotpixels(GLint, GLint, GLint, GLint):** This is a helper function of the function for drawing a circle. This function is used to take advantage of the eight-way symmetry property of the circle. It draws a point using the coordinates it receives as arguments and then draws its reflections in the remaining octants.

3. **void draw_circle(GLint, GLint, GLint)**: This function is used to draw a circle on the screen with the help of its helper functions. It receives the coordinates of the center of the circle and the radius of the circle as its arguments.

4. **void draw_object():** This is the function that draws all the objects visible on the screen. All elements such as the buildings, windows, doors, road, divider, cars, traffic light, grass, sky, sun/moon, and cloud/stars are drawn by this function.

5. **void signal():** This function draws the red or green light on the traffic signal depending on user input.

6. **void keyboard(char, int, int):** This function receives the user input as its argument i.e. the key pressed by the user, and correspondingly controls whether the signal is to be red or green.

7. **void idle():** This function controls whether the cars are to be moving or stationary depending on whether the signal is red or green.

8. **void main_menu(int):** This function receives the user input as its argument i.e. the option selected by the user in the dropdown menu appearing on right click of their mouse, and controls, whether the sky to be displayed, is for night or day time.

9. **void drawstring(int, int, char*):** This function is used to display strings/sentences on the screen. It receives the string to displayed and its starting position's coordinates as its arguments.

10. **void frontscreen():** This function is used to display the information regarding the project and is the first screen that shows up when the project is started.

