UCS505

Computer Graphics

Project

Project Topic

Traffic Rule Trainer

Submitted To

Ms Kudratdeep Aulakh

Submitted By

Gaurish Garg (101803621)

Rishabh Malhotra (101803615)

Batch: (COE-28)

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

June 2021

Table of Contents:

Introduction

Traffic-Rule Trainer is an interactive program built in C++ using OpenGL library to teach kids about basic traffic rules. The player can interact with the application and give the car commands either to go left or right, forward or backward, and is prompted by the program when it fails to follow given traffic rules. The logic for traffic rules has been coded in C++.

The program makes use of various inbuilt OpenGL functions and a few user-defined functions.

The program provides an interactive user interface. Initially on execution, the program shows a home screen featuring the details of the authors of the project. After pressing Home Key on the Keyboard, the initial scene is setup including the traffic signals, footpaths and car. The user is free to move the car by using the arrow keys on the keyboard. If the user violates any of the following three conditions, then corresponding warning message is displayed.

- Jumping Red Light
- Not moving in Left Lane
- Running Over Footpath

The program provides suitable user interface to teach traffic rules to kids in an interactive way.

Concepts of Computer Graphics Used

Different graphics concepts are used in our project. Some are mentioned as:

- We learned how to make various characters and objects using basic shapes such as gl_polygon.
- We learned how to define a coordinate system using orthographic projection using the gluortho2d in-built function.
- We learned to make shapes having different colors giving a similar effect of gradient.
- We have learnt how to take input from keys and special keys to control the movement in the game.
- We have learned how to move certain objects and do animation using glutTimerFunc in-built function.
- We have learned how to change state based on certain condition and display that state change on the screen.

User Defined Functions

- **void display()**

  This function is used to setup and initially render the home page and after pressing home button, it then renders out the car traffic light scene.

- **void output(float x, float y, float r, float g, float b, string input)**

  This function is used to render out string to the screen, in which x and y are coordinates on the screen and r, g and b are color values, input is the text that needs to be displayed.

- **void update(int value)**

  This function is used to update the traffic light signal at a set interval of time as mentioned in the code. It is a conditionally recursive function to update the traffic light signal periodically with different time intervals of 15 seconds for red, 12 seconds for green and 3 seconds for yellow. This function is recursively called in the GlutTimerFunc.

- **void road()**

  This function defines the coordinates of the road through GL_POLYGON which is then rendered in the display function.

- **void middlelines()**

  This function is used to draw the middle line on the defined road using GL_LINES.

- **void footpath()**

  This function is used to draw the footpaths by defining the coordinates with GL_POLYGON.

- **void car()**

  This function to render out the car using gl_polygon. It also contains the code for translation of the car based on the translation values updated in the specialkey() function.
  We've used glTranslatef() to translate the car.

- **void conditions()**

  The function is used to check the boundary and red light jump conditions and renders corresponding errors using the output function, meaning it renders 'don't jump red light' when the user is crossing the road on a red light, and it also notifies the user if the car jumps over the footpath.

- **void signal(void)**

  This function draws the signal stand and paints the traffic light on the user screen based on the active traffic light.

- **void specialkey(int key, int x, int y)**

  This function takes keyboard strokes as input and updates the translation values for the car thus enabling the execution of corresponding action like translating car forward on pressing Up key on keyboard. This function is called in the GlutSpecialFunc.

