Computer Graphics

(UCS505)

Project Report On

<u>2-D Racing Car</u>

<u>Submitted by:-</u>

Parth Vohra (102016044)

Bipasha Gupta (102196004)

<u>Group</u>: 3CSE10

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

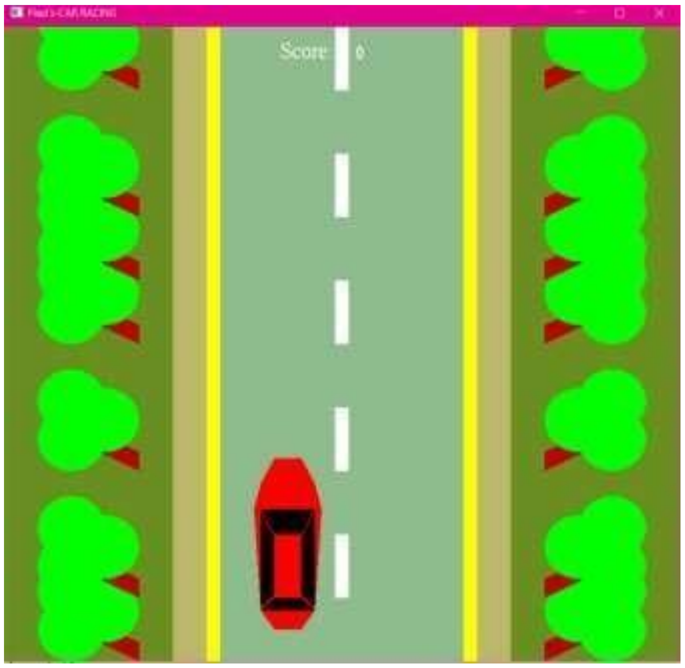
Computer Science and Engineering Department

Thapar Institute of Engineering and Technology, Patiala

Table of contents

1. Introduction of Project

For this Car Racing Game, we would like to accomplish a video game imitating the existing game with a projective view. The theme of our game is to increase the concentration of the player as the speed of the car increases with levels along with overcoming the obstacles that come in it’s path. The player’s goal is to make the highest possible score to avoid bumping into the obstacles.



This project demonstrates the creation of a moving racing car along with a race track and scenery. OpenGL is used to make this possible by virtue of its various functionalities.

We make use of simple geometric figures like rectangles and polygons to construct the parts of racing car and the track. Circles and parallelograms are used to generate the trees. Rectangles are used to generate obstacles.

The code implemented makes use of various OpenGL functions for translation and keyboard call back function, built-in functions for solids and many more.

The concepts of computer graphics stand a backbone to achieve the aforementioned idea. Primitive drawing, event driven interactions and basic animation have been the important concepts brought out by this application.

The report is chalked out into sections describing the computer graphics concepts used superseded by the briefing on functions used. Following this, the detailed description of how the implementation is done effectively using these functions and C++ language is presented. The source code is provided along with necessary comments to enhance readability of code. The screenshots have been provided for amelioration of our little effort. The conclusion and the future enhancements proposed conclude the report. The maximum efforts are been made to ensure that the view is aesthetically pleasing and eye-catching.

2. Instructions to play the game:

To start the car first of all, press UP arrow from the keyboard.

Once the car starts moving just control the car movement using LEFT and RIGHT keyboard keys.

In case you hit with an obstacle the game gets over and to start the game again press the UP key twice or thrice.

3. Computer Graphics concepts used:

In computer graphics, use graphics.h which provide direct functions to draw different coordinate shapes (like circle, rectangle etc). By using these concepts we can draw different objects like car, track, trees etc. In this program, we will draw a moving car using rectangles and polygons. OpenGL uses several matrices to transform geometry and associated data. Those matrices are:

- **Modelview** – places object geometry in the global, *unprojected* space
- **Projection** – projects global coordinates into clip space; you may think of it as kind of a lens
- **Texture** – adjusts texture coordinates before; mostly used to implement texture projection (i.e. projecting a texture as if it was a slide in a projector)
- **Color** – adjusts the vertex colors. Seldomly touched at all

All these matrices are used all the time. Since they follow all the same rules OpenGL has only one set of matrix manipulation functions: glPushMatrix, glPopMatrix.

glPushMatrix() :
push the current matrix into the current matrix stack.
glPopMatrix() : pop the current matrix from the
current matrix stack.

1. Circles: We have used circles to draw leaves of trees on the both sides of the track in our scenery using GL_POLYGON from the GL/glut library.

GL_POLYGON

Draws a single, convex polygon. Vertices 1 through N define this polygon.

2. Parallelogram: Parallelograms are used to draw trunk of trees using GL_QUADS from the GL/glut library.

GL_QUADS

Treats each group of four vertices as an independent quadrilateral.
Vertices 4 n - 3 , 4 n - 2 , 4 n - 1 , and 4 n define quadrilateral n. N 4 quadrilaterals are drawn.

3. Rectangles: They are used to draw path, lane, car, obstacles and footpath using GL_POLYGON from the GL/glut library.
4. glColor3f() function is used to give different colors to elements of our project from Gl/glut library. Different colors are used to represent different levels.

glRasterPos() : Specify the raster position for pixel operations. The GL maintains a 3D position in window coordinates. This position, called the raster position, is used to position pixel and bitmap write operations. glutBitmapCharacter(): glutBitmapCharacter renders a bitmap character using OpenGL.

GLUT_BITMAP_HELVETICA_18

A 18-point proportional spaced Helvetica font. The exact bitmaps to be used is defined by the standard X.

4. User Defined functions:

Level1 functions:

- display(): This function is used to display all the elements of our project of level1.
- draw_all(): Draws all the elements of our project of level1.
- tree_l(): Used to draw left side trees of our scenery in level1. ● tree_r(): Used to draw right side trees of our scenery level2.

Level2 functions:

- display_level2(): This function is used to display all the elements of our project of level2.
- draw_all_level2(): Draws all the elements of our project of level2.
- tree_l2(): Used to draw left side trees of our scenery in level2.
- tree_r2(): Used to draw right side trees of our scenery in level2.

- obstracule(): Draws obstacle which the car has to bypass to move ahead in level1&2.

Level3 functions:

- display_level3(): This function is used to display all the elements of our project of level3.
- draw_all_level3(): Draws all the elements of our project of level3.
- tree_l3(): Used to draw left side trees of our scenery in level3.
- tree_r3(): Used to draw right side trees of our scenery in level3.

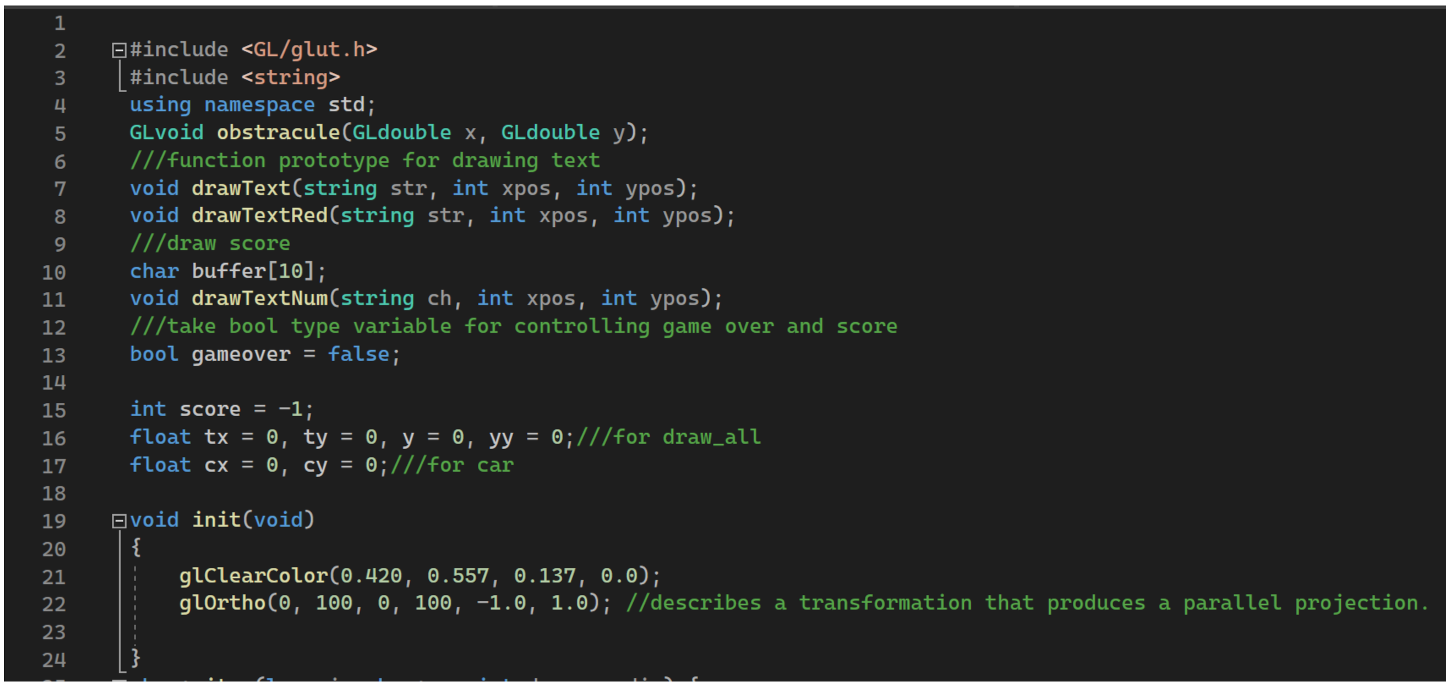
- obstracule3(): Draws obstacle which the car has to bypass to move ahead in level3.

- car(): Displays car which is main element of our project which is moving.
- drawText(): Used to display “Score:”.
- drawTextRed(): Used to display the text “Gameover…” when our car strike the obstacle.
- drawTextNum(): Used to display scores.
- controlAllexceptCar(): This function controls all the functions in our project except the car function.
- spe_key(): Control the movement of the car using keyboard keys.

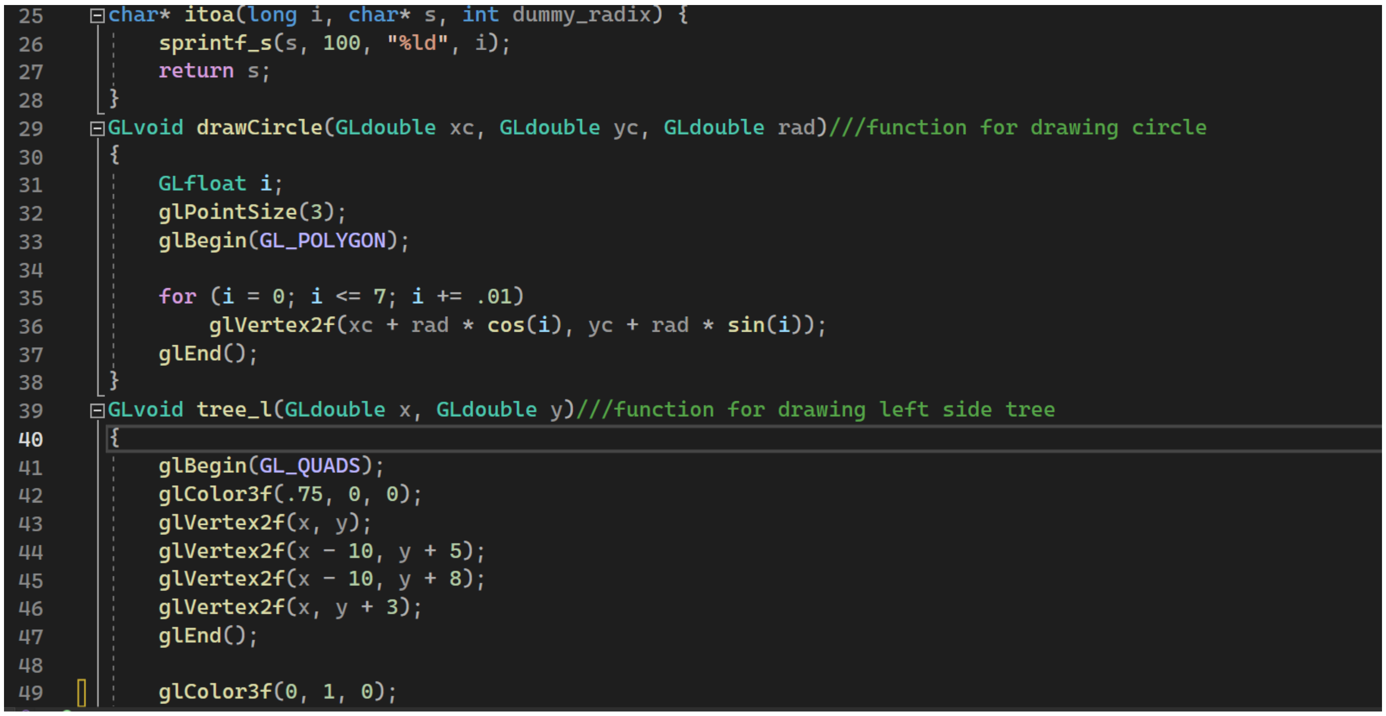
5. Code:

// ConsoleApplication4.cpp : This file contains the 'main' function. Program execution begins and ends there.

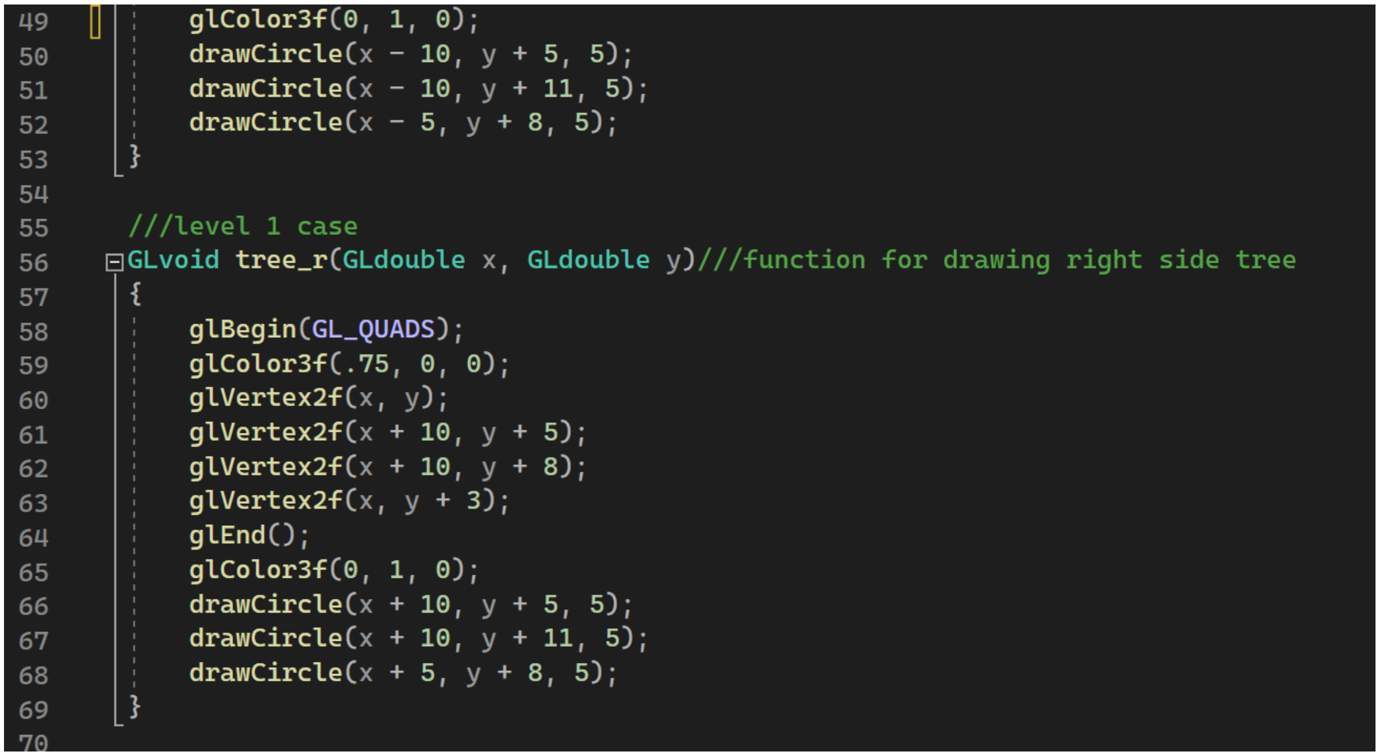
//



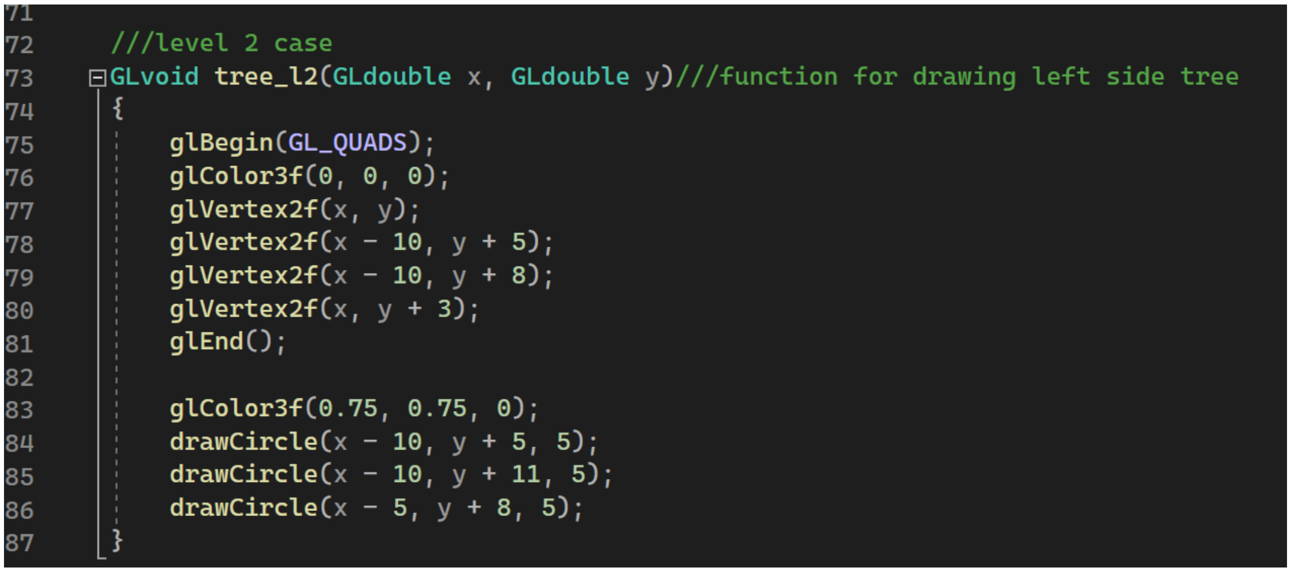
```

#include <GL/glut.h>
#include <string>
using namespace std;
GLvoid obstracule(GLdouble x, GLdouble y);
///function prototype for drawing text
void drawText(string str, int xpos, int ypos);
void drawTextRed(string str, int xpos, int ypos);
///draw score
char buffer[10];
void drawTextNum(string ch, int xpos, int ypos);
///take bool type variable for controlling game over and score
bool gameover = false;

int score = -1;
float tx = 0, ty = 0, y = 0, yy = 0;///for draw_all
float cx = 0, cy = 0;///for car

void init(void)
{
    glClearColor(0.420, 0.557, 0.137, 0.0);
    glOrtho(0, 100, 0, 100, -1.0, 1.0); //describes a transformation that produces a parallel projection.

}
```