Code:

```
#include<stdlib.h>
#include<string>
#include<string.h>
#include<stdio.h>
#include<GL/glut.h>
#define SPEED 5       //speed of traffic


float x = 0.0;        //movement of car
int light = 1;        //1 for green-light, 0 for red-light
int day = 1;        //1 for day, 0 for night
int d, flag = 0;
int front_car2 = 0;
int front_car1 = front_car2 - 500;


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()
{

if (day == 1)
{
glColor3f(0.15, 0.86, 0.88);
glBegin(GL_POLYGON);
glVertex2f(0, 400);
glVertex2f(0, 700);
glVertex2f(1000, 700);
glVertex2f(1000, 400);
glEnd();



//sun
for (d = 0; d <= 30; d++)
{
glColor3f(0.96, 0.27, 0.13);
draw_circle(875, 625, d);
}



//cloud 1
for (d = 0; d <= 25; d++)
{
glColor3f(1.0, 1.0, 1.0);
draw_circle(75, 615, d);
draw_circle(115, 635, d);
draw_circle(100, 590, d);
draw_circle(130, 590, d);
draw_circle(160, 615, d);
}
// cloud 2
for (d = 0; d <= 25; d++)
{
glColor3f(1.0, 1.0, 1.0);
draw_circle(75+400, 615 + 30, d);
draw_circle(115 + 400, 635 + 30, d);
draw_circle(100 + 400, 590 + 30, d);
draw_circle(130 + 400, 590 + 30, d);
draw_circle(160 + 400, 615 + 30, d);
}
}
else
{
//night sky
glColor3f(0.0, 0.0, 0.0);
glBegin(GL_POLYGON);
glVertex2f(0, 400);
glVertex2f(0, 700);
glVertex2f(1000, 700);
glVertex2f(1000, 400);
glEnd();


//moon
for (d = 0; d <= 30; d++)
```

```
{
glColor3f(1.0, 1.0, 1.0);
draw_circle(100, 625, d);
}
//star
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(520, 630);
glVertex2f(534, 630);
glVertex2f(527, 644);
glVertex2f(520, 639);
glVertex2f(534, 639);
glVertex2f(527, 625);
glEnd();


//star
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(250, 550);
glVertex2f(264, 550);
glVertex2f(257, 564);
glVertex2f(250, 559);
glVertex2f(264, 559);
glVertex2f(257, 545);
glEnd();


//star
```

```
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(320, 680);
glVertex2f(334, 680);
glVertex2f(327, 694);
glVertex2f(320, 689);
glVertex2f(334, 689);
glVertex2f(327, 675);
glEnd();


//star
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(640, 600);
glVertex2f(654, 600);
glVertex2f(647, 614);
glVertex2f(640, 609);
glVertex2f(654, 609);
glVertex2f(647, 595);
glEnd();


//star
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(770, 520);
glVertex2f(784, 520);
glVertex2f(777, 534);
```

```
glVertex2f(770, 529);
glVertex2f(784, 529);
glVertex2f(777, 515);
glEnd();


//star
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_TRIANGLES);
glVertex2f(900, 590);
glVertex2f(914, 590);
glVertex2f(907, 604);
glVertex2f(900, 599);
glVertex2f(914, 599);
glVertex2f(907, 585);
glEnd();
}


//grass
glColor3f(0.24, 0.79, 0.27);
glBegin(GL_POLYGON);
glVertex2f(0, 200);
glVertex2f(0, 400);
glVertex2f(1000, 400);
glVertex2f(1000, 200);
glEnd();
```

```
//road
glColor3f(0.0, 0.0, 0.0);
glBegin(GL_POLYGON);
glVertex2f(0, 0);
glVertex2f(0, 200);
glVertex2f(1000, 200);
glVertex2f(1000, 0);
glEnd();


//divider on the road
for (d = 0; d < 1000; d += 200)
{
glColor3f(1.0, 1.0, 1.0);
glBegin(GL_POLYGON);
glVertex2f(d, 90);
glVertex2f(d, 110);
glVertex2f(125 + d, 110);
glVertex2f(125 + d, 90);
glEnd();
}


//building 1
glColor3f(0.81, 0.60, 0.28);
glBegin(GL_POLYGON);
glVertex2f(50, 200);
glVertex2f(50, 500);
glVertex2f(250, 500);
```

```
glVertex2f(250, 200);
glEnd();

//window 1
glColor3f(0.52, 0.63, 0.60);
glBegin(GL_POLYGON);
glVertex2f(75, 375);
glVertex2f(75, 450);
glColor3f(0.19, 0.29, 0.26);
glVertex2f(150, 450);
glVertex2f(150, 375);
glEnd();



//door
glColor3f(0.52, 0.63, 0.60);
glBegin(GL_POLYGON);
glVertex2f(115, 200);
glVertex2f(115, 325);
glColor3f(0.19, 0.29, 0.26);
glVertex2f(185, 325);
glVertex2f(185, 200);
glEnd();



//building 2
glColor3f(0.55, 0.14, 0.14);
glBegin(GL_POLYGON);
glVertex2f(325, 200);
```

```
glVertex2f(325, 575);
glVertex2f(750, 575);
glVertex2f(750, 200);
glEnd();


//window 1,2,3