Code

```
#include<stdlib.h>
#include<GL/glut.h>
#include<iostream>
using namespace std;
int moveh = 0;
int movev = 0;
bool R = true;
bool G = false;
bool Y = false;
int times = 15000;
int updater = times;
int startflag = 0;
void specialkey(int key, int x, int y)
{
	switch (key)
	{
	case GLUT_KEY_UP:
		movev++;
		glutPostRedisplay();
		break;
	case GLUT_KEY_DOWN:
		movev--;
		glutPostRedisplay();
		break;
	case GLUT_KEY_LEFT:
		moveh--;
		glutPostRedisplay();
		break;
	case GLUT_KEY_RIGHT:
		moveh++;
		glutPostRedisplay();
		break;
	case GLUT_KEY_HOME:
		startflag++;
		glutPostRedisplay();
	}
}
void output(float x, float y, float r, float g, float b, string input)
{
	glColor3f(r, g, b);
	glRasterPos2f(x, y);
	int len, i;
	len = input.length();
	for (i = 0; i < len; i++) {
		glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, input[i]);
	}
}
void myinit()
```

```
{
	glClearColor(0.0, 1.0, 0.0, 0.0); //rgb and alpha for transparency idhar green karlo
	//glColor3f(0.0, 0.0, 0.0);
	glPointSize(10.0); //pt size in pixels //PUT 10 PX HERE
	glMatrixMode(GL_PROJECTION);
	glLoadIdentity();
	gluOrtho2D(-249, 249, -249, 249.0);

}
void road()
{
	glColor3f(0.2773, 0.2812, 0.2968);
	glBegin(GL_POLYGON);
	glVertex2f(-150, -249);
	glVertex2f(-150, 249);
	glVertex2f(150, 249);
	glVertex2f(150, -249);
	glEnd();
}
void middleline()
{
	glBegin(GL_LINES);
	glColor3f(1.0, 1.0, 1.0);
	glLineWidth(10);
	float y = -250;
	while (y <= 250)
	{
		glVertex2f(0, y);
		glVertex2f(0, y + 10);

		y = y + 20;

	}

	glEnd();
}
void footpath()
{

	glBegin(GL_POLYGON);
	glColor3f(1.0, 1.0, 1.0);
	glVertex2f(-170, -249);
	glVertex2f(-170, 249);
	glVertex2f(-150, 249);
	glVertex2f(-150, -249);
	glEnd();


	glBegin(GL_POLYGON);
	glColor3f(1.0, 1.0, 1.0);
```

```
		glVertex2f(170, -249);
		glVertex2f(170, 249);
		glVertex2f(150, 249);
		glVertex2f(150, -249);
		glEnd();
}
void car()
{
		glPushMatrix();
		glTranslatef(moveh, movev, 0);
		//front hood
		glColor3f(1, 0, 0);
		glBegin(GL_POLYGON);
		glVertex2f(-100, -125);
		glVertex2f(-87.5, -105);
		glVertex2f(-62.5, -105);
		glVertex2f(-50, -125);
		glEnd();
		//front windsheild
		glColor3f(0.5273, 0.8078, 0.9216);
		glBegin(GL_POLYGON);
		glVertex2f(-100, -125);
		glVertex2f(-93.75, -115);
		glVertex2f(-56.25, -115);
		glVertex2f(-50, -125);
		glEnd();
		//body
		glColor3f(1, 0.0, 0.0);
		glBegin(GL_POLYGON);
		glVertex2f(-100, -200);
		glVertex2f(-100, -125);
		glVertex2f(-50, -125);
		glVertex2f(-50, -200);
		glEnd();
		//bodyend
		//back hood
		glColor3f(1, 0, 0);
		glBegin(GL_POLYGON);
		glVertex2f(-100, -200);
		glVertex2f(-87.5, -220);
		glVertex2f(-62.5, -220);
		glVertex2f(-50, -200);
		glEnd();
		//front windsheild
		glColor3f(0.5273, 0.8078, 0.9216);
		glBegin(GL_POLYGON);
		glVertex2f(-100, -200);
		glVertex2f(-93.75, -210);
		glVertex2f(-56.25, -210);
		glVertex2f(-50, -200);
```

```
	glEnd();
	glPopMatrix();
	//cout << "\n" << movev;
	//glutPostRedisplay();


}
void update(int value)
{
	cout << value;
	if (R)
	{
		cout << "\nIn Red Condition: " << R << " " << Y << " " << G << endl;
		//TO MAKE TO GREEN
		G = true;
		Y = false;
		R = false;
		updater = 0.8*times;
		cout << "\nAfter exxec Red Condition: " << R << " " << Y << " " << G <<
endl;
		glutTimerFunc(updater, update, 0);
		return;

	}
	if (G)
	{
		cout << "\nIn green Condition: " << R << " " << Y << " " << G << endl;
		Y = true;
		G = false;
		R = false;
		updater = 0.2*times;
		cout << "\nAfter exxec green Condition: " << R << " " << Y << " " << G <<
endl;
		glutTimerFunc(updater, update, 0);

		return;

	}
	if (Y)
	{
		cout << "\nIn yellow Condition: " << R << " " << Y << " " << G << endl;
		R = true;
		G = false;
		Y = false;
		updater = times;
		cout << "\nafter exec yellow Condition: " << R << " " << Y << " " << G <<
endl;
		glutTimerFunc(updater, update, 0);

		return;
```

```
		}

		//glutPostRedisplay();
}
void signal()
{
		//plane signal stand
		glColor3f(0.0, 0.0, 0.0);
		glBegin(GL_POLYGON);
		glVertex2f(-135, 120);
		glVertex2f(-135, 230);
		glVertex2f(-165, 230);
		glVertex2f(-165, 120);
		glEnd();
		glBegin(GL_POLYGON);
		glVertex2f(-145, 90);
		glVertex2f(-145, 120);
		glVertex2f(-155, 120);
		glVertex2f(-155, 90);
		glEnd();
		//slots for colors
		glColor3f(0.515625, 0.40625
			, 0.4375);
		glBegin(GL_POLYGON);
		glVertex2f(-140, 125);
		glVertex2f(-140, 155);
		glVertex2f(-160, 155);
		glVertex2f(-160, 125);
		glEnd();
		glBegin(GL_POLYGON);
		glVertex2f(-140, 160);
		glVertex2f(-140, 190);