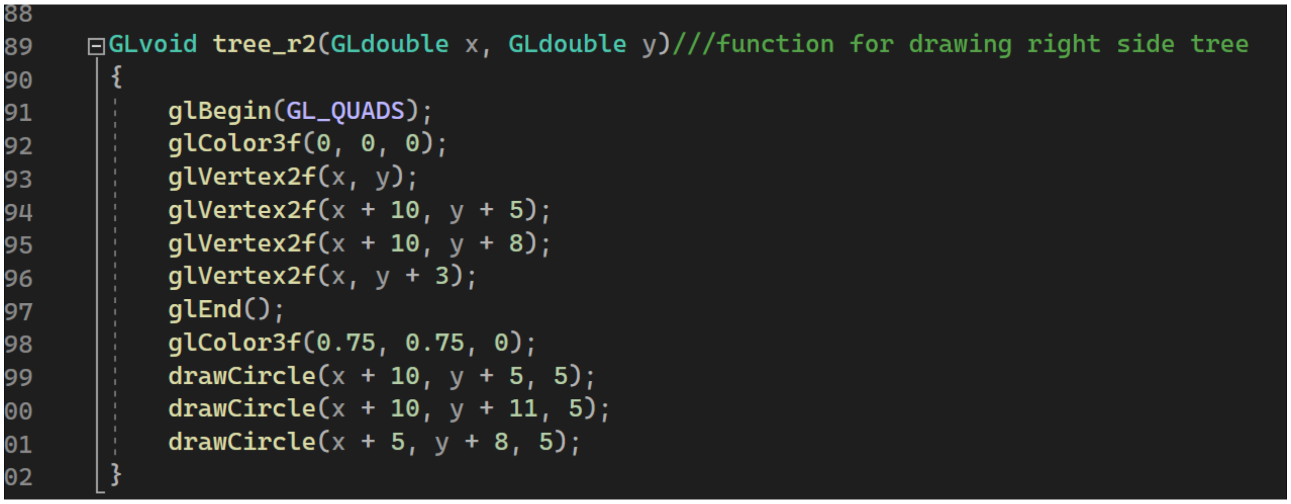
```
char* itoa(long i, char* s, int dummy_radix) {
    sprintf_s(s, 100, "%ld", i);
    return s;
}
GLvoid drawCircle(GLdouble xc, GLdouble yc, GLdouble rad)///function for drawing circle
{
    GLfloat i;
    glPointSize(3);
    glBegin(GL_POLYGON);

    for (i = 0; i <= 7; i += .01)
        glVertex2f(xc + rad * cos(i), yc + rad * sin(i));
    glEnd();
}
GLvoid tree_l(GLdouble x, GLdouble y)///function for drawing left side tree
{
    glBegin(GL_QUADS);
    glColor3f(.75, 0, 0);
    glVertex2f(x, y);
    glVertex2f(x - 10, y + 5);
    glVertex2f(x - 10, y + 8);
    glVertex2f(x, y + 3);
    glEnd();

    glColor3f(0, 1, 0);
```



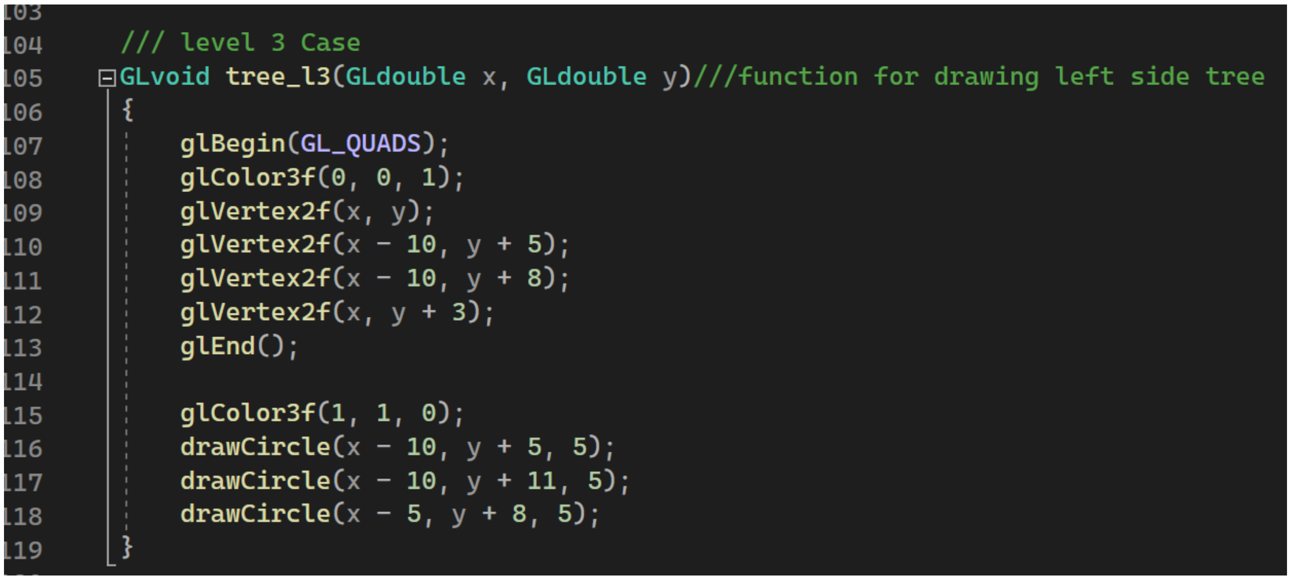
```
    glColor3f(0, 1, 0);
    drawCircle(x - 10, y + 5, 5);
    drawCircle(x - 10, y + 11, 5);
    drawCircle(x - 5, y + 8, 5);
}

///level 1 case
GLvoid tree_r(GLdouble x, GLdouble y)///function for drawing right side tree
{
    glBegin(GL_QUADS);
    glColor3f(.75, 0, 0);
    glVertex2f(x, y);
    glVertex2f(x + 10, y + 5);
    glVertex2f(x + 10, y + 8);
    glVertex2f(x, y + 3);
    glEnd();
    glColor3f(0, 1, 0);
    drawCircle(x + 10, y + 5, 5);
    drawCircle(x + 10, y + 11, 5);
    drawCircle(x + 5, y + 8, 5);
}
```



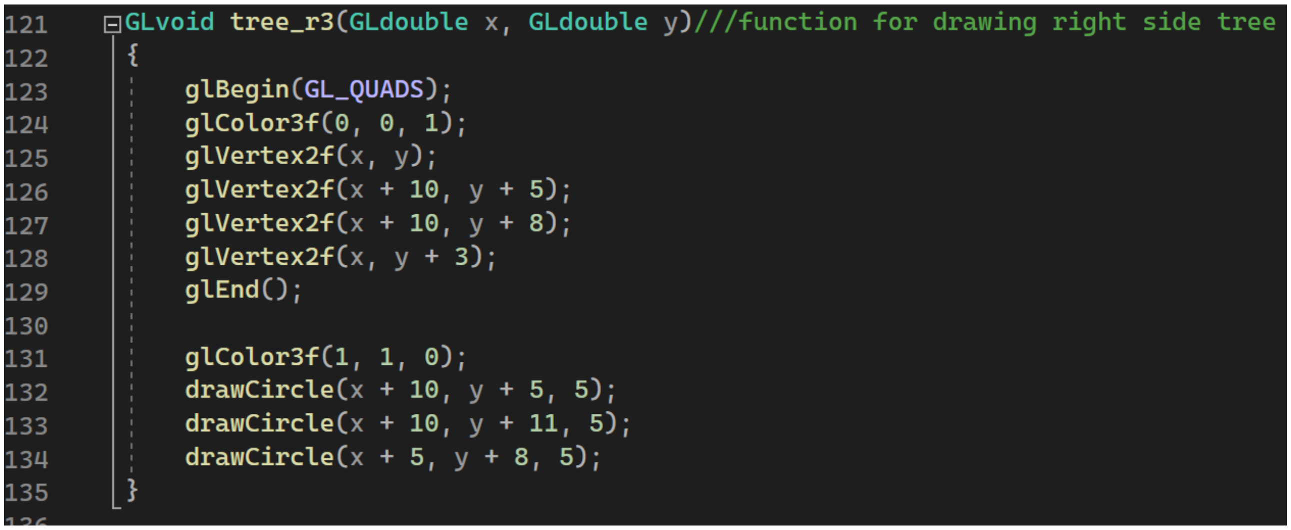
```
///level 2 case
GLvoid tree_l2(GLdouble x, GLdouble y)///function for drawing left side tree
{
    glBegin(GL_QUADS);
    glColor3f(0, 0, 0);
    glVertex2f(x, y);
    glVertex2f(x - 10, y + 5);
    glVertex2f(x - 10, y + 8);
    glVertex2f(x, y + 3);
    glEnd();

    glColor3f(0.75, 0.75, 0);
    drawCircle(x - 10, y + 5, 5);
    drawCircle(x - 10, y + 11, 5);
    drawCircle(x - 5, y + 8, 5);
}
```

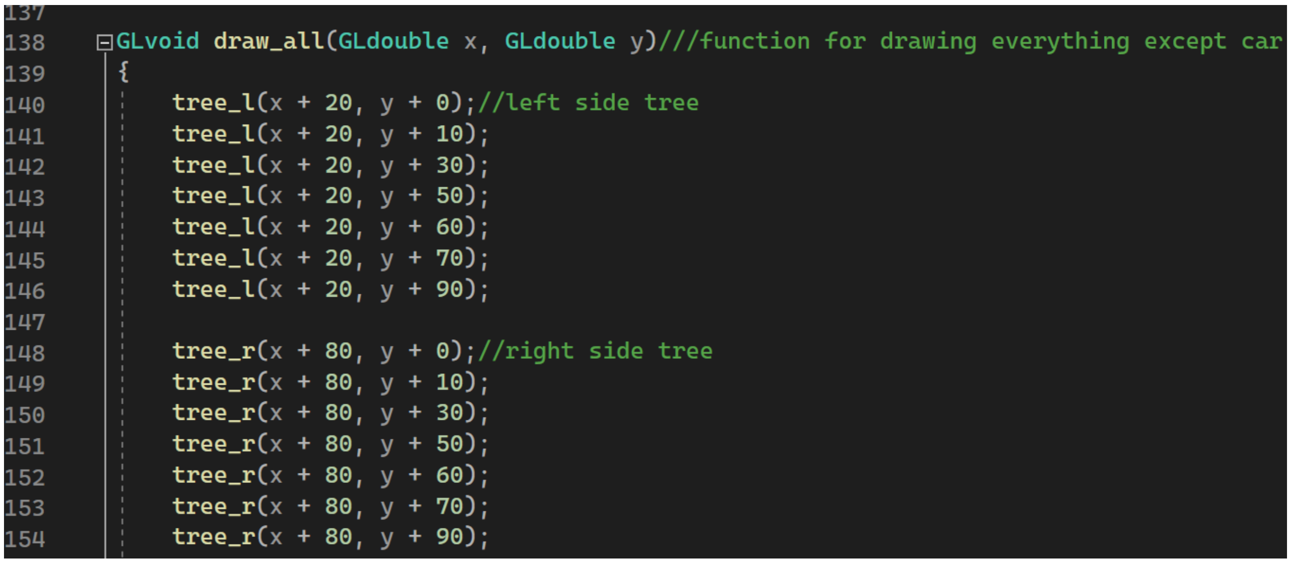


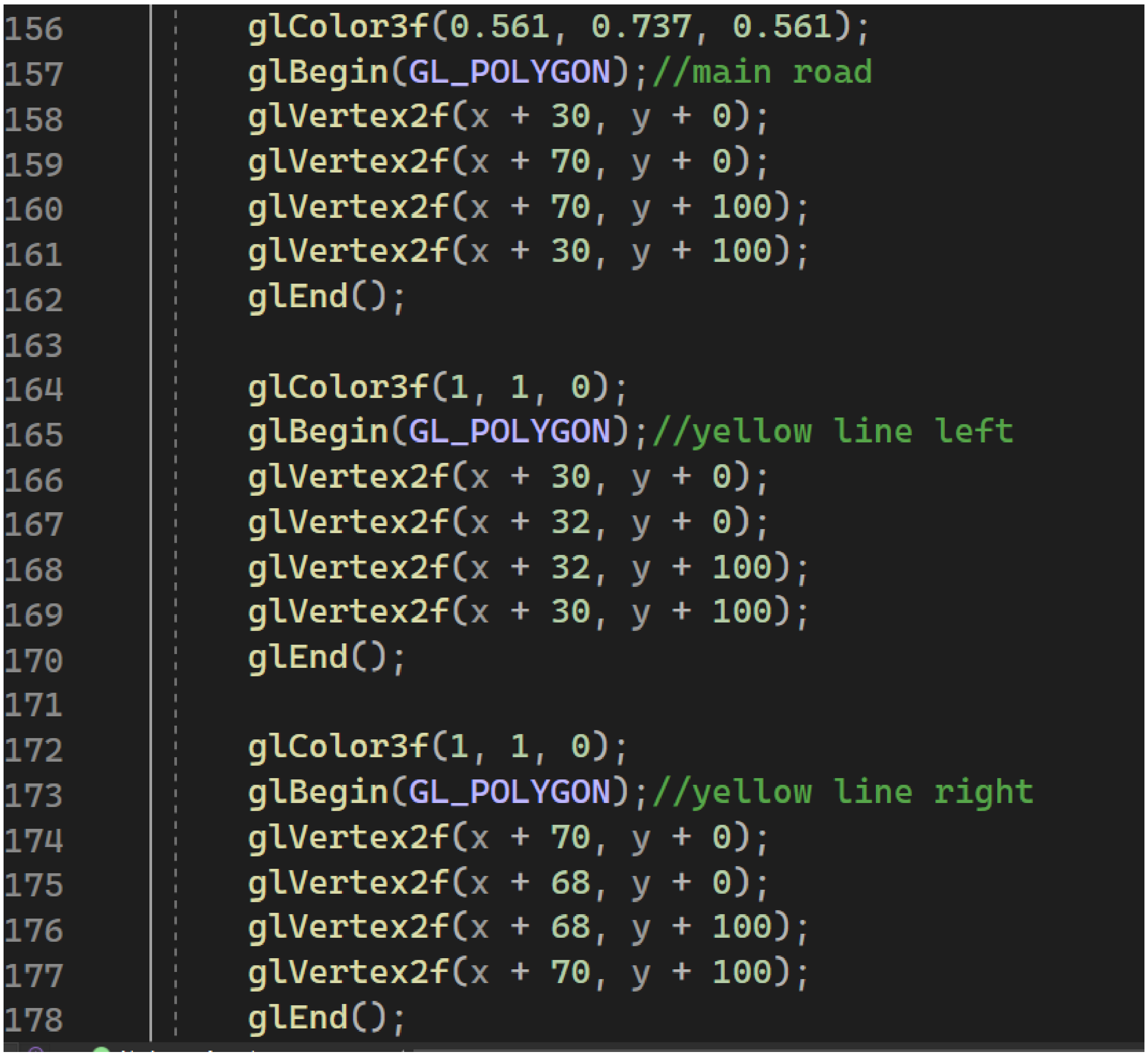
```
GLvoid tree_r2(GLdouble x, GLdouble y)///function for drawing right side tree
{
    glBegin(GL_QUADS);
    glColor3f(0, 0, 0);
    glVertex2f(x, y);
    glVertex2f(x + 10, y + 5);
    glVertex2f(x + 10, y + 8);
    glVertex2f(x, y + 3);
    glEnd();
    glColor3f(0.75, 0.75, 0);
    drawCircle(x + 10, y + 5, 5);
    drawCircle(x + 10, y + 11, 5);
    drawCircle(x + 5, y + 8, 5);
}
```