for (d = 375; d <= 625; d += 125)
{
glColor3f(0.52, 0.63, 0.60);
glBegin(GL_POLYGON);
glVertex2f(d, 450);
glVertex2f(d, 525);
glColor3f(0.19, 0.29, 0.26);
glVertex2f(75 + d, 525);
glVertex2f(75 + d, 450);
glEnd();
}


//window 4
glColor3f(0.52, 0.63, 0.60);
glBegin(GL_POLYGON);
glVertex2f(375, 325);
glVertex2f(375, 400);
glColor3f(0.19, 0.29, 0.26);
glVertex2f(450, 400);
glVertex2f(450, 325);
glEnd();
```

```
//door
glColor3f(0.52, 0.63, 0.60);
glBegin(GL_POLYGON);
glVertex2f(512.5, 200);
glVertex2f(512.5, 350);
glColor3f(0.19, 0.29, 0.26);
glVertex2f(612.5, 350);
glVertex2f(612.5, 200);
glEnd();

//signal base part
glColor3f(0.40, 0.21, 0.17);
glBegin(GL_POLYGON);
glVertex2f(920, 200);
glVertex2f(920, 325);
glVertex2f(945, 325);
glVertex2f(945, 200);
glEnd();

//signal top part
glColor3f(0.87, 0.82, 0.81);
glBegin(GL_POLYGON);
glVertex2f(895, 325);
glVertex2f(895, 475);
glVertex2f(970, 475);
glVertex2f(970, 325);
glEnd();
```

```
//signal lights
for (d = 0; d <= 20; d++)
{
glColor3f(0.0, 0.0, 0.0);
draw_circle(932.5, 450, d);
glColor3f(1.0, 1.0, 0.0);
draw_circle(932.5, 400, d);
glColor3f(0.0, 1.0, 0.0);
draw_circle(932.5, 350, d);
}

//car1
glColor3f(0.70, 0.10, 0.25);
glBegin(GL_POLYGON);
glVertex2f(front_car1 - 350 + x, 150);
glVertex2f(front_car1 - 365 + x, 165);
glVertex2f(front_car1 - 365 + x, 200);
glVertex2f(front_car1 - 275 + x, 250);
glVertex2f(front_car1 - 150 + x, 250);
glVertex2f(front_car1 - 100 + x, 200);
glVertex2f(front_car1 + x, 175);
glVertex2f(front_car1 + x, 160);
glVertex2f(front_car1 - 20 + x, 150);
glEnd();


//car1 window1
glColor3f(0.4, 0.4, 0.4);
glBegin(GL_POLYGON);
```

```
glVertex2f(front_car1 - 200 + x, 200);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car1 - 200 + x, 235);
glColor3f(0.4, 0.4, 0.4);
glVertex2f(front_car1 - 160 + x, 235);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car1 - 125 + x, 200);
glEnd();

//car1 window2
glColor3f(0.4, 0.4, 0.4);
glBegin(GL_POLYGON);
glVertex2f(front_car1 - 325 + x, 200);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car1 - 260 + x, 235);
glColor3f(0.4, 0.4, 0.4);
glVertex2f(front_car1 - 220 + x, 235);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car1 - 220 + x, 200);
glEnd();



//car1-lights
//rear
glColor3f(0.89, 0.47, 0.20);
glBegin(GL_POLYGON);
glVertex2f(front_car1 - 365 + x, 175);
glVertex2f(front_car1 - 365 + x, 200);
glColor3f(1.0, 1.0, 0.0);
```

```
glVertex2f(front_car1 - 350 + x, 200);
glVertex2f(front_car1 - 350 + x, 175);
glEnd();


//front
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_POLYGON);
glVertex2f(front_car1 - 20 + x, 165);
glVertex2f(front_car1 - 20 + x, 175);
glColor3f(0.89, 0.47, 0.20);
glVertex2f(front_car1 + x, 175);
glVertex2f(front_car1 + x, 165);
glEnd();


//car2
glColor3f(0.14, 0.42, 0.55);
glBegin(GL_POLYGON);
glVertex2f(front_car2 - 350 + x, 150);
glVertex2f(front_car2 - 350 + x, 200);
glVertex2f(front_car2 - 300 + x, 300);
glVertex2f(front_car2 - 125 + x, 300);
glVertex2f(front_car2 - 60 + x, 225);
glVertex2f(front_car2 + x, 190);
glVertex2f(front_car2 + x, 150);
glEnd();
```

```
//car2-window1
glColor3f(0.4, 0.4, 0.4);
glBegin(GL_POLYGON);
glVertex2f(front_car2 - 200 + x, 225);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car2 - 200 + x, 285);
glColor3f(0.4, 0.4, 0.4);
glVertex2f(front_car2 - 135 + x, 285);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car2 - 105 + x, 225);
glEnd();

//car2-window2
glColor3f(0.4, 0.4, 0.4);
glBegin(GL_POLYGON);
glVertex2f(front_car2 - 300 + x, 225);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car2 - 275 + x, 285);
glColor3f(0.4, 0.4, 0.4);
glVertex2f(front_car2 - 215 + x, 285);
glColor3f(0.1, 0.1, 0.1);
glVertex2f(front_car2 - 215 + x, 225);
glEnd();

//car2-lights
//front
glColor3f(1.0, 1.0, 0.0);
glBegin(GL_POLYGON);
glVertex2f(front_car2 - 25 + x, 175);
```

