Computer Graphics

Project on

Simulation of Traffic signal Using OpenGL

Table of Contents

Sr. No.	Description	Page No.
1.	Introduction to Project	3
2.	Computer Graphics concepts used	4
3.	User Defined Functions	6
4.	Code	7
5.	Output/ Screenshots	25

1. Introduction to Project

As we know, the traffic signals direct the flow of traffic with the exemption of signals with turning arrows which should be compulsory in accord with each other. Traffic signals help us to drive our vehicles in a safer manner lowering the risk of accidents if properly followed.

In this Project simulation, we are implementing Traffic Signal. We have three lights on the signal Red, Green, and Orange. Red means stop, Green means go, and Orange means ready to go.

The same method we implemented in our project which used in real life. We have used this traffic signal at a crossroad to avoid misshaping. When there is a green signal it means the vehicle can move or run on the road but if this green signal turns off and the red signal is turned on then it indicates the driver to stop his vehicle and same time signal allows some other side vehicle to cross the road. But if the vehicle is in the middle of a cross or the vehicle has crossed the signal then there is no meaning of red light for the driver. When we press button R then the red light will glow and all vehicles will stop. When we press the button G green light will glow and then vehicles start moving.

2. 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 human being 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 (open graphics library) is a standard specification defining a cross-language across platform API for writing applications that produce 2D and 3D computer graphics.
OpenGL is an application program interface (API) offering various functions to implement primitives, models, and images. This offer functions to create and manipulate render lighting, coloring, and viewing the models. OpenGL offers different coordinate systems and frames. OpenGL offers translation, rotation, and scaling of objects. The main idea behind this project is to implement traffic signals on roads.

- **glBegin()** - glBegin accepts a single argument that specifies in which of ten ways the vertices are interpreted. **Taking n as an integer count starting at one**, and N as the total number of vertices specified, the interpretations are as follows:

    - **GL_POINTS**. Treats each vertex as a single point.
    - **GL_POLYGON -** Draws a single, convex polygon. Vertices 1 through N define this polygon.
    - **GL_LINES -** Treats each pair of vertices as an independent line segment.

- **8-colors:-** By using different combinations of RGB values (ranging from 0-to 1) we were able to make 8 different colors that can be used by any tool in our software using
  glColor3f() — set the current color

- **glRasterPos** — specify the raster position for pixel operations

- **glutBitmapCharacter -**without using any display lists, glutBitmapCharacter renders the character in the named bitmap font

- **glutIdleFunc** sets the global idle callback to be func so a GLUT program can perform background processing tasks or continuous animation when window system events are not being received

- **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.

- **Translation -** Translation is a movement of objects without deformation. Every position or point is translated by the same amount

- **Mid Point Circle Drawing Algorithm –** for drawing circles

3. User-Defined Functions

- **void draw_circle() -** finding circle coordinates in aquadrant
- **void draw_pixel()** - Drawing pixels using glBegin(GL_POINTS)
- **Void plotpixels()** - Plot all 4 quadrants pixels of a circle
- **void draw_object() -** for drawing objects like day sky, night sky, sun clouds, moon, stars, grass, road, trees, buildings,vehicles etc.
- **void signal() -** setting green or red signal
- **void keyboard() -** Used to control the traffic signal and day-night transition
- **void idle()** - for translating objects like vehicles and clouds
- **void myinit() -** making intiliasations like matrixmode and many more
- **void display() -** to change the screen from frontscreen to main screen
- **void draw_string() -** Display strings on front screen

- **void frontscreen() -** Display name of project, team members, college name and mentor name on the front screen