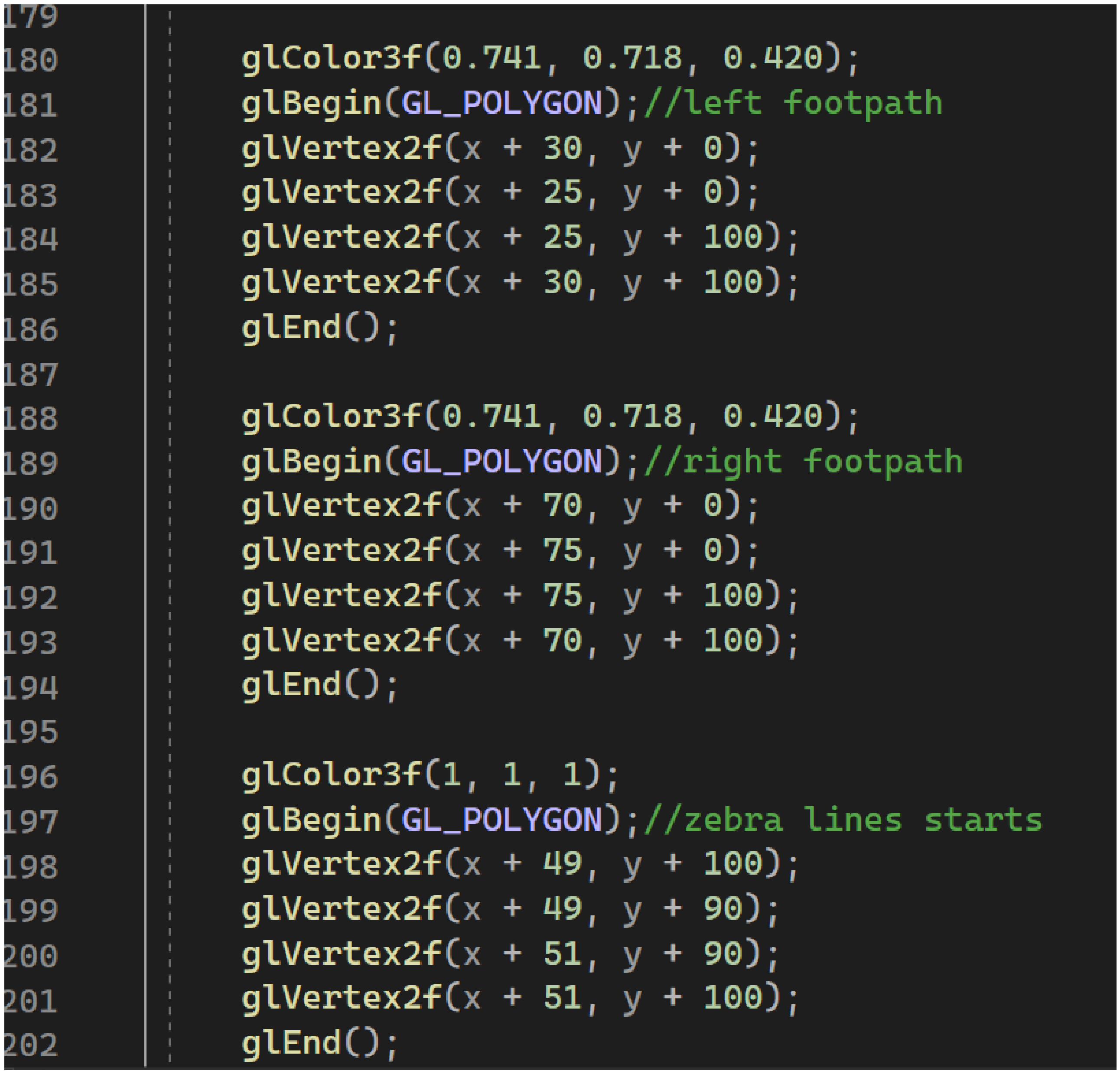
```
/// level 3 Case
GLvoid tree_l3(GLdouble x, GLdouble y)///function for drawing left side tree
{
    glBegin(GL_QUADS);
    glColor3f(0, 0, 1);
    glVertex2f(x, y);
    glVertex2f(x - 10, y + 5);
    glVertex2f(x - 10, y + 8);
    glVertex2f(x, y + 3);
    glEnd();

    glColor3f(1, 1, 0);
    drawCircle(x - 10, y + 5, 5);
    drawCircle(x - 10, y + 11, 5);
    drawCircle(x - 5, y + 8, 5);
}
```

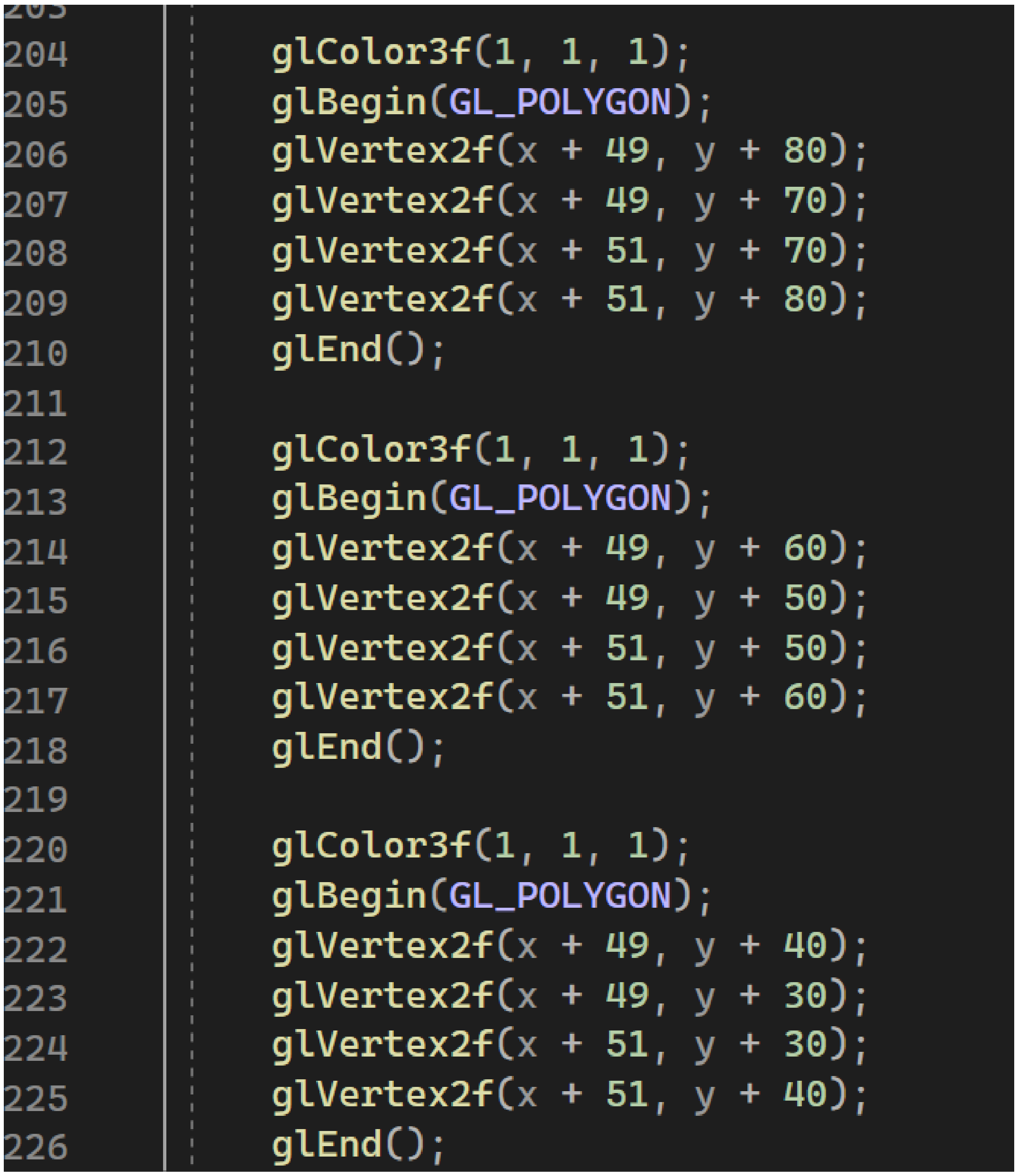

```
GLvoid tree_r3(GLdouble x, GLdouble y)///function for drawing right side tree
{
    glBegin(GL_QUADS);
    glColor3f(0, 0, 1);
    glVertex2f(x, y);
    glVertex2f(x + 10, y + 5);
    glVertex2f(x + 10, y + 8);
    glVertex2f(x, y + 3);
    glEnd();

    glColor3f(1, 1, 0);
    drawCircle(x + 10, y + 5, 5);
    drawCircle(x + 10, y + 11, 5);
    drawCircle(x + 5, y + 8, 5);
}
```


```
GLvoid draw_all(GLdouble x, GLdouble y)///function for drawing everything except car
{
    tree_l(x + 20, y + 0);//left side tree
    tree_l(x + 20, y + 10);
    tree_l(x + 20, y + 30);
    tree_l(x + 20, y + 50);
    tree_l(x + 20, y + 60);
    tree_l(x + 20, y + 70);
    tree_l(x + 20, y + 90);

    tree_r(x + 80, y + 0);//right side tree
    tree_r(x + 80, y + 10);
    tree_r(x + 80, y + 30);
    tree_r(x + 80, y + 50);
    tree_r(x + 80, y + 60);
    tree_r(x + 80, y + 70);
    tree_r(x + 80, y + 90);
```

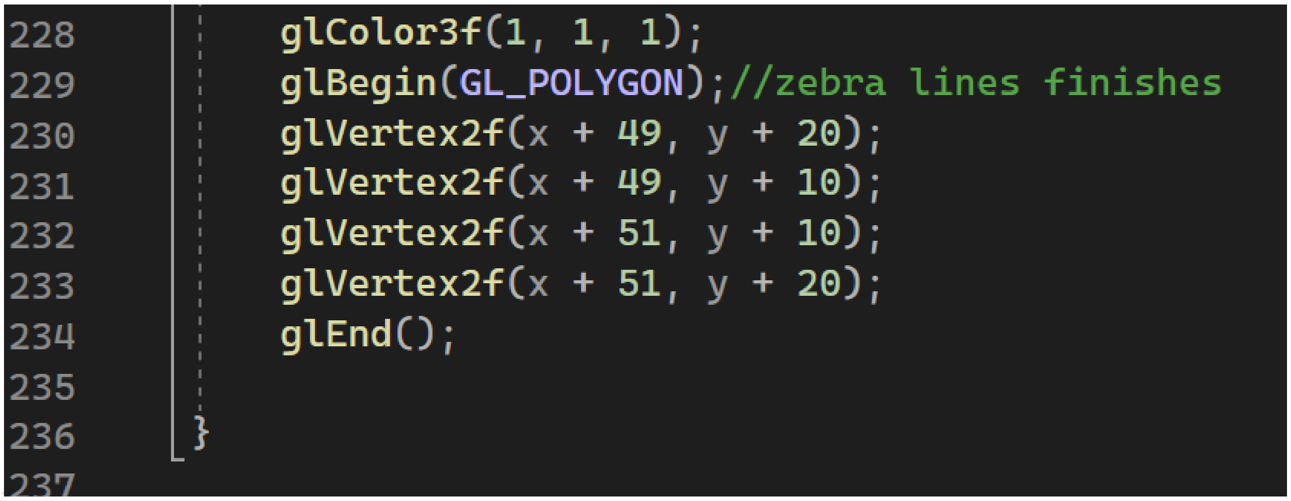


```
    glColor3f(0.561, 0.737, 0.561);
    glBegin(GL_POLYGON);//main road
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 70, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();

    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON);//yellow line left
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 32, y + 0);
    glVertex2f(x + 32, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();

    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON);//yellow line right
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 68, y + 0);
    glVertex2f(x + 68, y + 100);
    glVertex2f(x + 70, y + 100);
    glEnd();
```

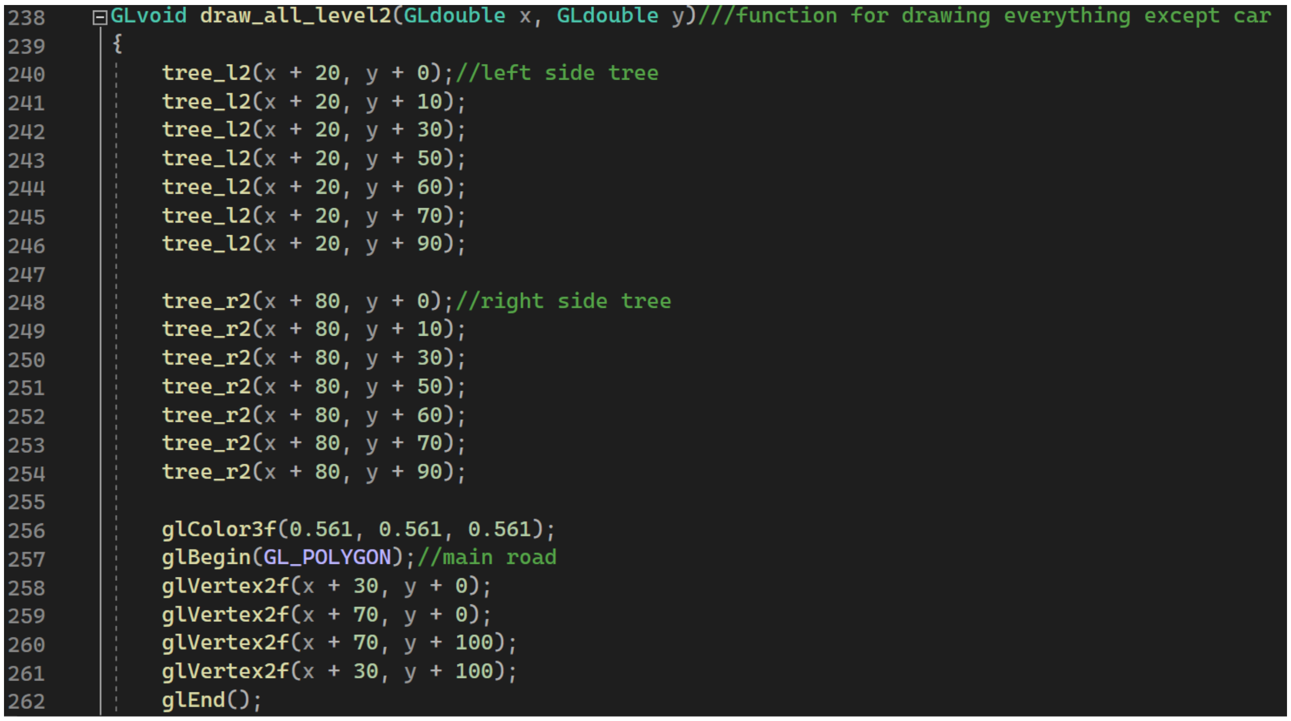

```

    glColor3f(0.741, 0.718, 0.420);
    glBegin(GL_POLYGON);//left footpath
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 25, y + 0);
    glVertex2f(x + 25, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();

    glColor3f(0.741, 0.718, 0.420);
    glBegin(GL_POLYGON);//right footpath
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 75, y + 0);
    glVertex2f(x + 75, y + 100);
    glVertex2f(x + 70, y + 100);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);//zebra lines starts
    glVertex2f(x + 49, y + 100);
    glVertex2f(x + 49, y + 90);
    glVertex2f(x + 51, y + 90);
    glVertex2f(x + 51, y + 100);
    glEnd();
```



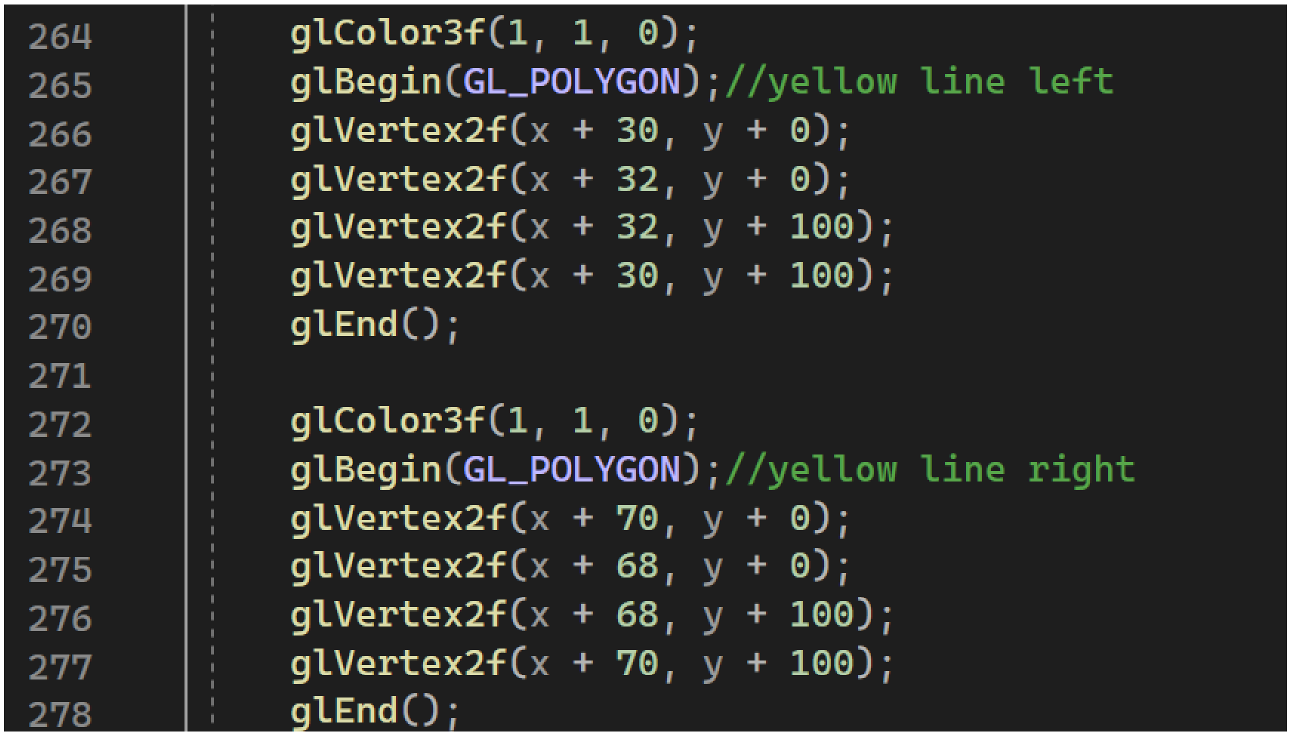
```
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 80);
    glVertex2f(x + 49, y + 70);
    glVertex2f(x + 51, y + 70);
    glVertex2f(x + 51, y + 80);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 60);
    glVertex2f(x + 49, y + 50);
    glVertex2f(x + 51, y + 50);
    glVertex2f(x + 51, y + 60);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 40);
    glVertex2f(x + 49, y + 30);
    glVertex2f(x + 51, y + 30);
    glVertex2f(x + 51, y + 40);
    glEnd();
```