		glVertex2f(-160, 190);
		glVertex2f(-160, 160);
		glEnd();
		glBegin(GL_POLYGON);
		glVertex2f(-140, 195);
		glVertex2f(-140, 225);
		glVertex2f(-160, 225);
		glVertex2f(-160, 195);
		glEnd();
		//colors
		if (R)
		{
			glColor3f(1.0, 0.0, 0.0);
			glBegin(GL_POLYGON);
			glVertex2f(-140, 195);
			glVertex2f(-140, 225);
			glVertex2f(-160, 225);
```

```
                glVertex2f(-160, 195);
                glEnd();
        }
        if (G)
        {
                glColor3f(0.0, 1.0, 0.0);
                glBegin(GL_POLYGON);
                glVertex2f(-140, 125);
                glVertex2f(-140, 155);
                glVertex2f(-160, 155);
                glVertex2f(-160, 125);
                glEnd();
        }
        if (Y)
        {
                glColor3f(1.0, 1.0, 0.0);
                glBegin(GL_POLYGON);
                glVertex2f(-140, 160);
                glVertex2f(-140, 190);
                glVertex2f(-160, 190);
                glVertex2f(-160, 160);
                glEnd();
        }
}
void conditions()
{
        if (((-105 + movev > 90)&&(-105+movev < 210))&&R) //WHERE 125 IS VET
POSITION OF RED LIGHT
        {
                //red light jump condition
                cout << "\nRed Light Jump";
                output(0, -245, 1, 1, 1, "Don't Jump Red Light");
        }
        if (-50 + moveh > 0)
        {
                cout << "\nKeep Left";
                output(-240, -245, 1, 1, 1, "Keep Left");

        }
        if ((-100 + moveh < -150) || (-100 + moveh > 150))
        {
                cout << "\nDon't Jump Over FootPath";
                output(-240, -245, 1, 1, 1, "Don't Run Over Footpath");

        }
}

void flash()
{
        glBegin(GL_POLYGON);
```

```
        glColor3f(0,0,1);
        glVertex2f(-250,-250);
        glVertex2f(-250, -220);
        glVertex2f(250, -220);
        glVertex2f(259, -250);
        glEnd();
}
void display()
{
        if (startflag==0)
        {

                glClearColor(0.0, 0.0, 0.0, 0.0); //rgb and alpha for transparency idhar green
karlo
                glClear(GL_COLOR_BUFFER_BIT);
                string Project_Name = "OpenGL Project";
                float length = Project_Name.length();
                output(0-length*10/2,200, 1, 1, 1, Project_Name);

                string Project_Name2 = "Submitted By";
                float length2 = Project_Name2.length();
                output(0 - length2 * 10 / 2, 160, 1, 1, 1, Project_Name2);

                string Project_Name3 = "Gaurish Garg - 101803621";
                float length3 = Project_Name3.length();
                output(0 - length3 * 10 / 2, 120, 1, 1, 0, Project_Name3);

                string Project_Name4 = "Rishabh Malhotra - 101803615";
                float length4 = Project_Name4.length();
                output(0 - length4 * 10 / 2, 80, 1, 1, 0, Project_Name4);

                string Project_Name5 = "Submitted To";
                float length5 = Project_Name5.length();
                output(0 - length5 * 10 / 2, 40, 1, 1, 1, Project_Name5);

                string Project_Name6 = "Ms Kudratdeep Aulakh";
                float length6 = Project_Name6.length();
                output(0 - length6 * 10 / 2, 0, 1, 1, 0, Project_Name6);
                glFlush();

                string Project_Name7 = "Computer Science and Engineering Department";
                float length7 = Project_Name7.length();
                output(0 - length7 * 10 / 2, -40, 1, 1, 1, Project_Name7);

                string Project_Name8 = "TIET Patiala";
                float length8 = Project_Name8.length();
                output(0 - length8 * 10 / 2, -80, 1, 1, 1, Project_Name8);
                glFlush();
                //glutPostRedisplay();
        }
```

```
	if (startflag==1)
	{
		glClearColor(0.0, 1.0, 0.0, 0.0); //rgb and alpha for transparency idhar green
karlo
		glClear(GL_COLOR_BUFFER_BIT);
		road();
		footpath();
		middleline();
		signal();
		car();
		flash();
		conditions();
		glFlush();
		glutPostRedisplay();
	}

	//glutSwapBuffers();
}
void main(int argc, char** argv)
{
	glutInit(&argc, argv);
	glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
	glutInitWindowSize(500, 500);  //for window size
	glutInitWindowPosition(0, 0); //for window position
	glutCreateWindow("Traffic-Rule-Trainer"); //title window
	glutSpecialFunc(specialkey); //specialkey function for keyboard input
	myinit();
	glutDisplayFunc(display);
	glutTimerFunc(updater, update, 0);
	glutMainLoop();
}
```