4. CODE:

```
#include<stdlib.h>
#include<stdio.h>
#include<string.h>
#include<GL/glut.h>
#include<iostream>
using namespace std;

//speed of traffic
#define SPEED 20.0
//movement of car
float x = 0.0, m = 0.0;
//1 for green-light, 0 for red-light
int light = 1, car = 0;
//1 for day, 0 for night
int day = 1;
int d, flag = 0;
int l;


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)
    {
        //blue sky
        glColor3f(0.0f, 0.5f, 0.6f);
        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
        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);
        }

    }
    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
    if (day == 1)
    {
        glColor3f(0.0, 0.9, 0.0);
    }
    else
    {
        glColor3f(0.0, 0.3, 0.0);
    }

    glBegin(GL_POLYGON);
    glVertex2f(0, 200);
    glVertex2f(0, 400);
    glVertex2f(1000, 400);
    glVertex2f(1000, 200);
    glEnd();

    //road
    glColor3f(0.329412, 0.329412, 0.329412);
    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();
    }

    //tree1
    glColor3f(0.9, 0.2, 0.0);
    glBegin(GL_POLYGON);
    glVertex2f(50, 200);
    glVertex2f(50, 255);
    glVertex2f(65, 255);
```

```
    glVertex2f(65, 200);
    glEnd();
    int l;
    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);
    }

    //tree2
    glColor3f(0.9, 0.2, 0.0);
    glBegin(GL_POLYGON);
    glVertex2f(150, 200);
    glVertex2f(150, 255);
    glVertex2f(165, 255);
    glVertex2f(165, 200);
    glEnd();

    for (l = 0; l <= 30; l++)
    {
        glColor3f(0.0, 0.5, 0.0);
        draw_circle(140, 250, l);
        draw_circle(170, 250, l);
    }

    for (l = 0; l <= 25; l++)
    {
        glColor3f(0.0, 0.5, 0.0);
        draw_circle(150, 290, l);
        draw_circle(170, 290, l);
```

```
    }

    for (l = 0; l <= 20; l++)
    {
        glColor3f(0.0, 0.5, 0.0);
        draw_circle(160, 315, l);
    }

    // tree 3
    glColor3f(0.9, 0.2, 0.0);
    glBegin(GL_POLYGON);
    glVertex2f(250, 200);
    glVertex2f(250, 255);
    glVertex2f(265, 255);
    glVertex2f(265, 200);
    glEnd();

    for (l = 0; l <= 30; l++)
    {
        glColor3f(0.0, 0.5, 0.0);
        draw_circle(240, 250, l);
        draw_circle(280, 250, l);
    }

    for (l = 0; l <= 25; l++)
    {
        glColor3f(0.0, 0.5, 0.0);
        draw_circle(250, 290, l);
        draw_circle(270, 290, l);
    }

    for (l = 0; l <= 20; l++)
    {
        glColor3f(0.0, 0.5, 0.0);
        draw_circle(260, 315, l);
    }

    glColor3f(0.71, 0.65, 0.26);
    glBegin(GL_POLYGON);
    glVertex2f(400, 200);
    glVertex2f(400, 450);
    glVertex2f(780, 450);
    glVertex2f(780, 200);
    glEnd();
```

```
glColor3f(0.85, 0.85, 0.1);
glBegin(GL_POLYGON);
glVertex2f(350, 450);
glVertex2f(450, 500);
glVertex2f(730, 500);
glVertex2f(830, 450);
glEnd();

glColor3f(0.6, 0.8, 0.196078);
glBegin(GL_POLYGON);
glVertex2f(425, 200);
glVertex2f(425, 200);
glVertex2f(425, 400);
glVertex2f(450, 425);
glVertex2f(550, 425);
glVertex2f(575, 400);
glVertex2f(575, 200);
glEnd();

glBegin(GL_POLYGON);
glVertex2f(600, 200);
glVertex2f(600, 400);
glVertex2f(625, 425);
glVertex2f(725, 425);
glVertex2f(750, 400);
glVertex2f(750, 200);
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();


// door
glColor3f(0.329412, 0.329412, 0.329412);
glBegin(GL_POLYGON);
glVertex2f(625, 200);
glVertex2f(625, 375);
glVertex2f(725, 375);
glVertex2f(725, 200);
glEnd();

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

glColor3f(0.5, 0.5, 0.5);
glBegin(GL_POLYGON);
glVertex2f(895, 325);
glVertex2f(895, 475);
glVertex2f(970, 475);
glVertex2f(970, 325);
glEnd();


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);
}
```

```
    if (car == 1)
    {

        //car1
        glColor4f(1.0f, 0.0f, 1.0f, 0.0f);
        glBegin(GL_POLYGON);
        glVertex2f(30 + x, 150);
        glVertex2f(15 + x, 165);
        glVertex2f(15 + x, 200);
        glVertex2f(105 + x, 250);
        glVertex2f(230 + x, 250);
        glVertex2f(280 + x, 200);
        glVertex2f(380 + x, 175);
        glVertex2f(380 + x, 160);
        glVertex2f(360 + x, 150);
        glEnd();


        //car1 window1
        glColor3f(0.4, 0.4, 0.4);