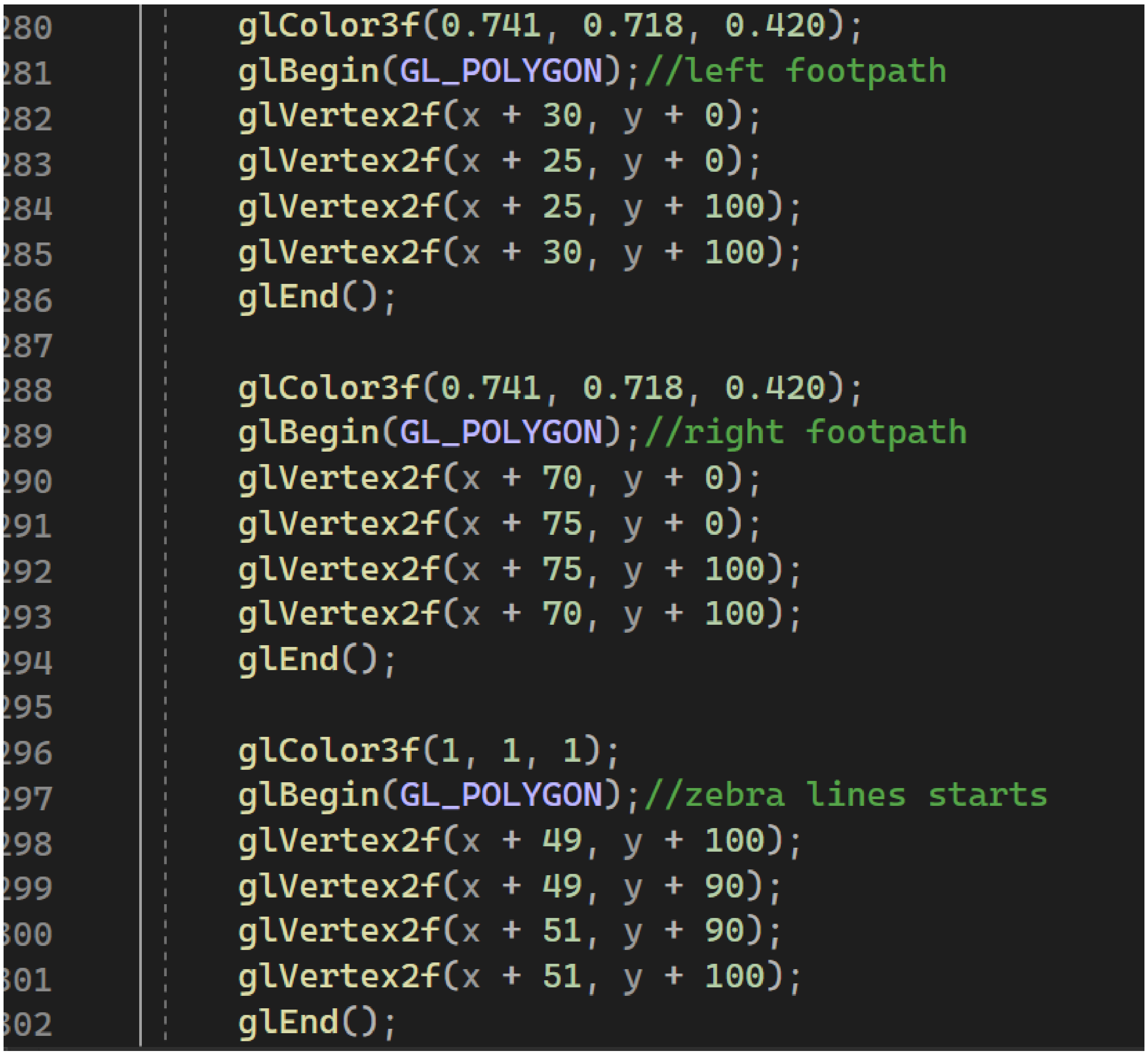
```
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);//zebra lines finishes
    glVertex2f(x + 49, y + 20);
    glVertex2f(x + 49, y + 10);
    glVertex2f(x + 51, y + 10);
    glVertex2f(x + 51, y + 20);
    glEnd();

}
```

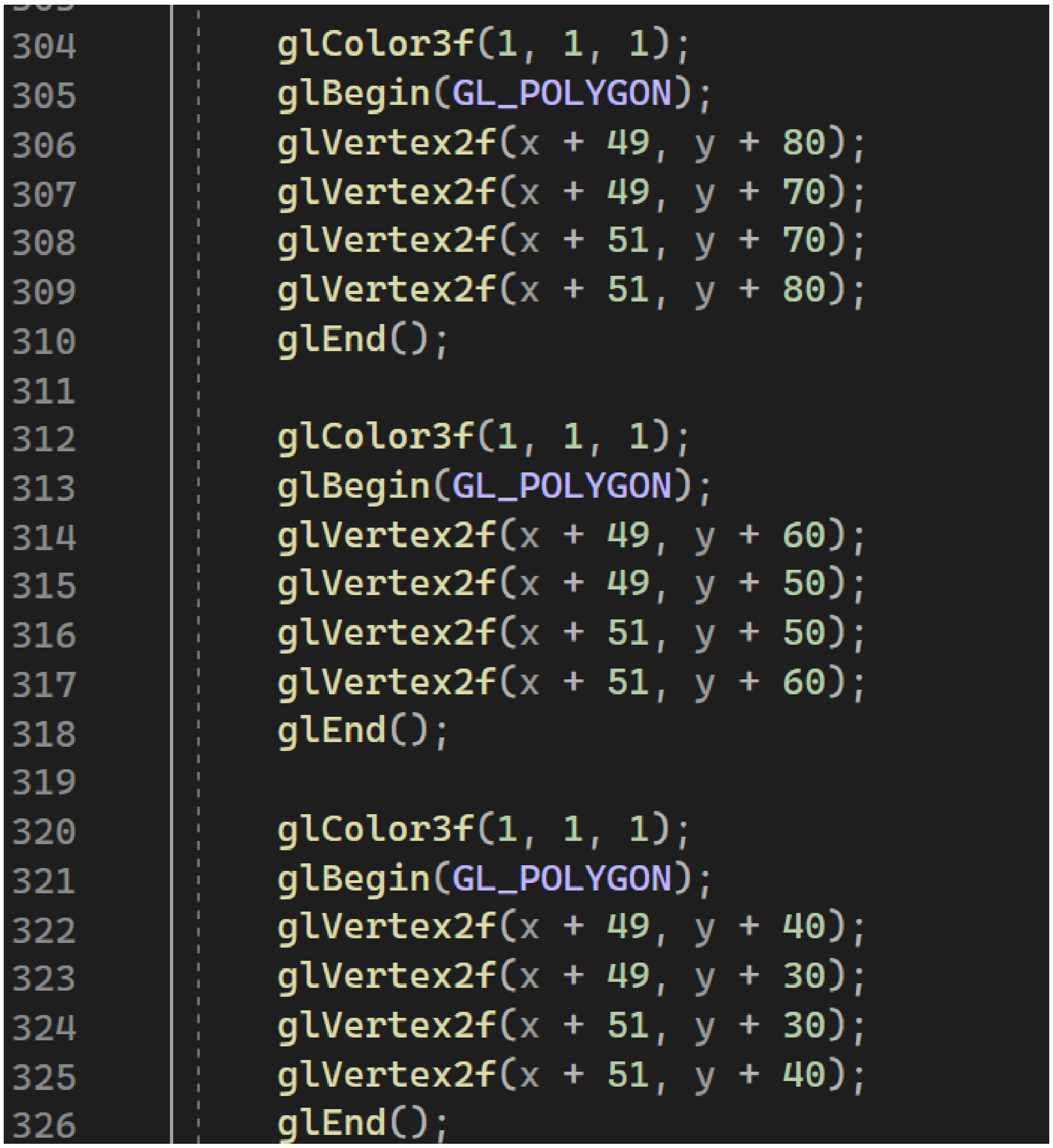


```
GLvoid draw_all_level2(GLdouble x, GLdouble y)///function for drawing everything except car
{
    tree_l2(x + 20, y + 0);//left side tree
    tree_l2(x + 20, y + 10);
    tree_l2(x + 20, y + 30);
    tree_l2(x + 20, y + 50);
    tree_l2(x + 20, y + 60);
    tree_l2(x + 20, y + 70);
    tree_l2(x + 20, y + 90);

    tree_r2(x + 80, y + 0);//right side tree
    tree_r2(x + 80, y + 10);
    tree_r2(x + 80, y + 30);
    tree_r2(x + 80, y + 50);
    tree_r2(x + 80, y + 60);
    tree_r2(x + 80, y + 70);
    tree_r2(x + 80, y + 90);

    glColor3f(0.561, 0.561, 0.561);
    glBegin(GL_POLYGON);//main road
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 70, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();
```

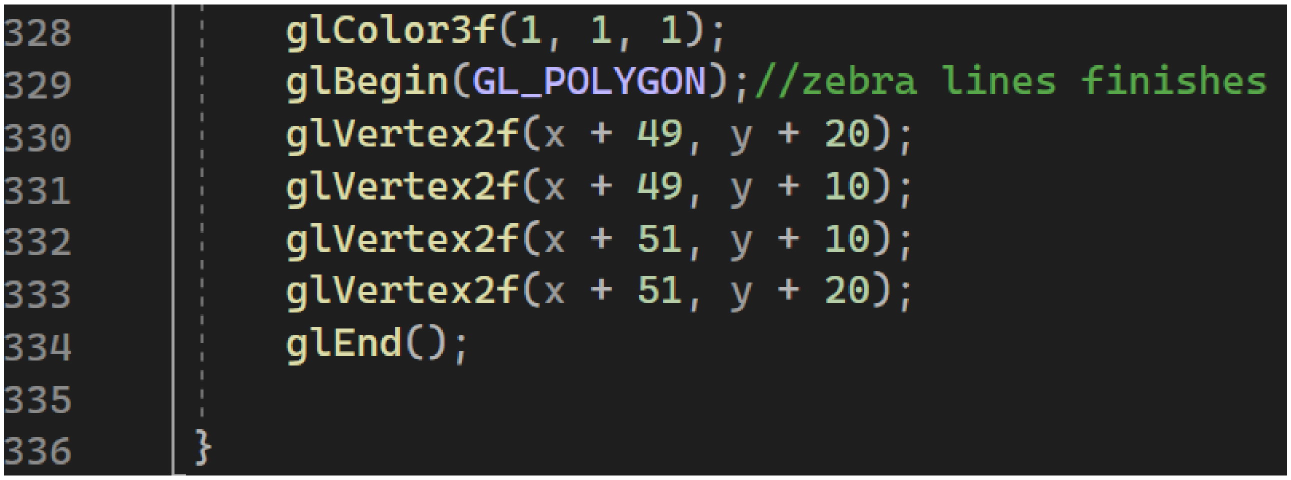


```
    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON);//yellow line left
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 32, y + 0);
    glVertex2f(x + 32, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();

    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON);//yellow line right
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 68, y + 0);
    glVertex2f(x + 68, y + 100);
    glVertex2f(x + 70, y + 100);
    glEnd();
```

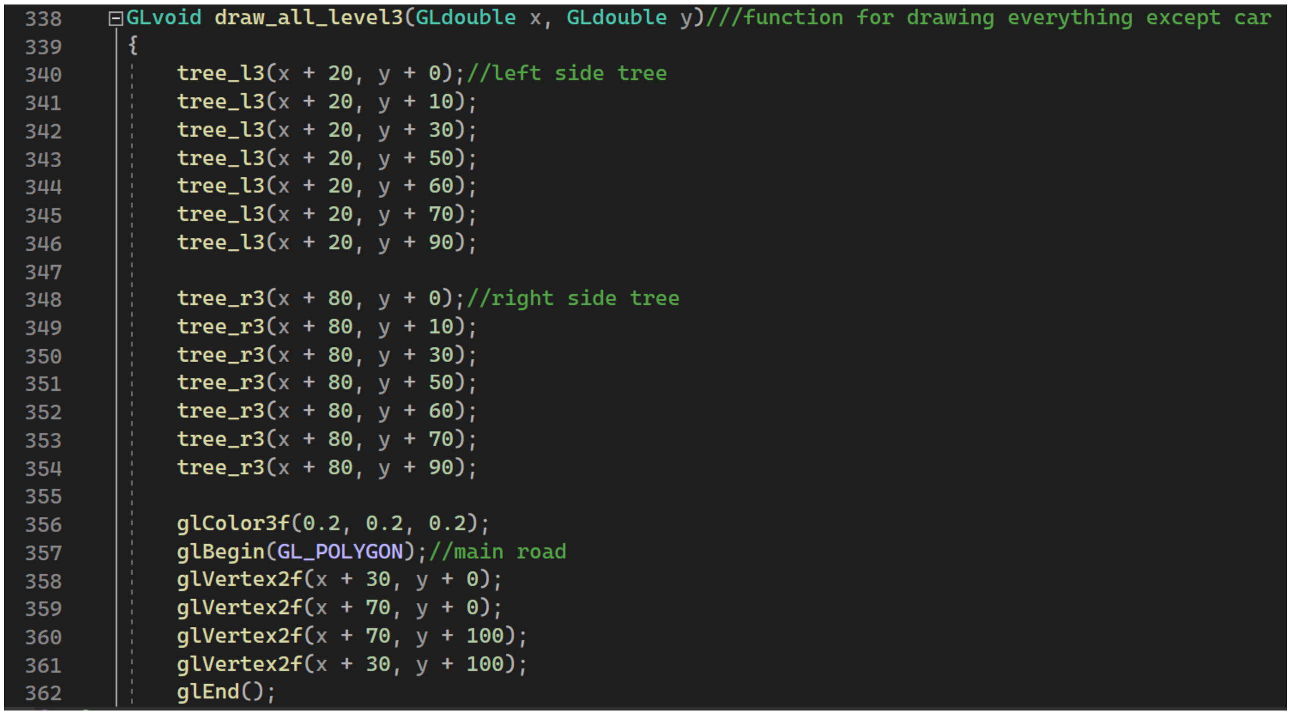

```
    glColor3f(0.741, 0.718, 0.420);
    glBegin(GL_POLYGON);//left footpath
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 25, y + 0);
    glVertex2f(x + 25, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();

    glColor3f(0.741, 0.718, 0.420);
    glBegin(GL_POLYGON);//right footpath
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 75, y + 0);
    glVertex2f(x + 75, y + 100);
    glVertex2f(x + 70, y + 100);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);//zebra lines starts
    glVertex2f(x + 49, y + 100);
    glVertex2f(x + 49, y + 90);
    glVertex2f(x + 51, y + 90);
    glVertex2f(x + 51, y + 100);
    glEnd();
```