Screenshots

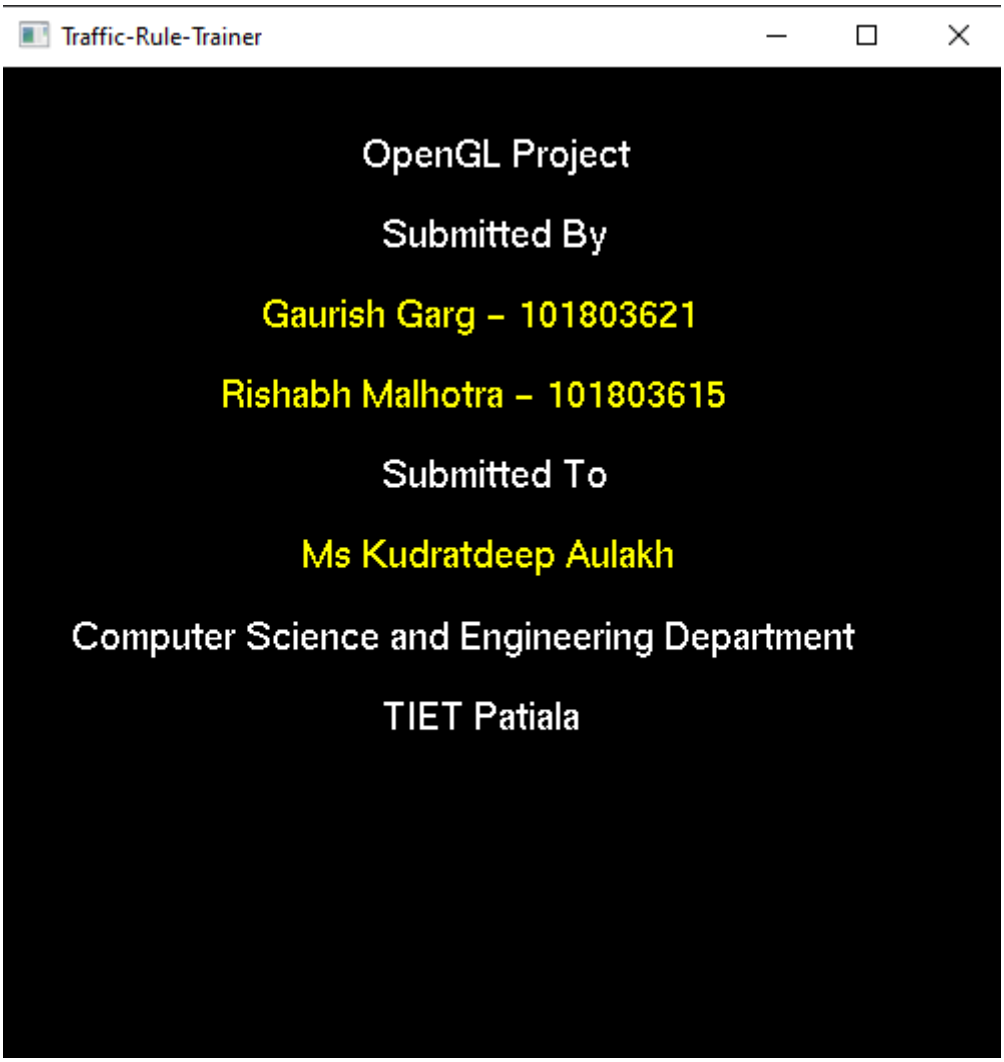

Figure 1 Home Screen

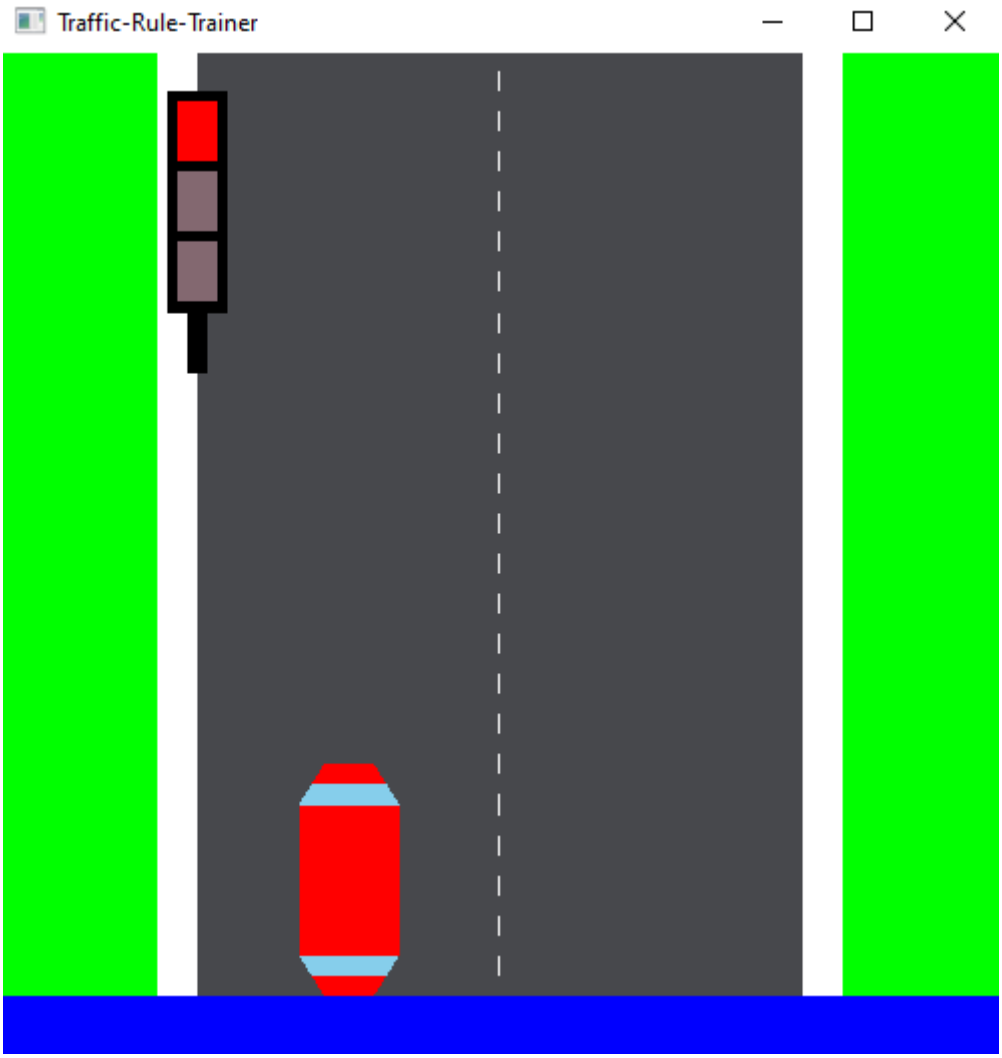