        glBegin(GL_POLYGON);
        glVertex2f(180 + x, 200);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(180 + x, 235);
        glColor3f(0.4, 0.4, 0.4);
        glVertex2f(180 + x, 235);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(255 + x, 200);
        glEnd();

        //car1 window2
        glColor3f(0.4, 0.4, 0.4);
        glBegin(GL_POLYGON);
        glVertex2f(55 + x, 200);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(120 + x, 235);
        glColor3f(0.4, 0.4, 0.4);
        glVertex2f(160 + x, 235);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(160 + x, 200);
        glEnd();


        //car1-lights
```

```
        //rear
        glColor3f(0.89, 0.47, 0.20);
        glBegin(GL_POLYGON);
        glVertex2f(15 + x, 175);
        glVertex2f(15 + x, 200);
        glColor3f(1.0, 1.0, 0.0);
        glVertex2f(30 + x, 200);
        glVertex2f(30 + x, 175);
        glEnd();


        //front
        glColor3f(1.0, 1.0, 0.0);
        glBegin(GL_POLYGON);
        glVertex2f(360 + x, 165);
        glVertex2f(360 + x, 175);
        glColor3f(0.89, 0.47, 0.20);
        glVertex2f(380 + x, 175);
        glVertex2f(380 + x, 165);
        glEnd();


        //car2
        glColor3f(0.0f, 0.5f, 0.5f);
        glBegin(GL_POLYGON);
        glVertex2f(460 + x, 150);
        glVertex2f(460 + x, 200);
        glVertex2f(510 + x, 300);
        glVertex2f(685 + x, 300);
        glVertex2f(750 + x, 225);
        glVertex2f(810 + x, 190);
        glVertex2f(810 + x, 150);
        glEnd();


        //car2-window1
        glColor3f(0.4, 0.4, 0.4);
        glBegin(GL_POLYGON);
        glVertex2f(610 + x, 225);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(610 + x, 285);
        glColor3f(0.4, 0.4, 0.4);
        glVertex2f(675 + x, 285);
        glColor3f(0.1, 0.1, 0.1);
```

```
        glVertex2f(705 + x, 225);
        glEnd();

        //car2-window2
        glColor3f(0.4, 0.4, 0.4);
        glBegin(GL_POLYGON);
        glVertex2f(510 + x, 225);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(535 + x, 285);
        glColor3f(0.4, 0.4, 0.4);
        glVertex2f(595 + x, 285);
        glColor3f(0.1, 0.1, 0.1);
        glVertex2f(595 + x, 225);
        glEnd();

        //car2-lights
        //front
        glColor3f(1.0, 1.0, 0.0);
        glBegin(GL_POLYGON);
        glVertex2f(785 + x, 175);
        glVertex2f(785 + x, 195);
        glColor3f(0.89, 0.47, 0.20);
        glVertex2f(810 + x, 190);
        glVertex2f(810 + x, 175);
        glEnd();


        //rear
        glColor3f(0.89, 0.47, 0.20);
        glBegin(GL_POLYGON);
        glVertex2f(460 + x, 175);
        glVertex2f(460 + x, 200);
        glColor3f(1.0, 1.0, 0.0);
        glVertex2f(485 + x, 200);
        glVertex2f(485 + x, 175);
        glEnd();


        //car1-wheels
        for (d = 0; d <= 30; d++)
        {
            glColor3f(0.075, 0.075, 0.075);
            draw_circle(105 + x, 150, d);
            glColor3f(0.075, 0.075, 0.075);
```

```
            draw_circle(255 + x, 150, d);
        }


        //car2-wheels
        for (d = 0; d <= 40; d++)
        {
            glColor3f(0.075, 0.075, 0.075);
            draw_circle(545 + x, 150, d);
            glColor3f(0.075, 0.075, 0.075);
            draw_circle(710 + x, 150, d);
        }
    }
}


void signal()
{
    if (light == 0)
    {
        for (d = 0; d <= 20; d++)
        {
            glColor3f(1.0, 0.0, 0.0);
            draw_circle(932.5, 450, d);
            glColor3f(0.0, 0.0, 0.0);
            draw_circle(932.5, 350, 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 'd':
    case 'D':day = 1;
        break;
    case 'n':
    case 'N':day = 0;
        break;

    case 'c':
    case 'C':
        car = 1;
        break;
    }
}


void idle()
{
    if (flag == 1)
    {
        glClearColor(1.0, 1.0, 1.0, 1.0);
        if (light == 0 && (x >= 100 && x <= 500))
        {
            x += 0;
            m += SPEED / 20;
        }
        if (light == 0 && (x >= 530 && x <= 1020))
        {
            x += 0;
            m += SPEED / 20;
        }
        if (light == 1)
        {
            x += SPEED / 10;
            m += SPEED / 20;
        }
```