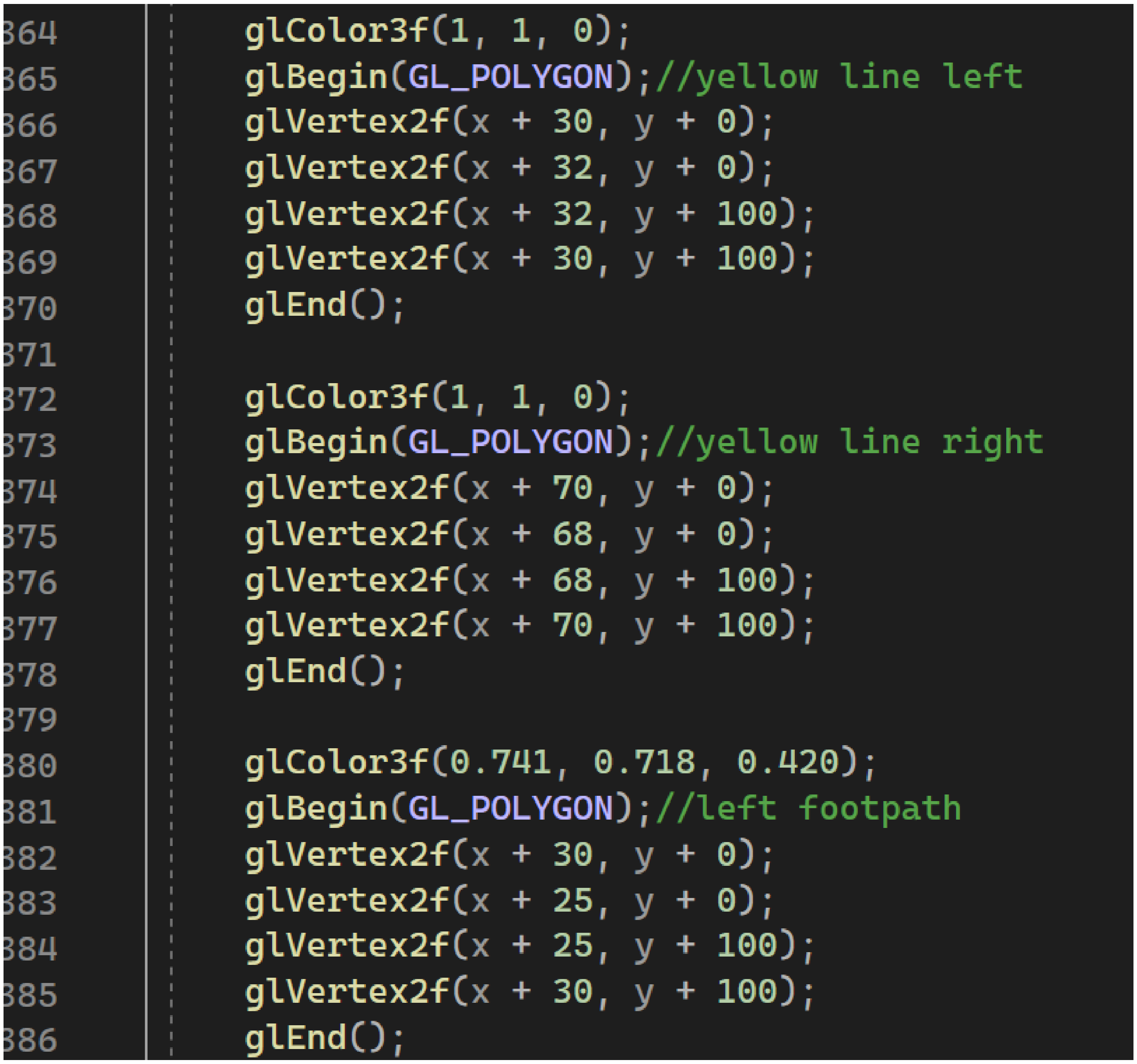


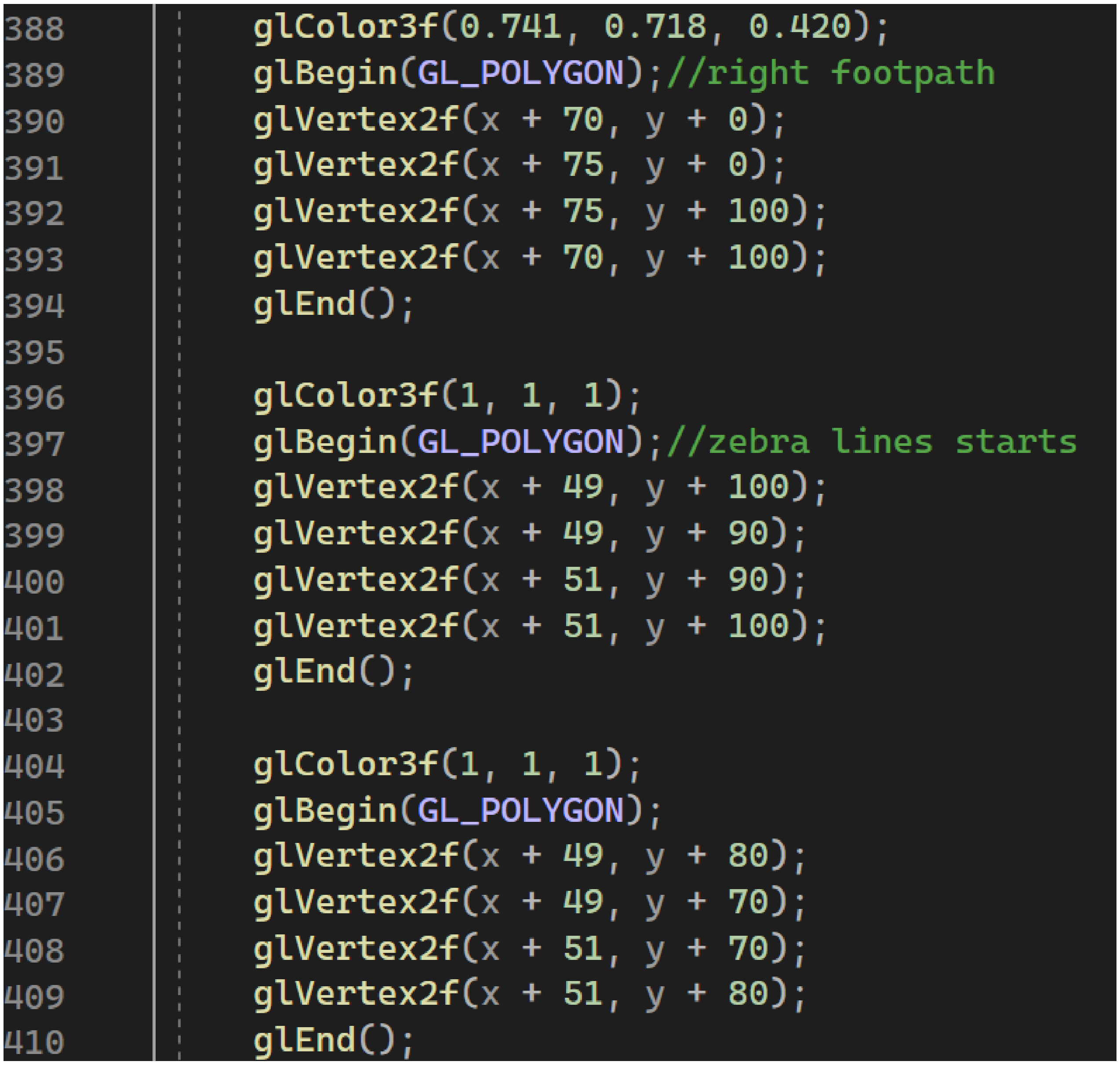
```
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 80);
    glVertex2f(x + 49, y + 70);
    glVertex2f(x + 51, y + 70);
    glVertex2f(x + 51, y + 80);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 60);
    glVertex2f(x + 49, y + 50);
    glVertex2f(x + 51, y + 50);
    glVertex2f(x + 51, y + 60);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 40);
    glVertex2f(x + 49, y + 30);
    glVertex2f(x + 51, y + 30);
    glVertex2f(x + 51, y + 40);
    glEnd();
```



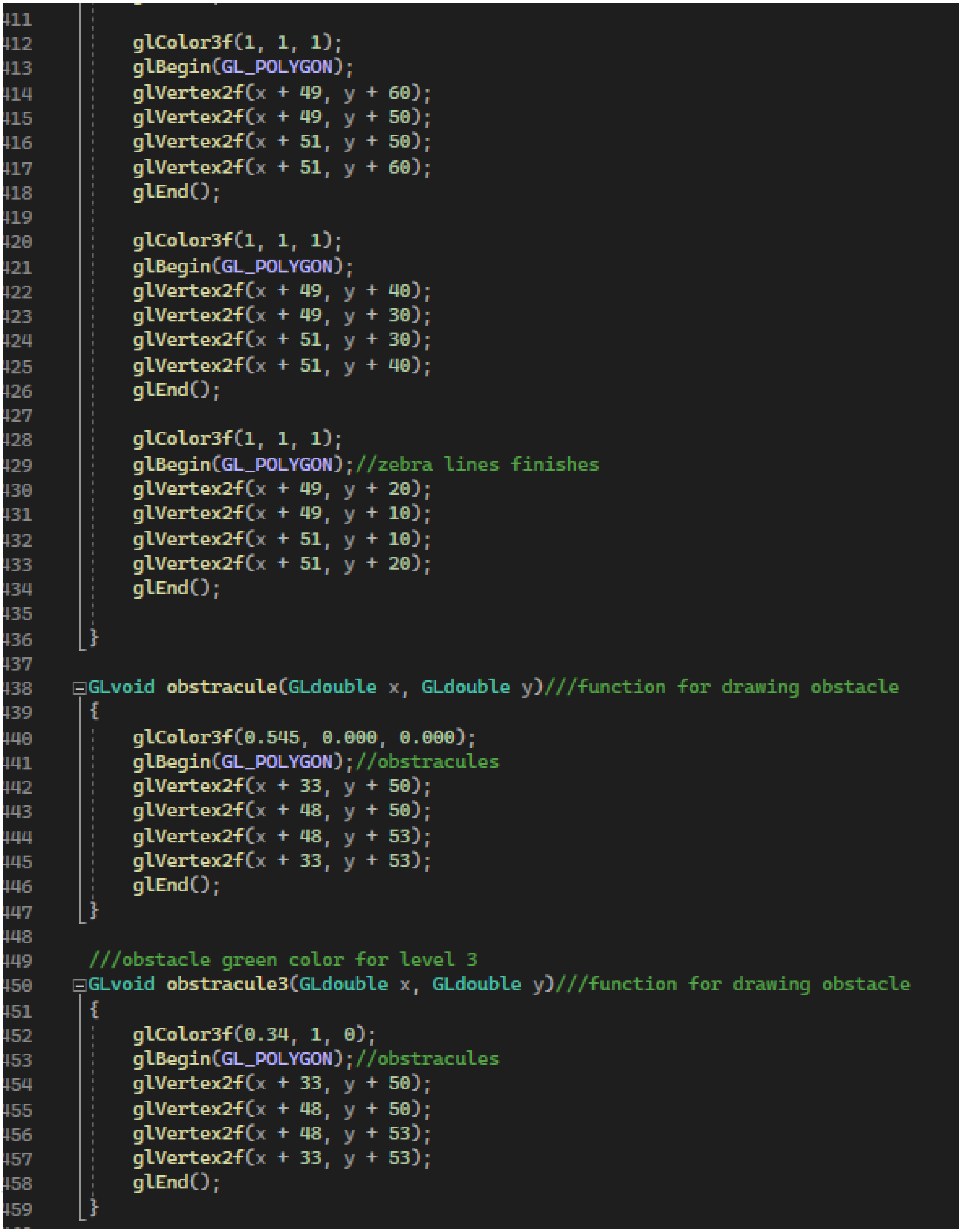
```
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);//zebra lines finishes
    glVertex2f(x + 49, y + 20);
    glVertex2f(x + 49, y + 10);
    glVertex2f(x + 51, y + 10);
    glVertex2f(x + 51, y + 20);
    glEnd();

}
```

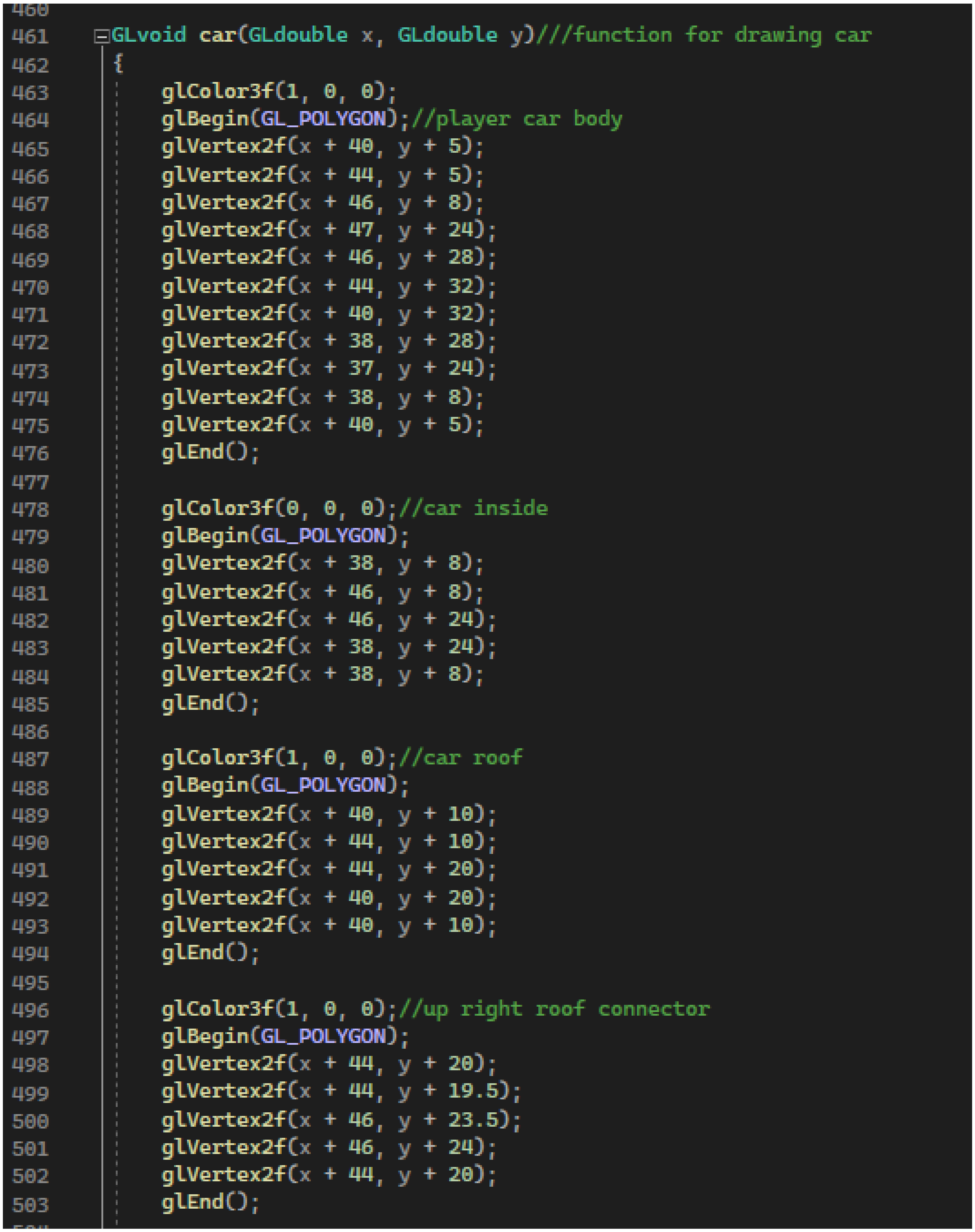

```
GLvoid draw_all_level3(GLdouble x, GLdouble y)///function for drawing everything except car
{
    tree_l3(x + 20, y + 0);//left side tree
    tree_l3(x + 20, y + 10);
    tree_l3(x + 20, y + 30);
    tree_l3(x + 20, y + 50);
    tree_l3(x + 20, y + 60);
    tree_l3(x + 20, y + 70);
    tree_l3(x + 20, y + 90);

    tree_r3(x + 80, y + 0);//right side tree
    tree_r3(x + 80, y + 10);
    tree_r3(x + 80, y + 30);
    tree_r3(x + 80, y + 50);
    tree_r3(x + 80, y + 60);
    tree_r3(x + 80, y + 70);
    tree_r3(x + 80, y + 90);

    glColor3f(0.2, 0.2, 0.2);
    glBegin(GL_POLYGON);//main road
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 70, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();
```


```
    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON);//yellow line left
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 32, y + 0);
    glVertex2f(x + 32, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();

    glColor3f(1, 1, 0);
    glBegin(GL_POLYGON);//yellow line right
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 68, y + 0);
    glVertex2f(x + 68, y + 100);
    glVertex2f(x + 70, y + 100);
    glEnd();

    glColor3f(0.741, 0.718, 0.420);
    glBegin(GL_POLYGON);//left footpath
    glVertex2f(x + 30, y + 0);
    glVertex2f(x + 25, y + 0);
    glVertex2f(x + 25, y + 100);
    glVertex2f(x + 30, y + 100);
    glEnd();
```

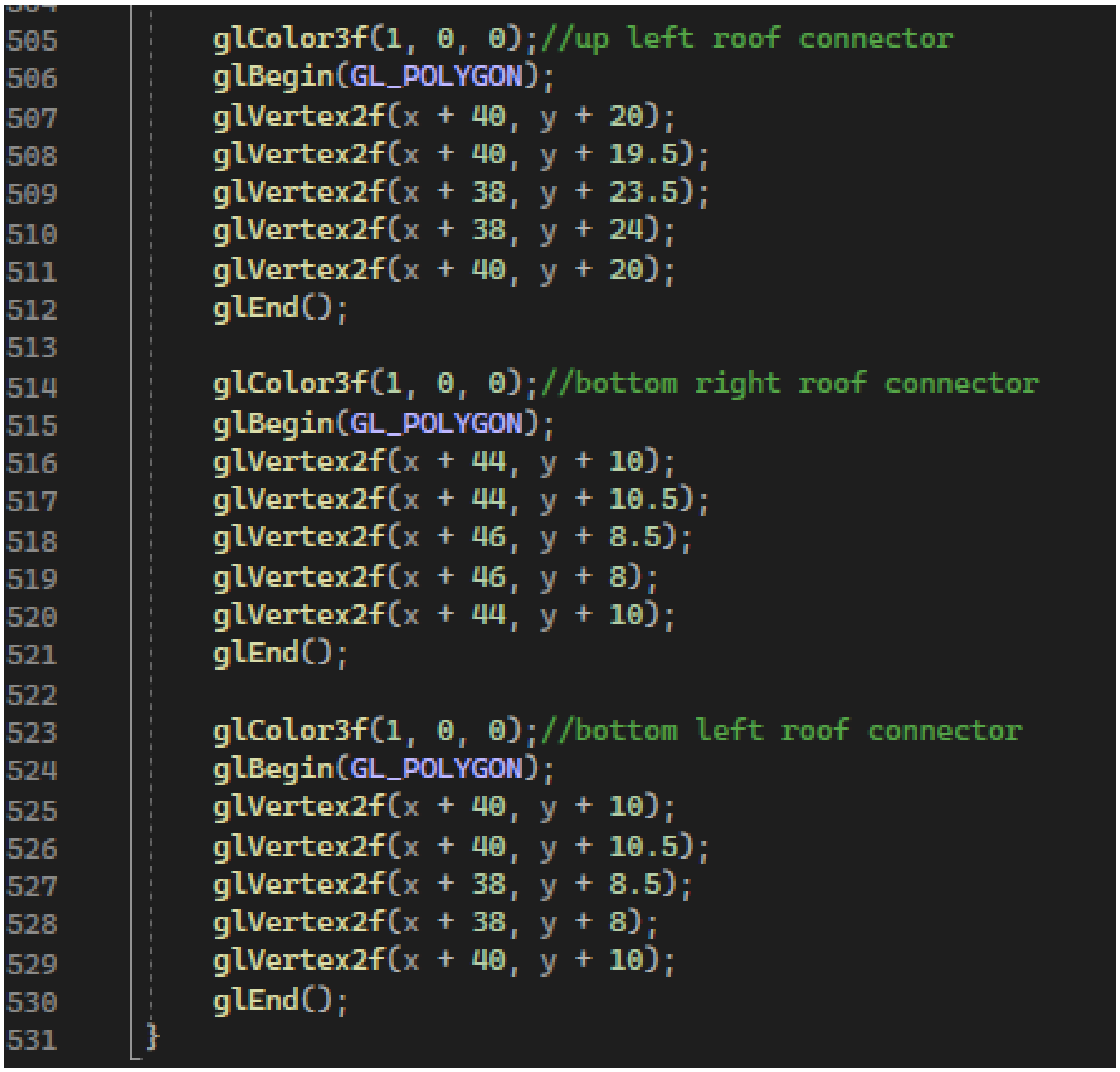


```
    glColor3f(0.741, 0.718, 0.420);
    glBegin(GL_POLYGON);//right footpath
    glVertex2f(x + 70, y + 0);
    glVertex2f(x + 75, y + 0);
    glVertex2f(x + 75, y + 100);
    glVertex2f(x + 70, y + 100);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);//zebra lines starts
    glVertex2f(x + 49, y + 100);
    glVertex2f(x + 49, y + 90);
    glVertex2f(x + 51, y + 90);
    glVertex2f(x + 51, y + 100);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 80);
    glVertex2f(x + 49, y + 70);
    glVertex2f(x + 51, y + 70);
    glVertex2f(x + 51, y + 80);
    glEnd();
```