Figure 2 Initial Scene after Homescreen

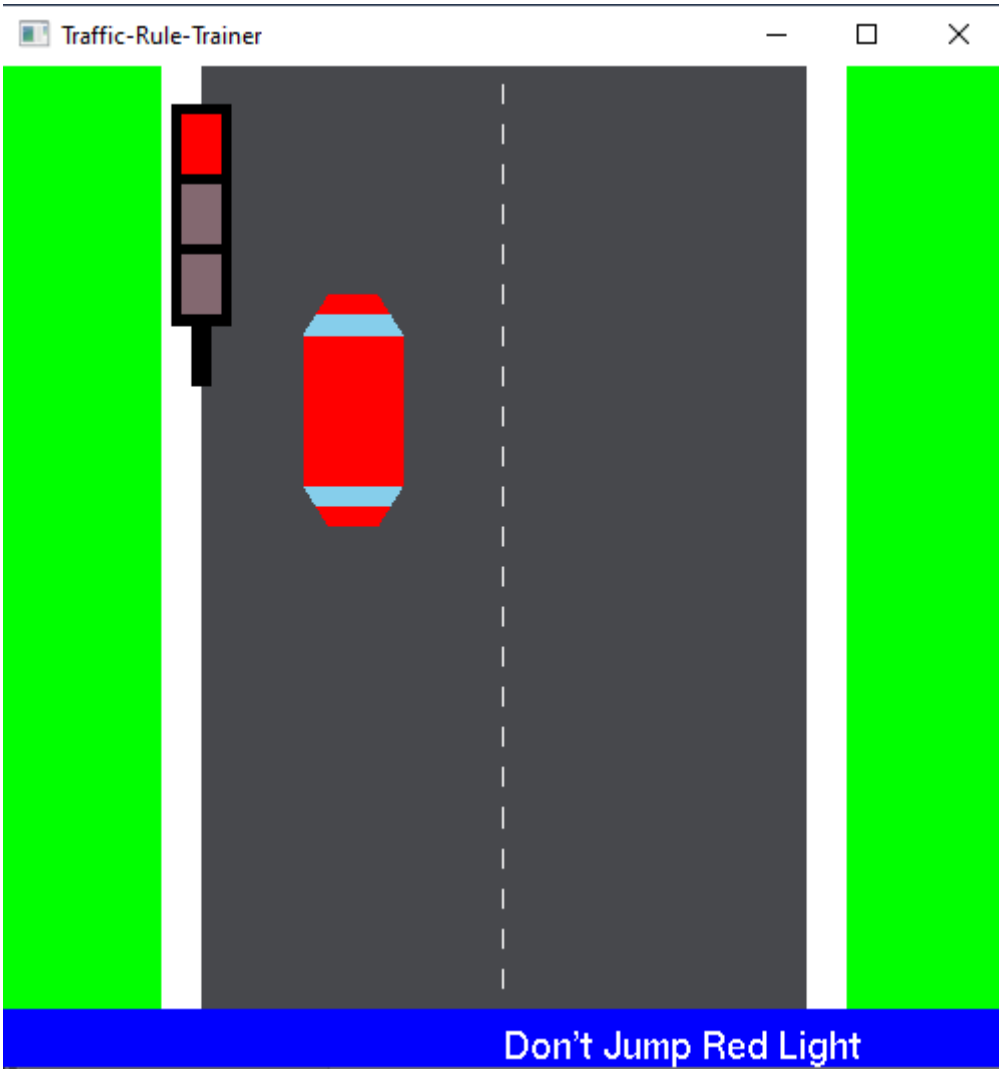