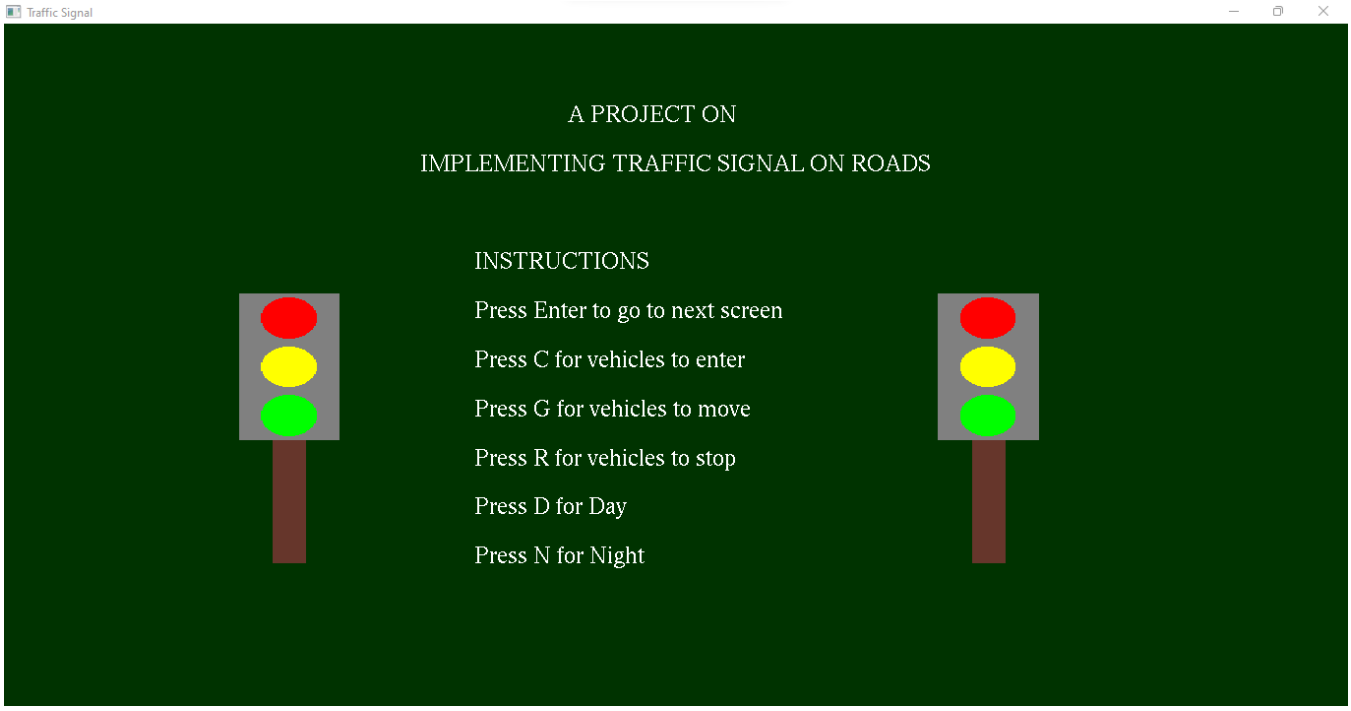
```
        if (x > 1050)
            x = 0.0;
        if (m > 500)
            m = 0.0;
        glutPostRedisplay();
    }
}

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, string str)
{
    glColor3f(1.0, 1.0, 1.0);
    glRasterPos2f(x, y);
    for (int i = 0; i < str.length(); i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24,
str[i]);
    }
}


void frontscreen()
{
    glClearColor(0.0, 0.2, 0.0, 0.0);
    glClear(GL_COLOR_BUFFER_BIT);
    glColor3f(1.0, 1.0, 1.0);
    drawstring(420, 600, "A PROJECT ON");
    drawstring(310, 550, "SIMULATION OF TRAFFIC SIGNAL
USING OPENGL");
    drawstring(350, 450, "INSTRUCTIONS:");
    drawstring(350, 400, "Press Enter to go to next
screen");
    drawstring(350, 350, "Press C for vehicles to enter");
    drawstring(350, 300, "Press G for vehicles to move");
```

```
    drawstring(350, 250, "Press R for vehicles to stop");
    drawstring(350, 200, "Press D for Day");
    drawstring(350, 150, "Press N for Night");

    //left signal
    glColor3f(0.40, 0.21, 0.17);
    glBegin(GL_POLYGON);
    glVertex2f(200, 150);
    glVertex2f(200, 275);
    glVertex2f(225, 275);
    glVertex2f(225, 150);
    glEnd();

    glColor3f(0.5, 0.5, 0.5);
    glBegin(GL_POLYGON);
    glVertex2f(175, 275);
    glVertex2f(175, 425);
    glVertex2f(250, 425);
    glVertex2f(250, 275);
    glEnd();

    for (d = 0; d <= 20; d++)
    {
        glColor3f(1.0, 0.0, 0.0);
        draw_circle(212.5, 400, d);
        glColor3f(1.0, 1.0, 0.0);
        draw_circle(212.5, 350, d);
        glColor3f(0.0, 1.0, 0.0);
        draw_circle(212.5, 300, d);
    }
    //right signal
    glColor3f(0.40, 0.21, 0.17);
    glBegin(GL_POLYGON);
    glVertex2f(720, 150);
    glVertex2f(720, 275);
    glVertex2f(745, 275);
    glVertex2f(745, 150);
    glEnd();

    glColor3f(0.5, 0.5, 0.5);
    glBegin(GL_POLYGON);
    glVertex2f(695, 275);
    glVertex2f(695, 425);
    glVertex2f(770, 425);
```