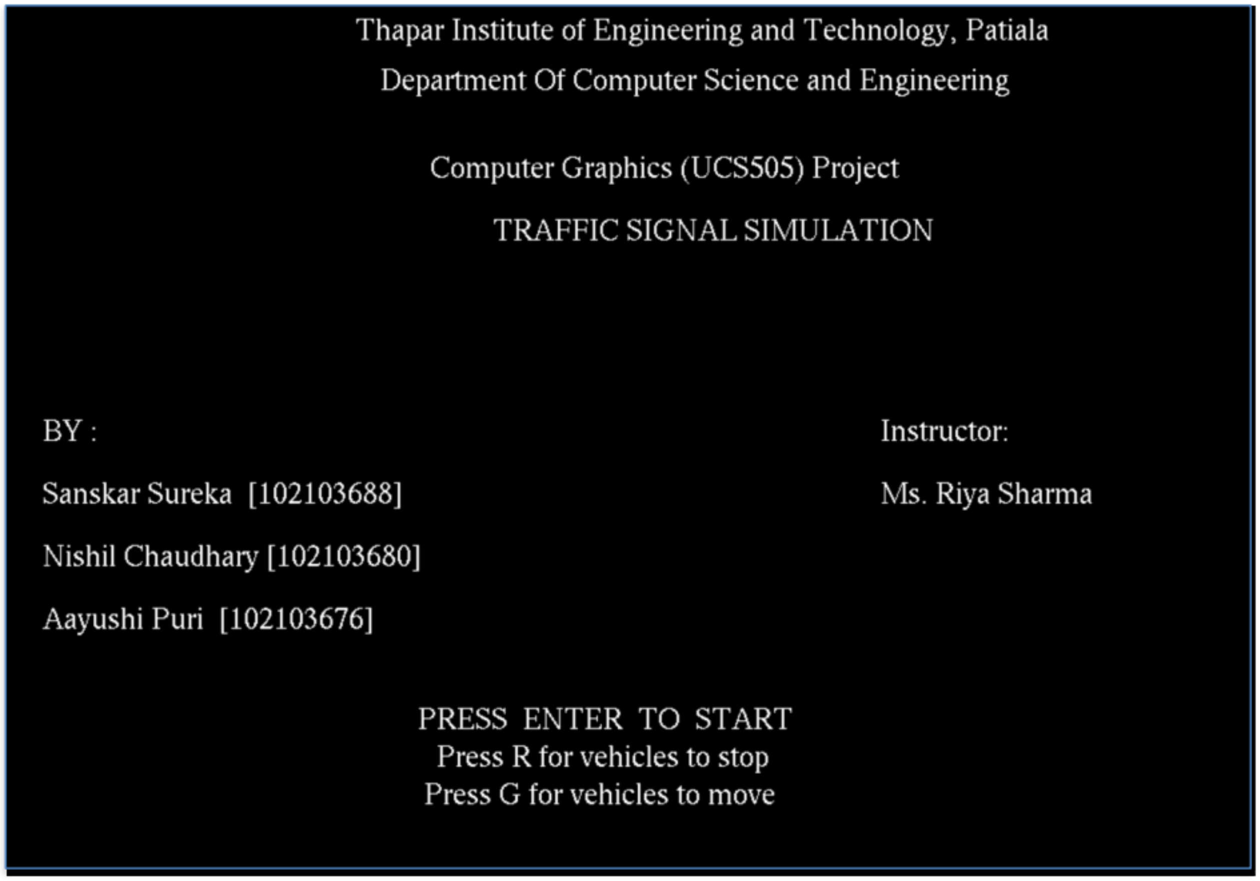
```
glVertex2f(front_car2 - 25 + x, 195);
glColor3f(0.89, 0.47, 0.20);
glVertex2f(front_car2 + x, 190);
glVertex2f(front_car2 + x, 175);
glEnd();


//rear
glColor3f(0.89, 0.47, 0.20);
glBegin(GL_POLYGON);
glVertex2f(front_car2 - 350 + x, 175);
glVertex2f(front_car2 - 350 + x, 200);
glColor3f(1.0, 1.0, 0.0);
glVertex2f(front_car2 - 325 + x, 200);
glVertex2f(front_car2 - 325 + x, 175);
glEnd();


//car1-wheels
for (d = 0; d <= 30; d++)
{
glColor3f(0.075, 0.075, 0.075);
draw_circle(front_car1 - 275 + x, 150, d);
glColor3f(0.075, 0.075, 0.075);
draw_circle(front_car1 - 125 + x, 150, d);
}


//car2-wheels
```

```
for (d = 0; d <= 40; d++)
{
glColor3f(0.075, 0.075, 0.075);
draw_circle(front_car2 - 265 + x, 150, d);
glColor3f(0.075, 0.075, 0.075);
draw_circle(front_car2 - 100 + x, 150, d);
}
}


void signal()
{
if (light == 0)
{
for (d = 0; d <= 20; d++)
{
glColor3f(0.0, 0.0, 0.0);
draw_circle(932.5, 350, d);
glColor3f(1.0, 0.0, 0.0);
draw_circle(932.5, 450, d);

}
}
else
{
for (d = 0; d <= 20; d++)
{
glColor3f(0.0, 0.0, 0.0);
draw_circle(932.5, 450, d);
glColor3f(0.0, 1.0, 0.0);
draw_circle(932.5, 350, d);
}
}
}


void keyboard(unsigned char key, int x, int y)
{
switch (key)
{
case 'r':
case 'R': light = 0;
break;
case 'g':
case 'G': light = 1;
break;
case 13: flag = 1;
break;
case 'm':
case 'M':day=1;
break;
case 'k':
case 'K':day=0;
break;
}
}
```

```
void idle()
{
if (flag == 1)
{
glClearColor(1.0, 1.0, 1.0, 1.0);
if (x > 2000) {
x = 0;
}
if (!(light == 0 && ((x >= 845 && x <= 895) || (x - 500 >= 845 && x - 500 <= 895))))
{
x += SPEED;
}

glutPostRedisplay();
}
}



void main_menu(int index)
{
switch (index)
{
case 1: day = 1;
break;



case 2: day = 0;