Figure 3 Red Light Jump Warning

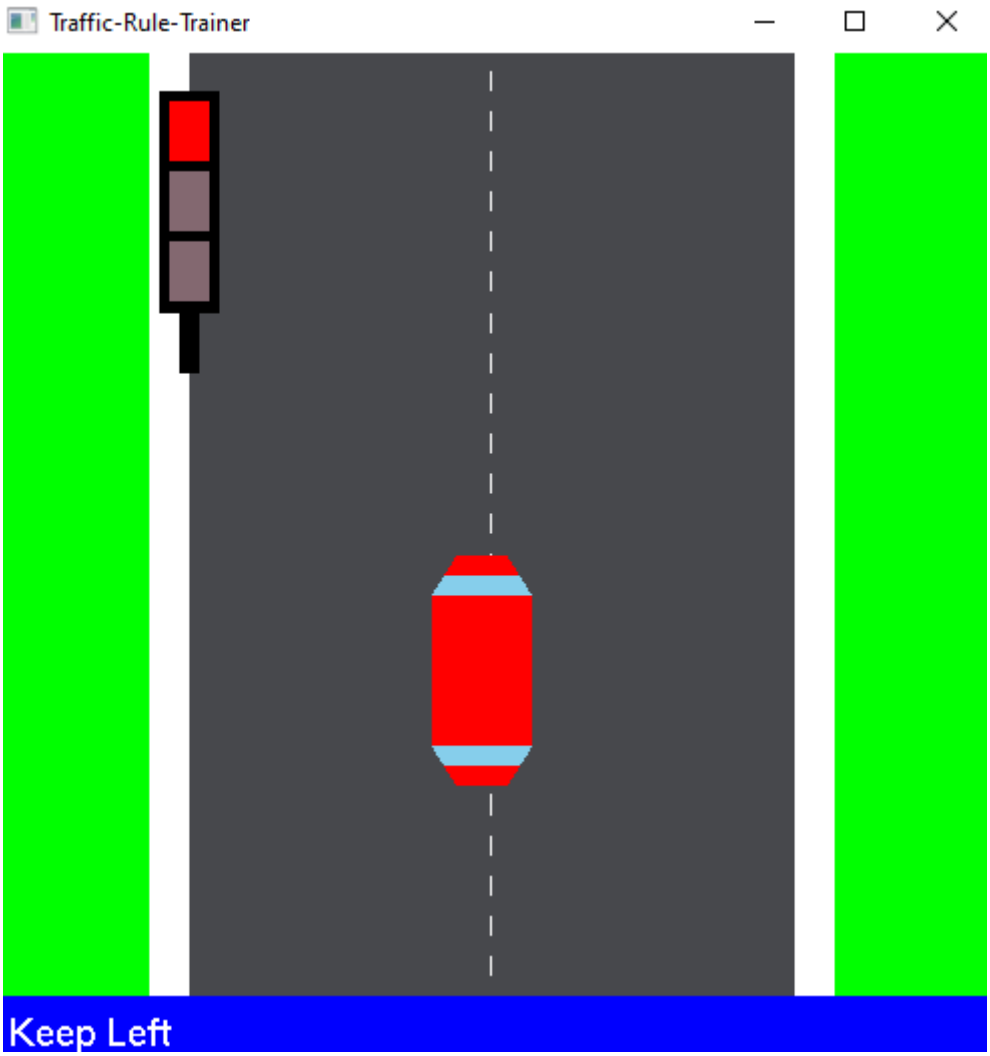

Figure 4 Keep Left Warning when car is