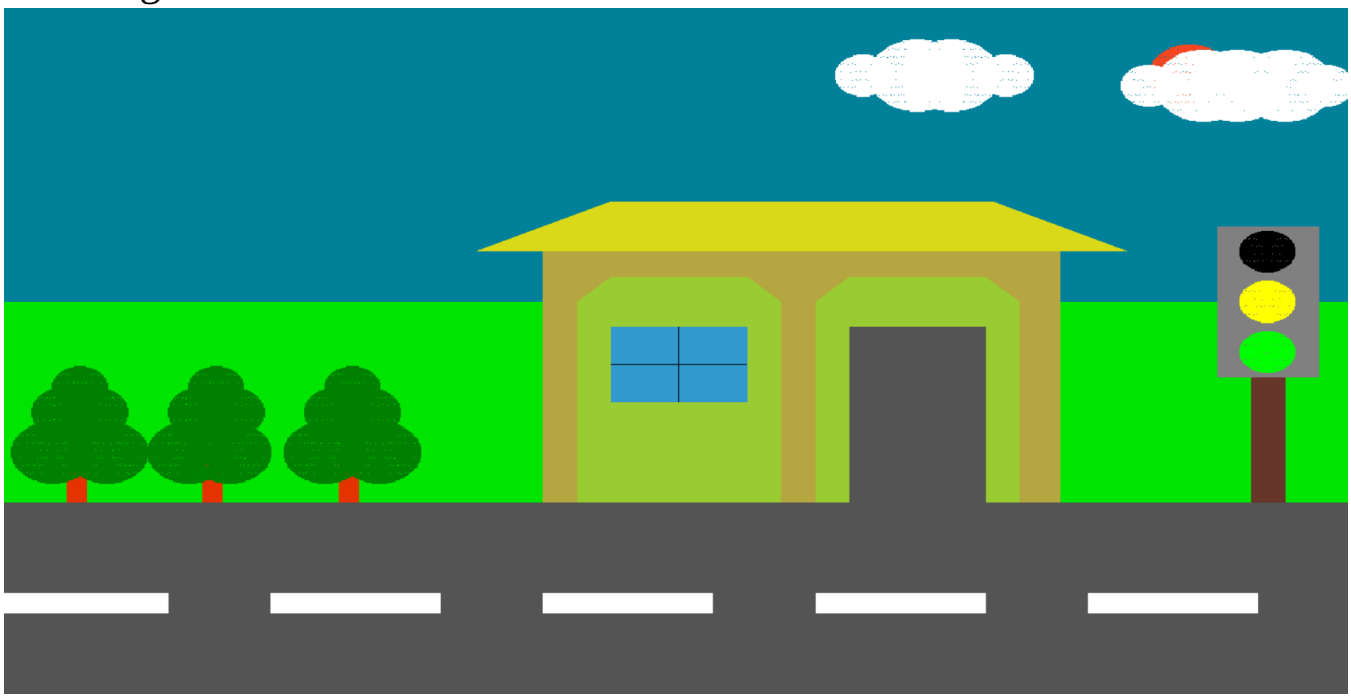
```
    glVertex2f(770, 275);
    glEnd();

    for (d = 0; d <= 20; d++)
    {
        glColor3f(1.0, 0.0, 0.0);
        draw_circle(732.5, 400, d);
        glColor3f(1.0, 1.0, 0.0);
        draw_circle(732.5, 350, d);
        glColor3f(0.0, 1.0, 0.0);
        draw_circle(732.5, 300, d);
    }
    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(1400.0, 800.0);
    glutInitWindowPosition(0, 0);
    glutCreateWindow("Traffic Signal");
    glutDisplayFunc(display);
    glutIdleFunc(idle);
    glutKeyboardFunc(keyboard);
    myinit();
    glutMainLoop();
}
```