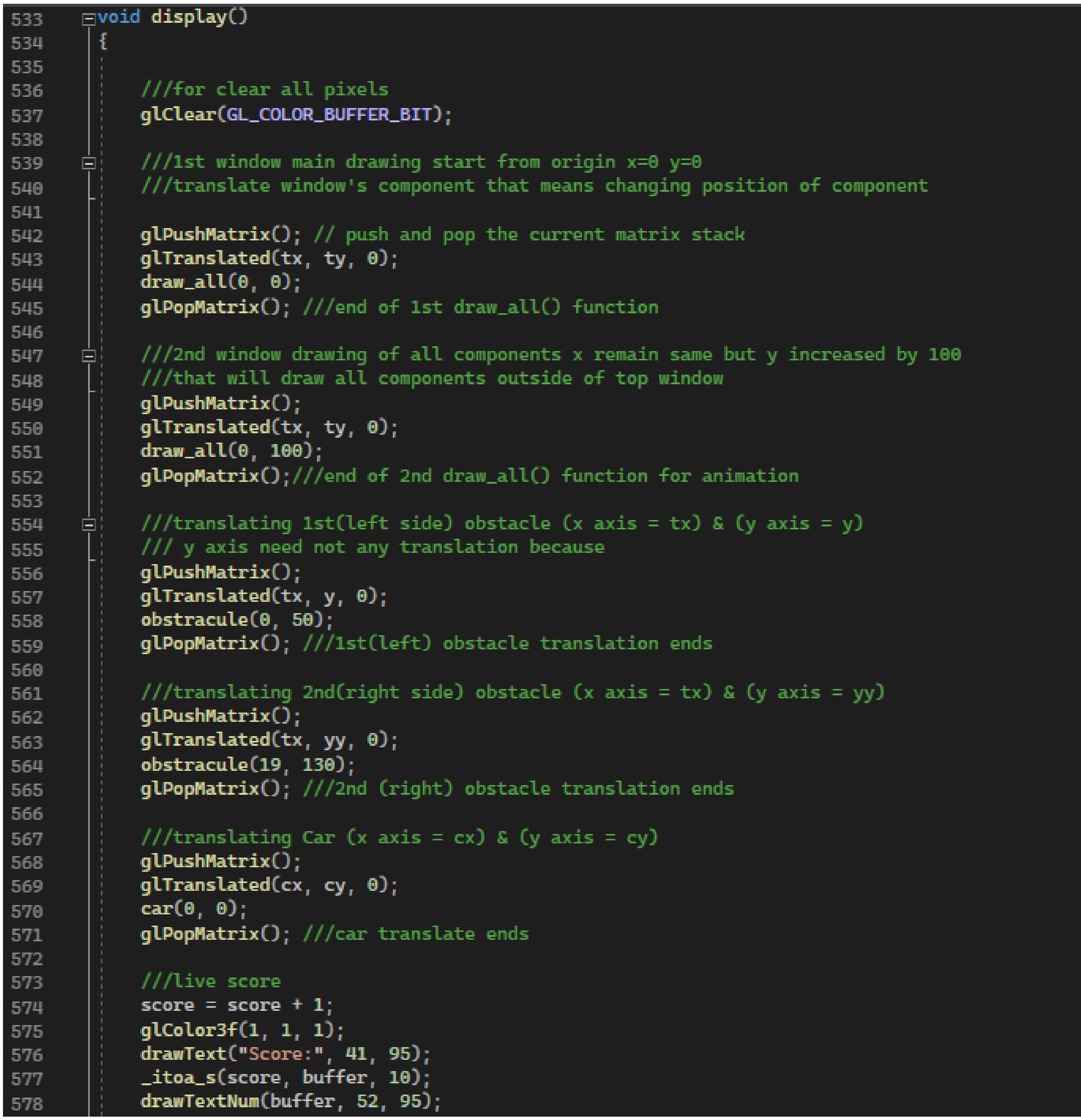

```
    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 60);
    glVertex2f(x + 49, y + 50);
    glVertex2f(x + 51, y + 50);
    glVertex2f(x + 51, y + 60);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);
    glVertex2f(x + 49, y + 40);
    glVertex2f(x + 49, y + 30);
    glVertex2f(x + 51, y + 30);
    glVertex2f(x + 51, y + 40);
    glEnd();

    glColor3f(1, 1, 1);
    glBegin(GL_POLYGON);//zebra lines finishes
    glVertex2f(x + 49, y + 20);
    glVertex2f(x + 49, y + 10);
    glVertex2f(x + 51, y + 10);
    glVertex2f(x + 51, y + 20);
    glEnd();

}

GLvoid obstracule(GLdouble x, GLdouble y)///function for drawing obstacle
{
    glColor3f(0.545, 0.000, 0.000);
    glBegin(GL_POLYGON);//obstracules
    glVertex2f(x + 33, y + 50);
    glVertex2f(x + 48, y + 50);
    glVertex2f(x + 48, y + 53);
    glVertex2f(x + 33, y + 53);
    glEnd();
}

///obstacle green color for level 3
GLvoid obstracule3(GLdouble x, GLdouble y)///function for drawing obstacle
{
    glColor3f(0.34, 1, 0);
    glBegin(GL_POLYGON);//obstracules
    glVertex2f(x + 33, y + 50);
    glVertex2f(x + 48, y + 50);
    glVertex2f(x + 48, y + 53);
    glVertex2f(x + 33, y + 53);
    glEnd();
}
```



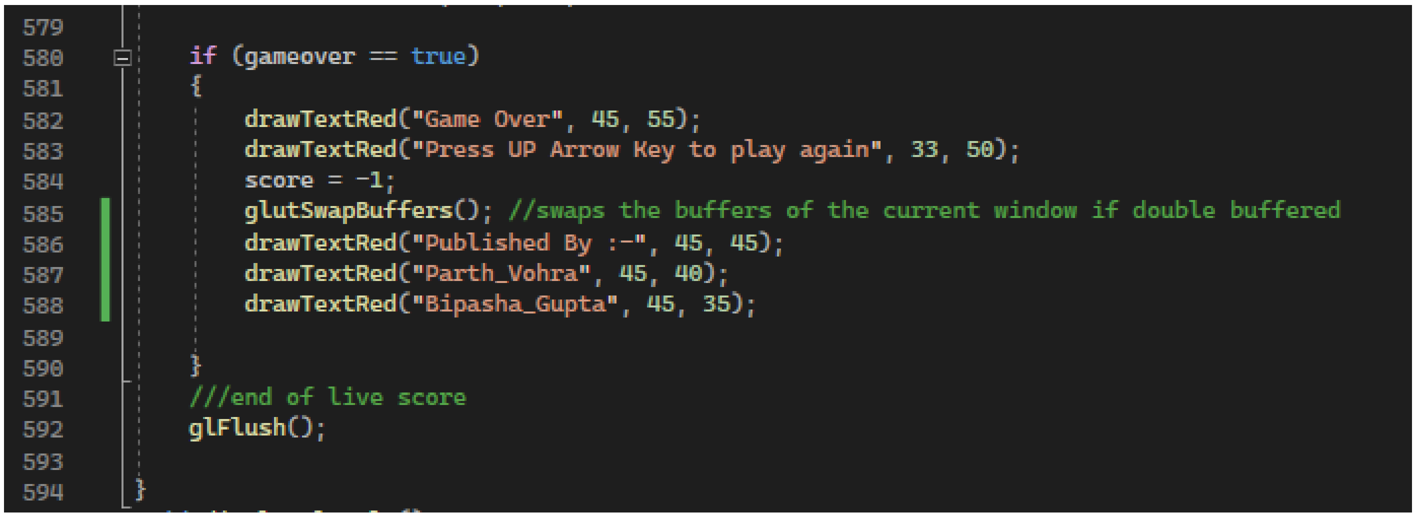
```
GLvoid car(GLdouble x, GLdouble y)///function for drawing car
{
    glColor3f(1, 0, 0);
    glBegin(GL_POLYGON);//player car body
    glVertex2f(x + 40, y + 5);
    glVertex2f(x + 44, y + 5);
    glVertex2f(x + 46, y + 8);
    glVertex2f(x + 47, y + 24);
    glVertex2f(x + 46, y + 28);
    glVertex2f(x + 44, y + 32);
    glVertex2f(x + 40, y + 32);
    glVertex2f(x + 38, y + 28);
    glVertex2f(x + 37, y + 24);
    glVertex2f(x + 38, y + 8);
    glVertex2f(x + 40, y + 5);
    glEnd();

    glColor3f(0, 0, 0);//car inside
    glBegin(GL_POLYGON);
    glVertex2f(x + 38, y + 8);
    glVertex2f(x + 46, y + 8);
    glVertex2f(x + 46, y + 24);
    glVertex2f(x + 38, y + 24);
    glVertex2f(x + 38, y + 8);
    glEnd();

    glColor3f(1, 0, 0);//car roof
    glBegin(GL_POLYGON);
    glVertex2f(x + 40, y + 10);
    glVertex2f(x + 44, y + 10);
    glVertex2f(x + 44, y + 20);
    glVertex2f(x + 40, y + 20);
    glVertex2f(x + 40, y + 10);
    glEnd();

    glColor3f(1, 0, 0);//up right roof connector
    glBegin(GL_POLYGON);
    glVertex2f(x + 44, y + 20);
    glVertex2f(x + 44, y + 19.5);
    glVertex2f(x + 46, y + 23.5);
    glVertex2f(x + 46, y + 24);
    glVertex2f(x + 44, y + 20);
    glEnd();
```



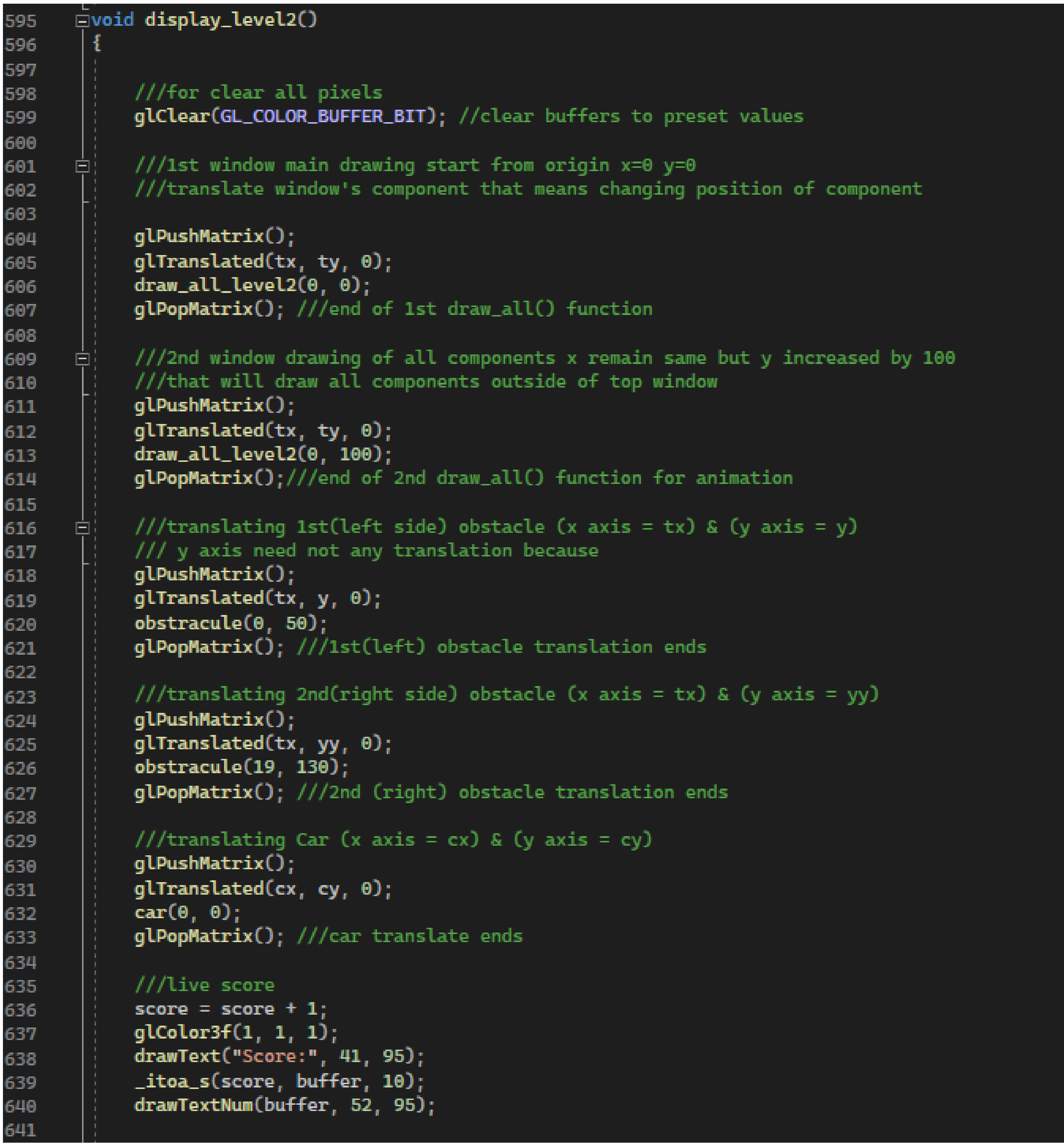
```
    glColor3f(1, 0, 0);//up left roof connector
    glBegin(GL_POLYGON);
    glVertex2f(x + 40, y + 20);
    glVertex2f(x + 40, y + 19.5);
    glVertex2f(x + 38, y + 23.5);
    glVertex2f(x + 38, y + 24);
    glVertex2f(x + 40, y + 20);
    glEnd();

    glColor3f(1, 0, 0);//bottom right roof connector
    glBegin(GL_POLYGON);
    glVertex2f(x + 44, y + 10);
    glVertex2f(x + 44, y + 10.5);
    glVertex2f(x + 46, y + 8.5);
    glVertex2f(x + 46, y + 8);
    glVertex2f(x + 44, y + 10);
    glEnd();

    glColor3f(1, 0, 0);//bottom left roof connector
    glBegin(GL_POLYGON);
    glVertex2f(x + 40, y + 10);
    glVertex2f(x + 40, y + 10.5);
    glVertex2f(x + 38, y + 8.5);
    glVertex2f(x + 38, y + 8);
    glVertex2f(x + 40, y + 10);
    glEnd();
}
```