break;
case 3: exit(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, 1000.0, 0.0, 700.0);
}


void drawstring(int x, int y, std::string strs)
{
glRasterPos2f(x, y);
for (int i=0;i<strs.length();i++)
{
glColor3f(1.0, 1.0, 1.0);
glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, strs[i]);
}
}
```

```
void frontscreen()
{
glClearColor(0.0, 0.0, 0.0, 0.0);
glClear(GL_COLOR_BUFFER_BIT);
glColor3f(1.0, 1.0, 1.0);
drawstring(280, 670, "Thapar Institute of Engineering and Technology, Patiala");
drawstring(300, 630, "Department Of Computer Science and Engineering");
drawstring(340, 560, "Computer Graphics (UCS505) Project");
drawstring(390, 510, "TRAFFIC SIGNAL");
drawstring(30, 350, "BY :");
drawstring(30, 300, "Sanskar Sureka[102103688]");
drawstring(30, 250, "Nishil Chaudhary  [102103680]");
drawstring(30, 200, " Aayushi Puri [102103676]");
drawstring(700, 350, "Instructor:");
drawstring(700, 300, "Ms. Riya
Sharma");
drawstring(330, 120, "PRESS  ENTER  TO  START");
drawstring(345, 90, "Press R for vehicles to stop");
drawstring(335, 60, "Press G for vehicles to move");
glFlush();
}


void display()
{
glClear(GL_COLOR_BUFFER_BIT);
if (flag == 0)
frontscreen();
if (flag == 1)
{draw_object(); signal();
}
glFlush();
}


int main(int argc, char** argv)
{
glutInit(&argc, argv);
glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
glutInitWindowSize(1000.0, 700.0);
glutInitWindowPosition(0, 0);
glutCreateWindow("Traffic Signal");
glutDisplayFunc(display);
glutIdleFunc(idle);
glutKeyboardFunc(keyboard);
myinit();
glutCreateMenu(main_menu);
glutAddMenuEntry("Day", 1);
glutAddMenuEntry("Night", 2);
glutAddMenuEntry("Quit", 3);
glutAttachMenu(GLUT_RIGHT_BUTTON);
glutMainLoop();
}
```