crossing the center line

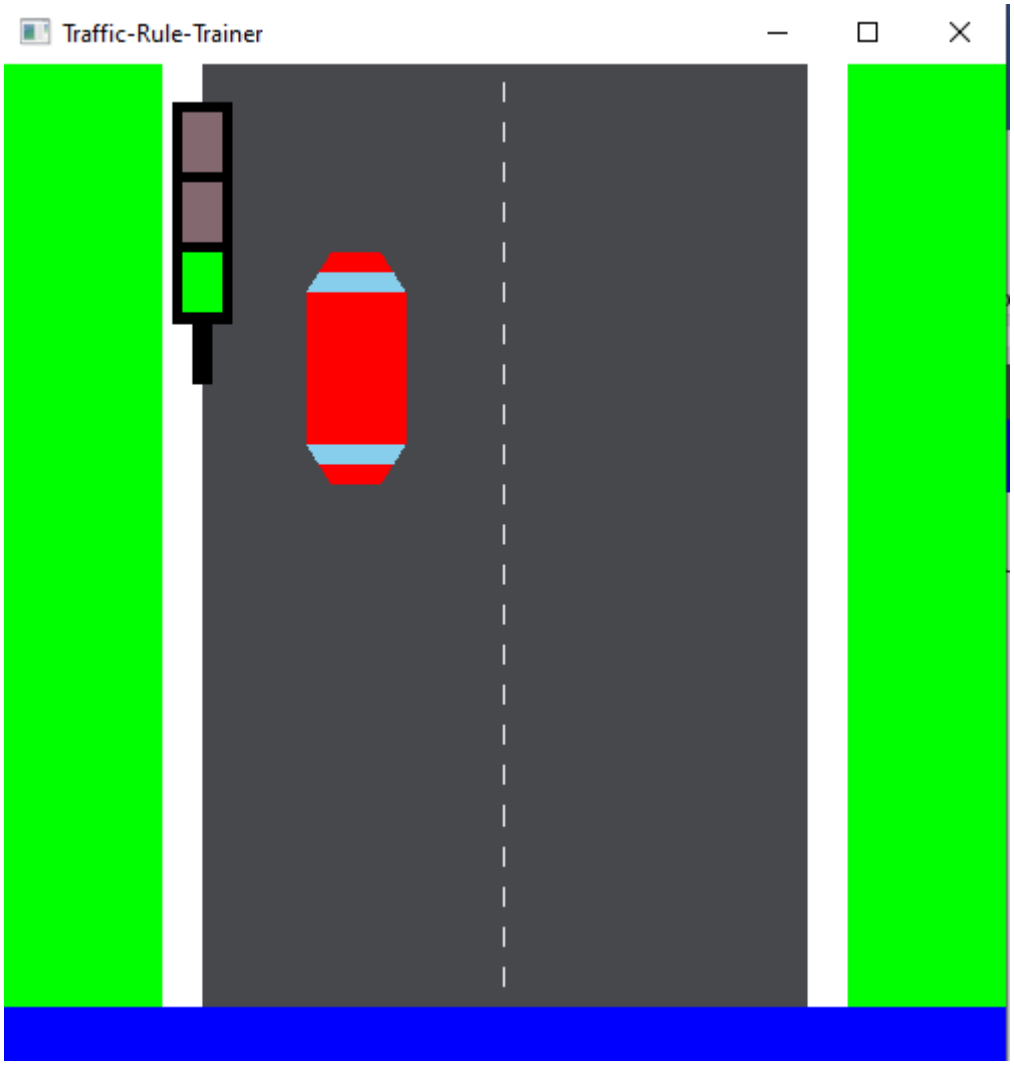

Figure 5 No warning when car is crossing