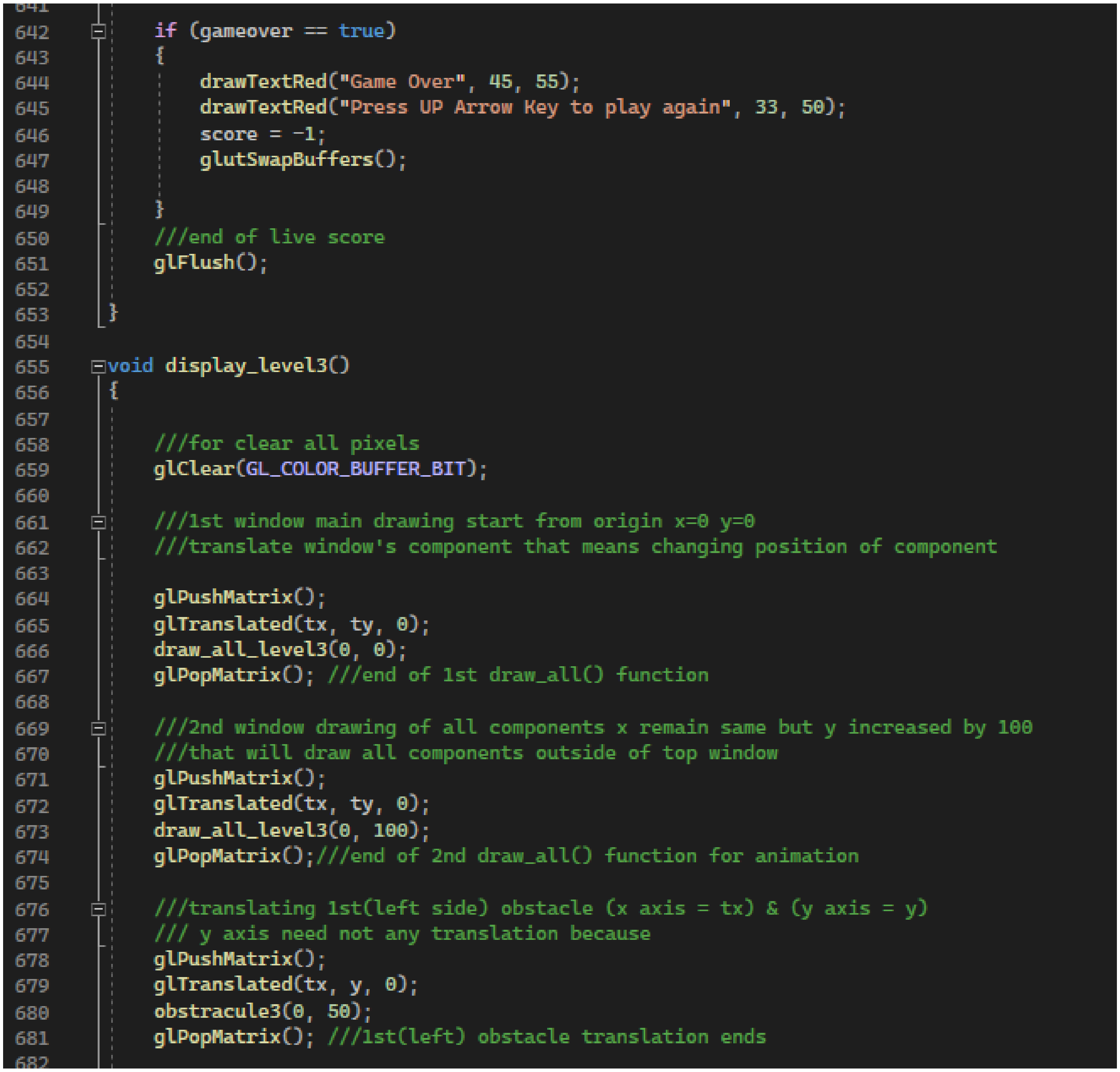
```
void display()
{

    ///for clear all pixels
    glClear(GL_COLOR_BUFFER_BIT);

    ///1st window main drawing start from origin x=0 y=0
    ///translate window's component that means changing position of component

    glPushMatrix(); // push and pop the current matrix stack
    glTranslated(tx, ty, 0);
    draw_all(0, 0);
    glPopMatrix(); ///end of 1st draw_all() function

    ///2nd window drawing of all components x remain same but y increased by 100
    ///that will draw all components outside of top window
    glPushMatrix();
    glTranslated(tx, ty, 0);
    draw_all(0, 100);
    glPopMatrix();///end of 2nd draw_all() function for animation

    ///translating 1st(left side) obstacle (x axis = tx) & (y axis = y)
    /// y axis need not any translation because
    glPushMatrix();
    glTranslated(tx, y, 0);
    obstracule(0, 50);
    glPopMatrix(); ///1st(left) obstacle translation ends

    ///translating 2nd(right side) obstacle (x axis = tx) & (y axis = yy)
    glPushMatrix();
    glTranslated(tx, yy, 0);
    obstracule(19, 130);
    glPopMatrix(); ///2nd (right) obstacle translation ends

    ///translating Car (x axis = cx) & (y axis = cy)
    glPushMatrix();
    glTranslated(cx, cy, 0);
    car(0, 0);
    glPopMatrix(); ///car translate ends

    ///live score
    score = score + 1;
    glColor3f(1, 1, 1);
    drawText("Score:", 41, 95);
    _itoa_s(score, buffer, 10);
    drawTextNum(buffer, 52, 95);
```



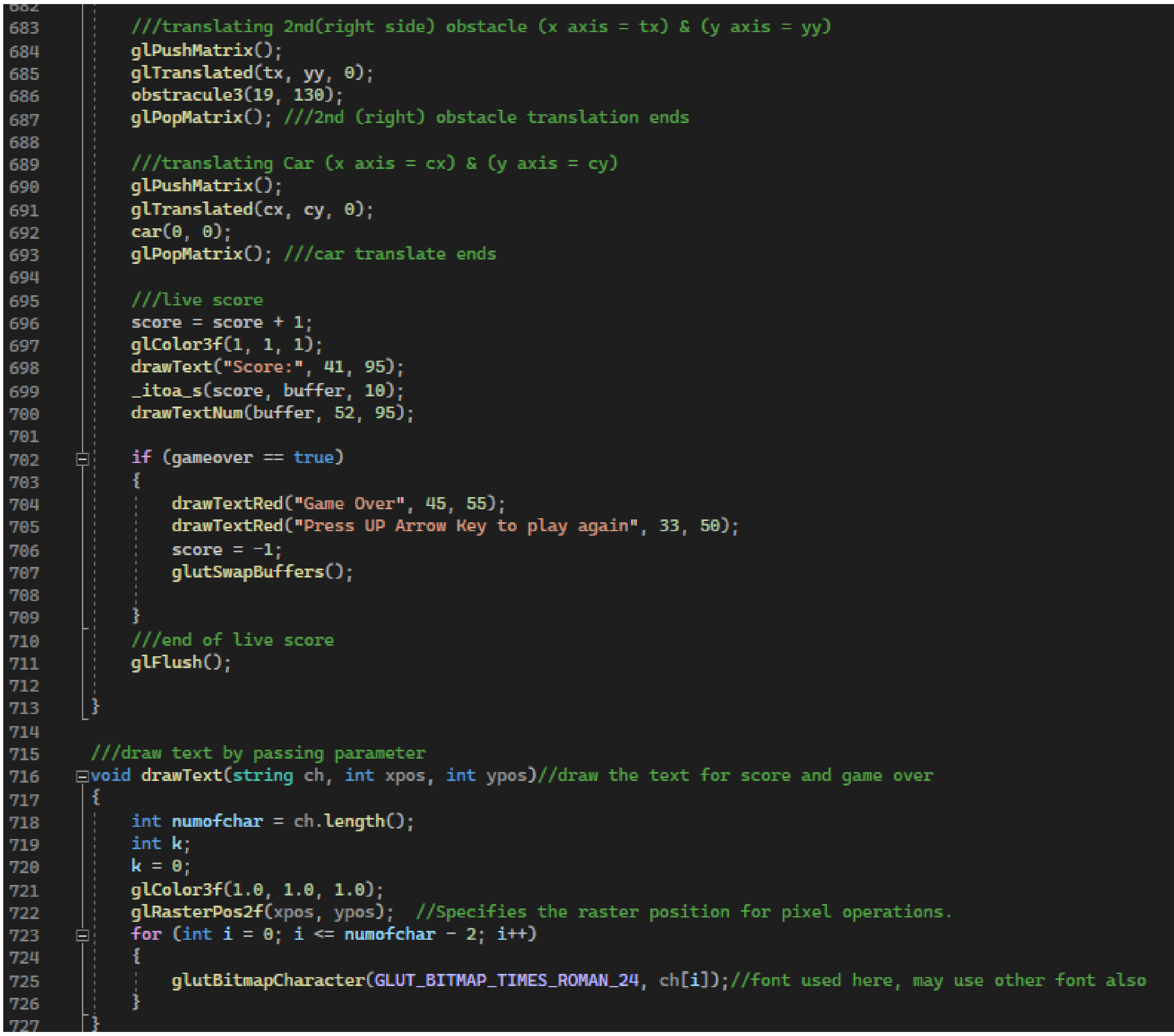
```

    if (gameover == true)
    {
        drawTextRed("Game Over", 45, 55);
        drawTextRed("Press UP Arrow Key to play again", 33, 50);
        score = -1;
        glutSwapBuffers(); //swaps the buffers of the current window if double buffered
        drawTextRed("Published By :-", 45, 45);
        drawTextRed("Parth_Vohra", 45, 40);
        drawTextRed("Bipasha_Gupta", 45, 35);

    }
    ///end of live score
    glFlush();

}
```



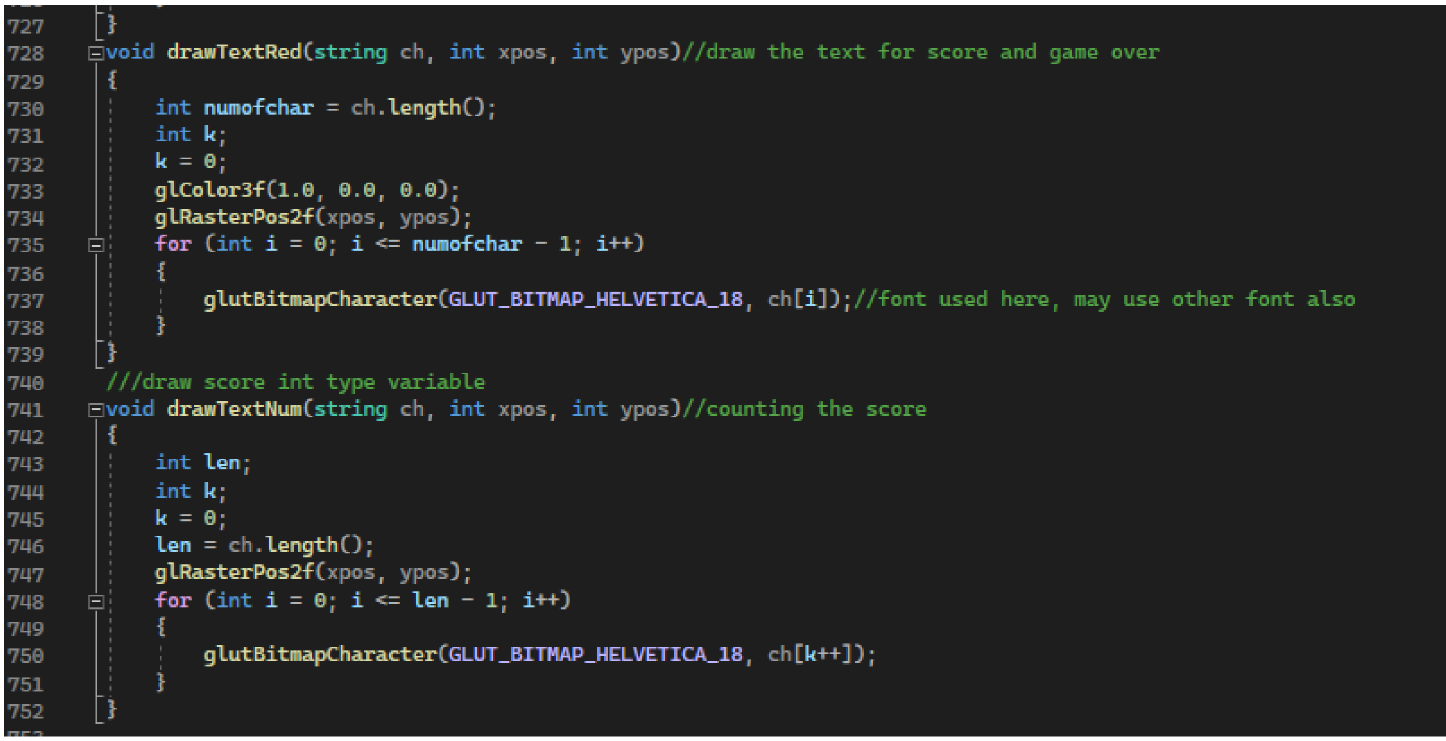
```
void display_level2()
{

    ///for clear all pixels
    glClear(GL_COLOR_BUFFER_BIT); //clear buffers to preset values

    ///1st window main drawing start from origin x=0 y=0
    ///translate window's component that means changing position of component

    glPushMatrix();
    glTranslated(tx, ty, 0);
    draw_all_level2(0, 0);
    glPopMatrix(); ///end of 1st draw_all() function

    ///2nd window drawing of all components x remain same but y increased by 100
    ///that will draw all components outside of top window
    glPushMatrix();
    glTranslated(tx, ty, 0);
    draw_all_level2(0, 100);
    glPopMatrix();///end of 2nd draw_all() function for animation

    ///translating 1st(left side) obstacle (x axis = tx) & (y axis = y)
    /// y axis need not any translation because
    glPushMatrix();
    glTranslated(tx, y, 0);
    obstracule(0, 50);
    glPopMatrix(); ///1st(left) obstacle translation ends

    ///translating 2nd(right side) obstacle (x axis = tx) & (y axis = yy)
    glPushMatrix();
    glTranslated(tx, yy, 0);
    obstracule(19, 130);
    glPopMatrix(); ///2nd (right) obstacle translation ends

    ///translating Car (x axis = cx) & (y axis = cy)
    glPushMatrix();
    glTranslated(cx, cy, 0);
    car(0, 0);
    glPopMatrix(); ///car translate ends

    ///live score
    score = score + 1;
    glColor3f(1, 1, 1);
    drawText("Score:", 41, 95);
    _itoa_s(score, buffer, 10);
    drawTextNum(buffer, 52, 95);

```



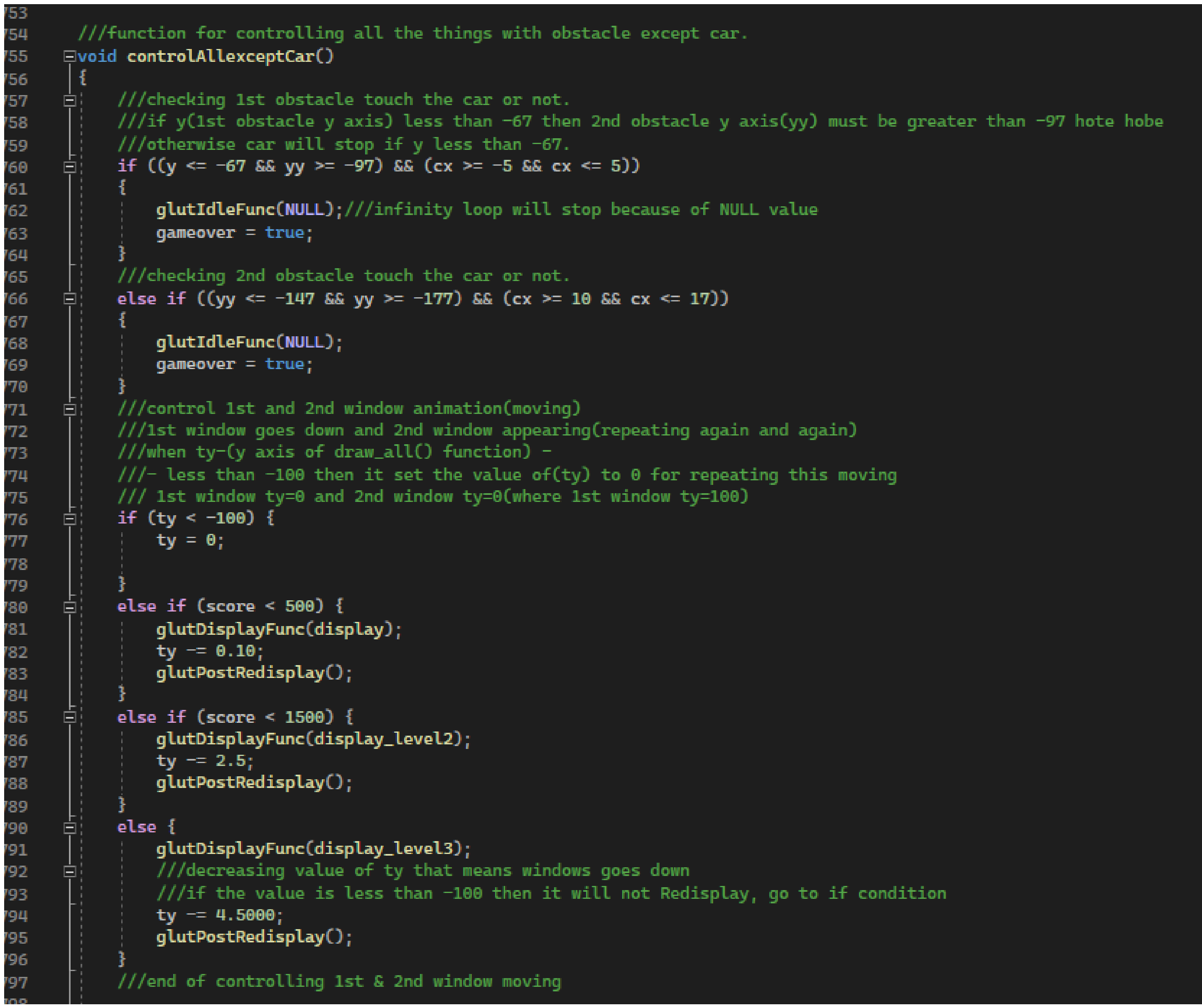
```
    if (gameover == true)
    {
        drawTextRed("Game Over", 45, 55);
        drawTextRed("Press UP Arrow Key to play again", 33, 50);
        score = -1;
        glutSwapBuffers();

    }
    ///end of live score
    glFlush();

}

void display_level3()
{

    ///for clear all pixels
    glClear(GL_COLOR_BUFFER_BIT);

    ///1st window main drawing start from origin x=0 y=0
    ///translate window's component that means changing position of component

    glPushMatrix();
    glTranslated(tx, ty, 0);
    draw_all_level3(0, 0);
    glPopMatrix(); ///end of 1st draw_all() function

    ///2nd window drawing of all components x remain same but y increased by 100
    ///that will draw all components outside of top window
    glPushMatrix();
    glTranslated(tx, ty, 0);
    draw_all_level3(0, 100);
    glPopMatrix();///end of 2nd draw_all() function for animation

    ///translating 1st(left side) obstacle (x axis = tx) & (y axis = y)
    /// y axis need not any translation because
    glPushMatrix();
    glTranslated(tx, y, 0);
    obstracule3(0, 50);
    glPopMatrix(); ///1st(left) obstacle translation ends
```