Screenshots:



Thapar Institute of Engineering and Technology, Patiala

Department Of Computer Science and Engineering

Computer Graphics (UCS505) Project

TRAFFIC SIGNAL SIMULATION

BY :

Sanskar Sureka [102103688]

Nishil Chaudhary [102103680]

Aayushi Puri [102103676]

Instructor:

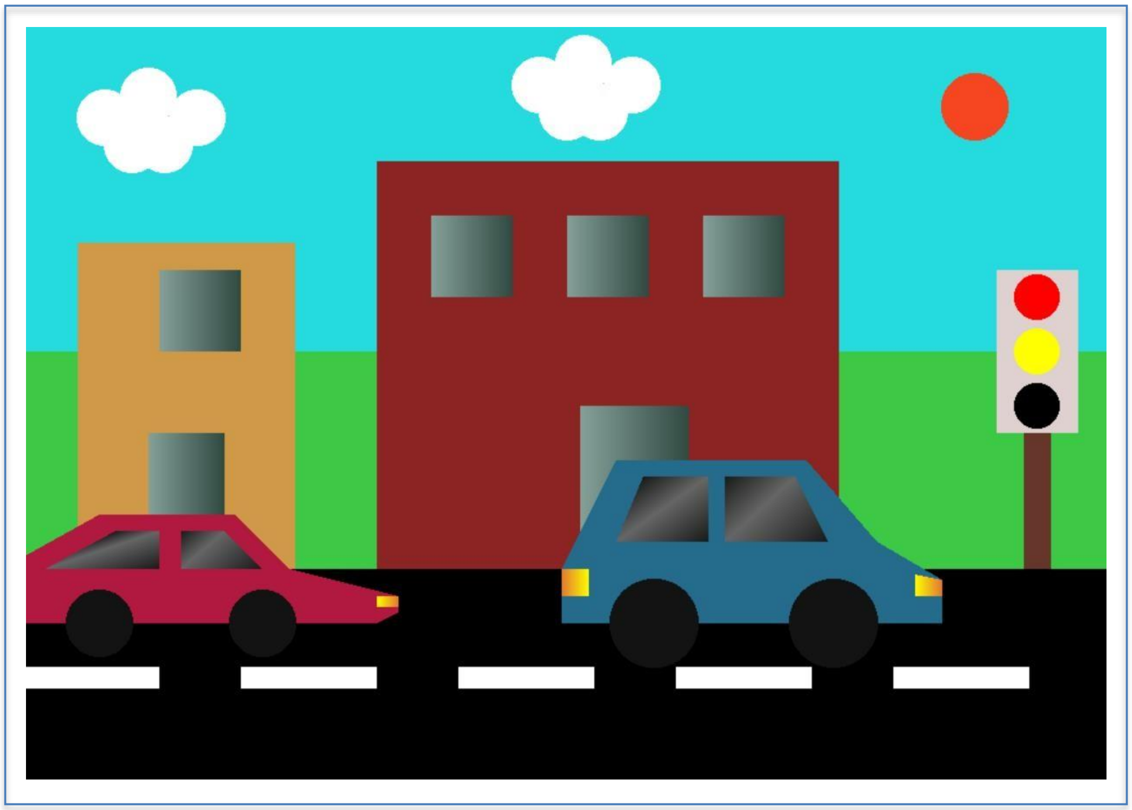
Ms. Riya Sharma

PRESS ENTER TO START

Press R for vehicles to stop

Press G for vehicles to move

A) **Day**



B) **Night**