in green light

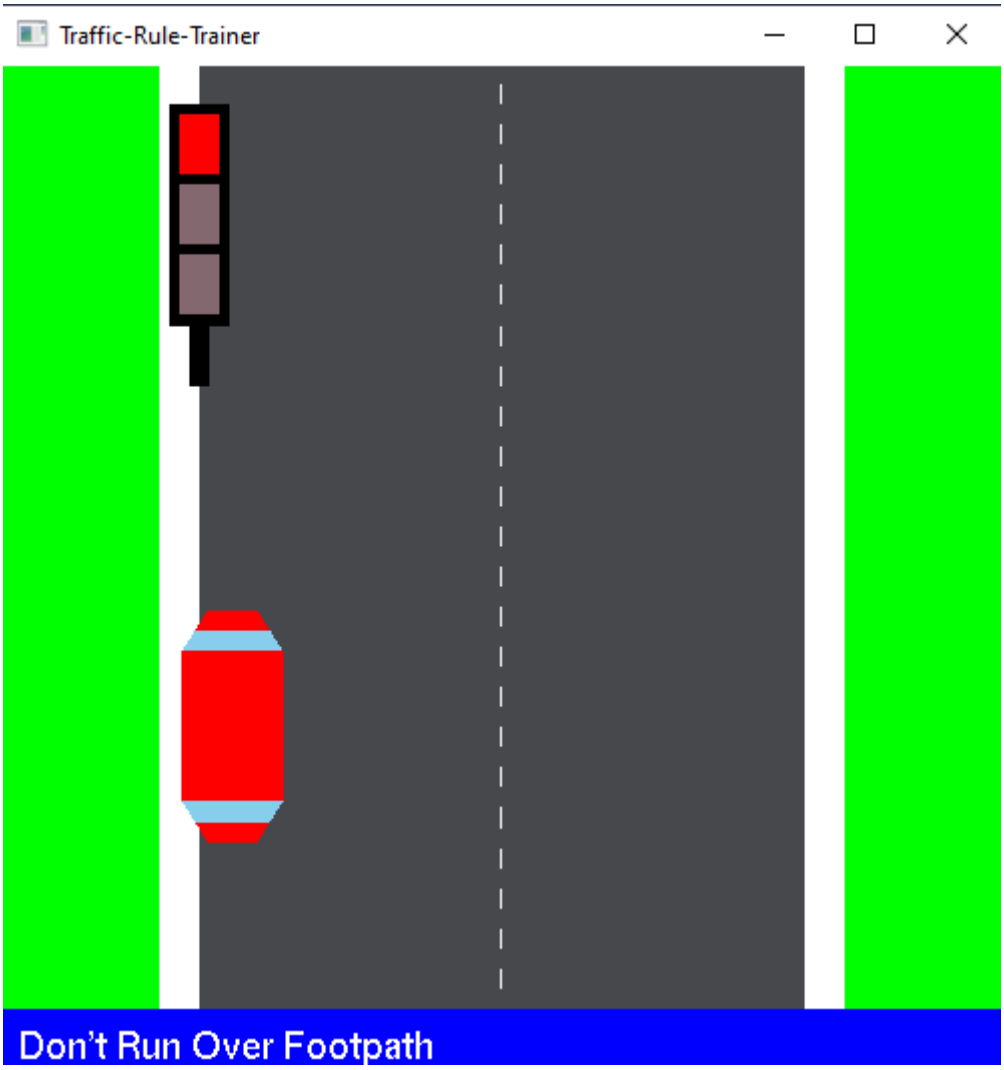

Figure 6 Warning when car runs over footpath