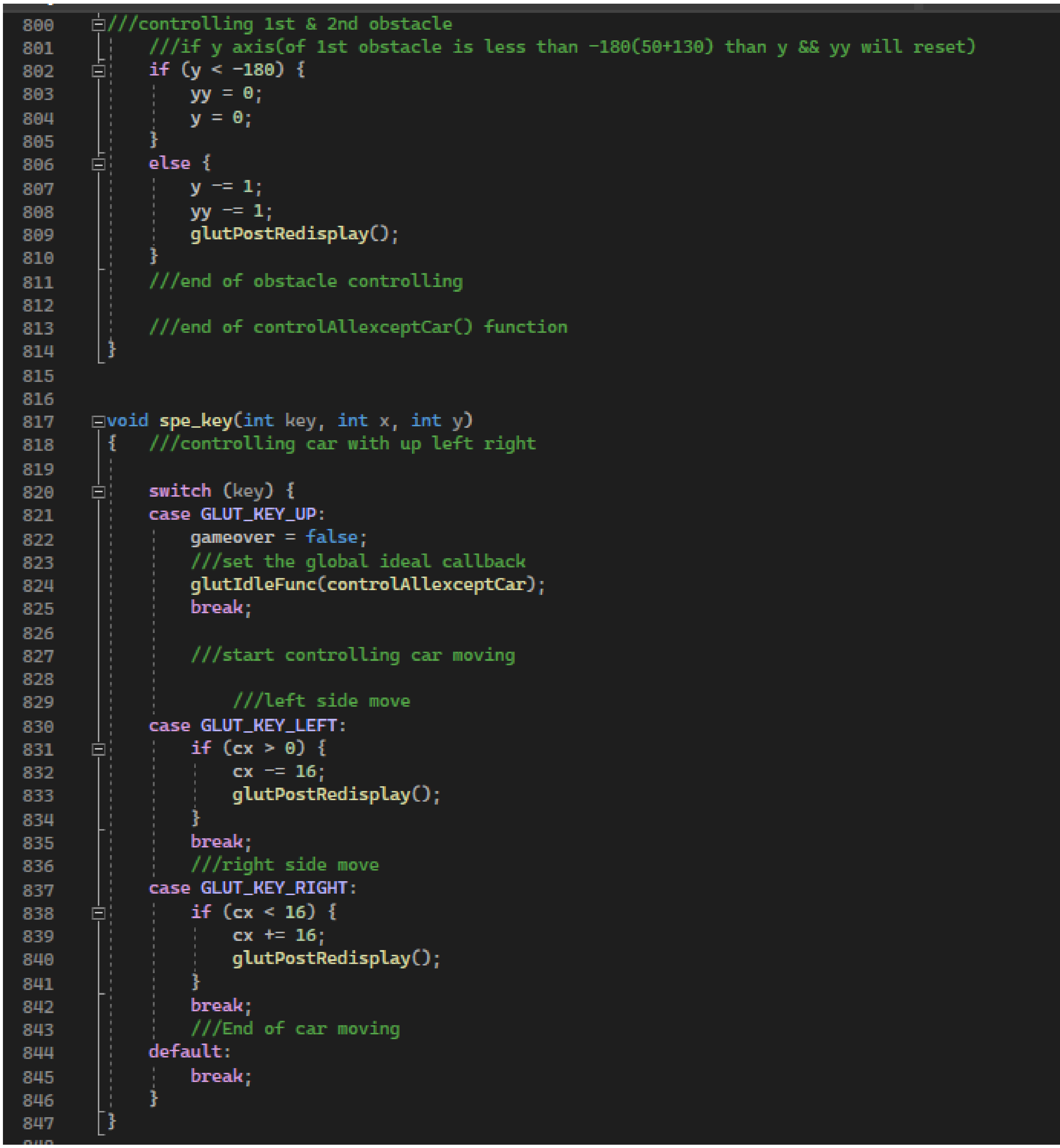
```
    ///translating 2nd(right side) obstacle (x axis = tx) & (y axis = yy)
    glPushMatrix();
    glTranslated(tx, yy, 0);
    obstracule3(19, 130);
    glPopMatrix(); ///2nd (right) obstacle translation ends

    ///translating Car (x axis = cx) & (y axis = cy)
    glPushMatrix();
    glTranslated(cx, cy, 0);
    car(0, 0);
    glPopMatrix(); ///car translate ends

    ///live score
    score = score + 1;
    glColor3f(1, 1, 1);
    drawText("Score:", 41, 95);
    _itoa_s(score, buffer, 10);
    drawTextNum(buffer, 52, 95);

    if (gameover == true)
    {
        drawTextRed("Game Over", 45, 55);
        drawTextRed("Press UP Arrow Key to play again", 33, 50);
        score = -1;
        glutSwapBuffers();

    }
    ///end of live score
    glFlush();

}

///draw text by passing parameter
void drawText(string ch, int xpos, int ypos)//draw the text for score and game over
{
    int numofchar = ch.length();
    int k;
    k = 0;
    glColor3f(1.0, 1.0, 1.0);
    glRasterPos2f(xpos, ypos);  //Specifies the raster position for pixel operations.
    for (int i = 0; i <= numofchar - 2; i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, ch[i]);//font used here, may use other font also
    }
}
```



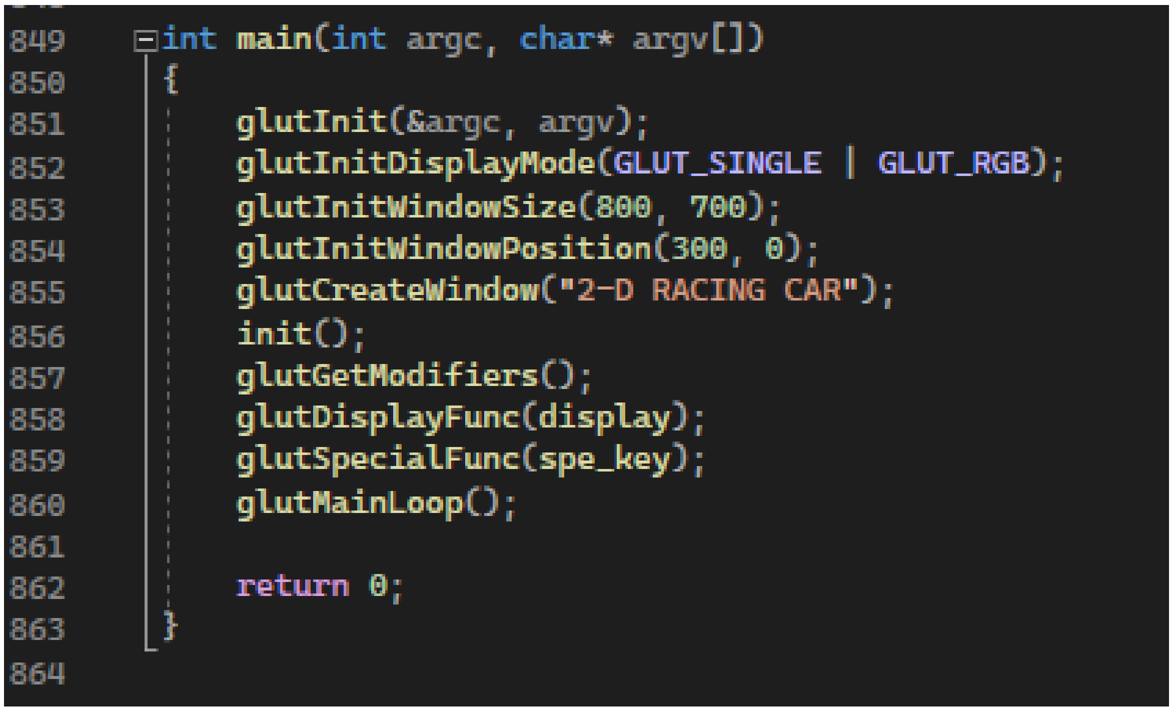
```
}
void drawTextRed(string ch, int xpos, int ypos)//draw the text for score and game over
{
    int numofchar = ch.length();
    int k;
    k = 0;
    glColor3f(1.0, 0.0, 0.0);
    glRasterPos2f(xpos, ypos);
    for (int i = 0; i <= numofchar - 1; i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[i]);//font used here, may use other font also
    }
}
///draw score int type variable
void drawTextNum(string ch, int xpos, int ypos)//counting the score
{
    int len;
    int k;
    k = 0;
    len = ch.length();
    glRasterPos2f(xpos, ypos);
    for (int i = 0; i <= len - 1; i++)
    {
        glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[k++]);
    }
}
```



```
///function for controlling all the things with obstacle except car.
void controlAllexceptCar()
{
    ///checking 1st obstacle touch the car or not.
    ///if y(1st obstacle y axis) less than -67 then 2nd obstacle y axis(yy) must be greater than -97 hote hobe
    ///otherwise car will stop if y less than -67.
    if ((y <= -67 && yy >= -97) && (cx >= -5 && cx <= 5))
    {
        glutIdleFunc(NULL);///infinity loop will stop because of NULL value
        gameover = true;
    }
    ///checking 2nd obstacle touch the car or not.
    else if ((yy <= -147 && yy >= -177) && (cx >= 10 && cx <= 17))
    {
        glutIdleFunc(NULL);
        gameover = true;
    }
    ///control 1st and 2nd window animation(moving)
    ///1st window goes down and 2nd window appearing(repeating again and again)
    ///when ty-(y axis of draw_all() function) -
    ///- less than -100 then it set the value of(ty) to 0 for repeating this moving
    /// 1st window ty=0 and 2nd window ty=0(where 1st window ty=100)
    if (ty < -100) {
        ty = 0;

    }
    else if (score < 500) {
        glutDisplayFunc(display);
        ty -= 0.10;
        glutPostRedisplay();
    }
    else if (score < 1500) {
        glutDisplayFunc(display_level2);
        ty -= 2.5;
        glutPostRedisplay();
    }
    else {
        glutDisplayFunc(display_level3);
        ///decreasing value of ty that means windows goes down
        ///if the value is less than -100 then it will not Redisplay, go to if condition
        ty -= 4.5000;
        glutPostRedisplay();
    }
    ///end of controlling 1st & 2nd window moving
```



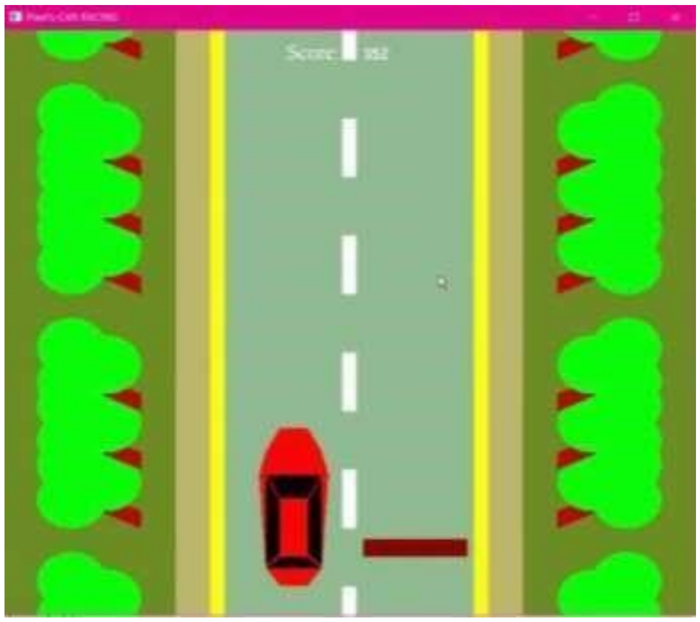
```
///controlling 1st & 2nd obstacle
    ///if y axis(of 1st obstacle is less than -180(50+130) than y && yy will reset)
    if (y < -180) {
        yy = 0;
        y = 0;
    }
    else {
        y -= 1;
        yy -= 1;
        glutPostRedisplay();
    }
    ///end of obstacle controlling

    ///end of controlAllexceptCar() function
}


void spe_key(int key, int x, int y)
{   ///controlling car with up left right

    switch (key) {
    case GLUT_KEY_UP:
        gameover = false;
        ///set the global ideal callback
        glutIdleFunc(controlAllexceptCar);
        break;

        ///start controlling car moving

            ///left side move
    case GLUT_KEY_LEFT:
        if (cx > 0) {
            cx -= 16;
            glutPostRedisplay();
        }
        break;
        ///right side move
    case GLUT_KEY_RIGHT:
        if (cx < 16) {
            cx += 16;
            glutPostRedisplay();
        }
        break;
        ///End of car moving
    default:
        break;
    }
}
```



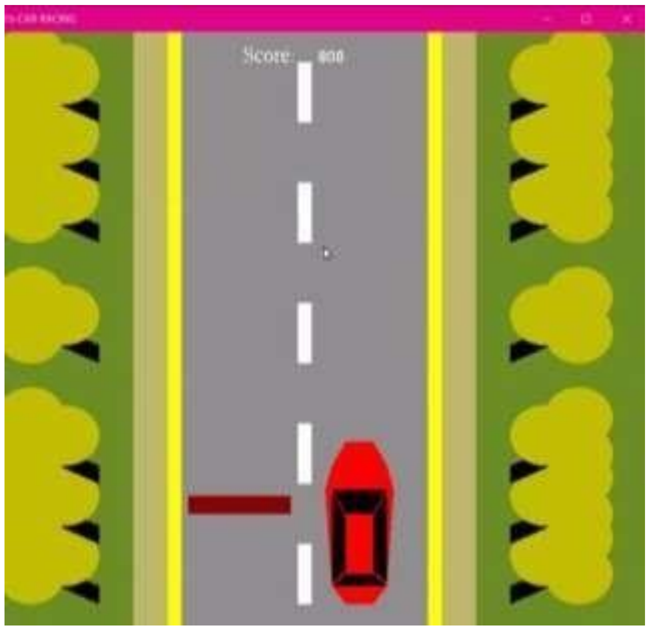
```
int main(int argc, char* argv[])
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
    glutInitWindowSize(800, 700);
    glutInitWindowPosition(300, 0);
    glutCreateWindow("2-D RACING CAR");
    init();
    glutGetModifiers();
    glutDisplayFunc(display);
    glutSpecialFunc(spe_key);
    glutMainLoop();

    return 0;
}
```

6. Output:

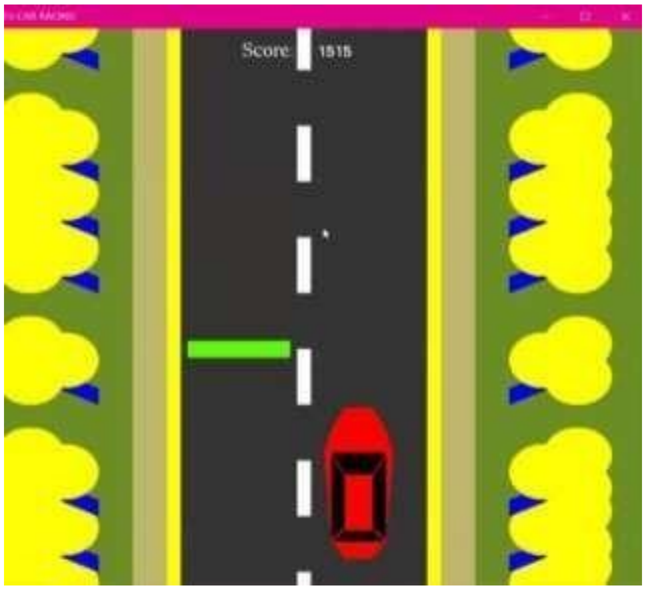
Level 1:



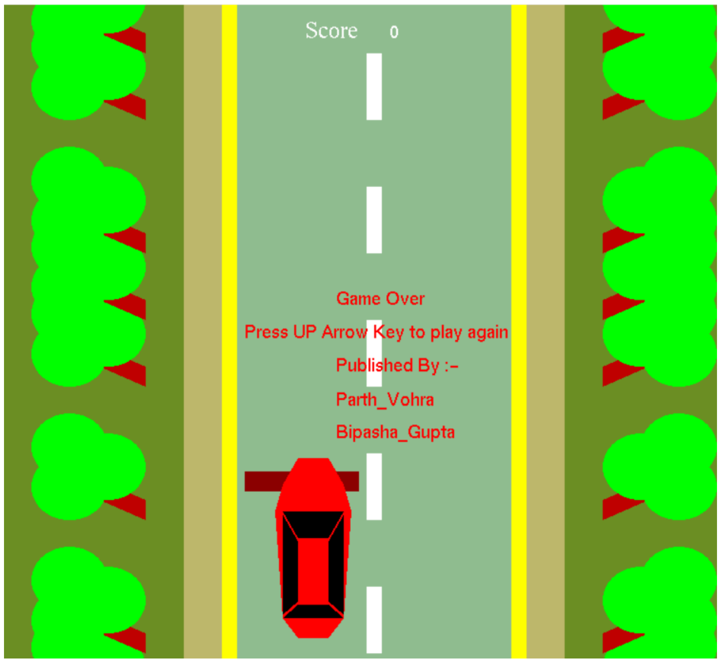
Level 2:



Level 3:



When the game is over:



Game Over
Press UP Arrow Key to play again
Published By :-
Parth_Vohra
Bipasha_Gupta