5. Snapshots:

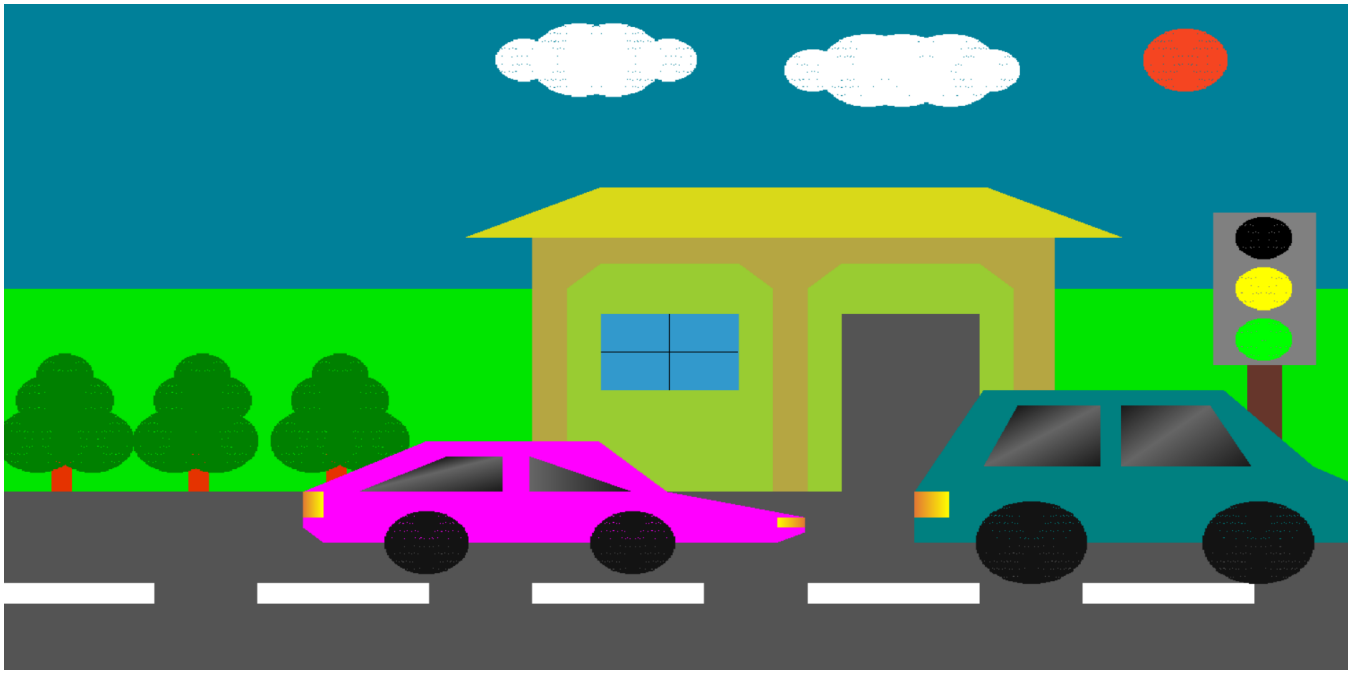
1. Front screen



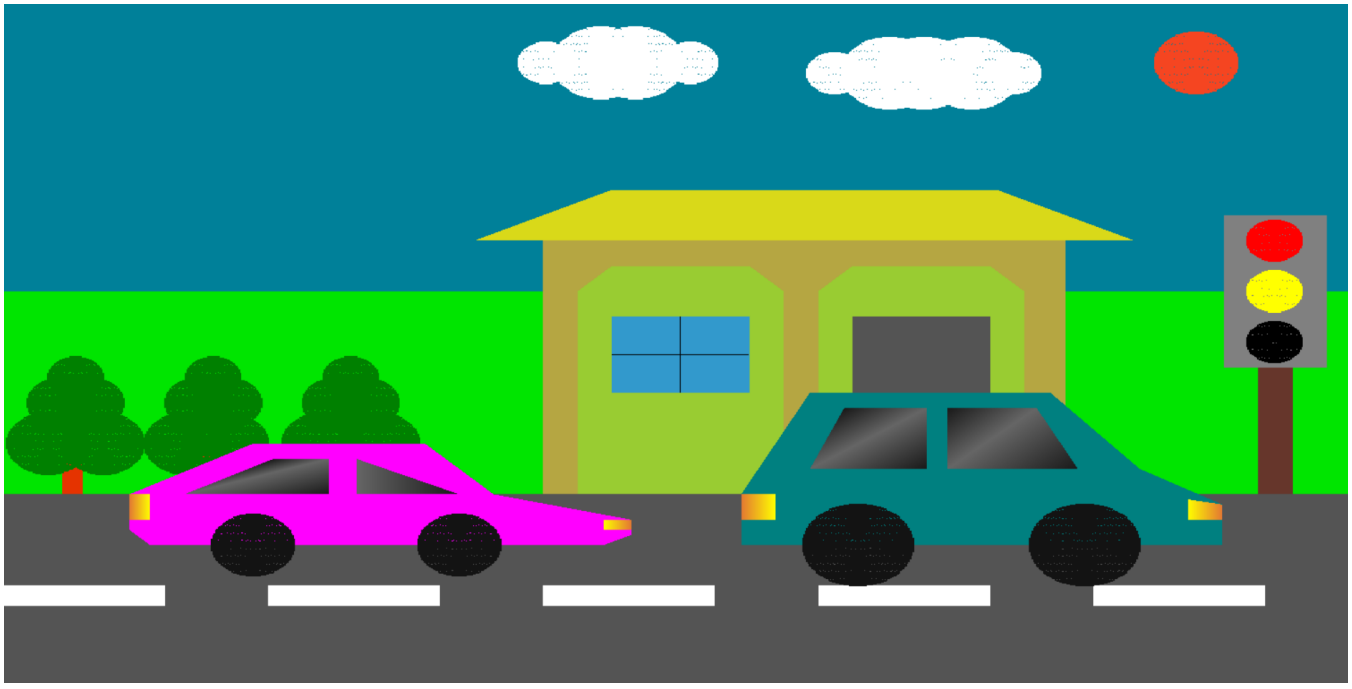
2. Background



3. Cars moving on green signal



4. Cars not moving on red signal



5. Night Back ground