

Computer Graphics

(UCS505)

Project Report On 2-D Racing Car

Submitted by:-

Parth Vohra (102016044)

Bipasha Gupta (102196004)

Group: 3CSE10



Computer Science and Engineering Department
Thapar Institute of Engineering and Technology, Patiala

Table of contents

S No	Description	Page No
1.	Introduction of project	3
2.	Instruction to play the Game	4
3.	Computer graphics concept used	4-5
4.	User Defined Functions	5-6
5.	Code	6-30
6.	Output	31-32

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The report is chalked out into sections describing the computer graphics concepts used superseded by the briefing on functions used.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 $4n - 3$, $4n - 2$, $4n - 1$, and $4n$ 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.

//

```

1
2  #include <GL/glut.h>
3  #include <string>
4  using namespace std;
5  GLvoid obstracule(GLdouble x, GLdouble y);
6  ///function prototype for drawing text
7  void drawText(string str, int xpos, int ypos);
8  void drawTextRed(string str, int xpos, int ypos);
9  ///draw score
10 char buffer[10];
11 void drawTextNum(string ch, int xpos, int ypos);
12 ///take bool type variable for controlling game over and score
13 bool gameover = false;
14
15 int score = -1;
16 float tx = 0, ty = 0, x = 0, yy = 0; ///for draw_all
17 float cx = 0, cy = 0; ///for car
18
19 void init(void)
20 {
21     glClearColor(0.420, 0.557, 0.137, 0.0);
22     glOrtho(0, 100, 0, 100, -1.0, 1.0); //describes a transformation that produces a parallel projection.
23 }
24

```

```

25 char* itoa(long i, char* s, int dummy_radix) {
26     sprintf_s(s, 100, "%ld", i);
27     return s;
28 }
29 GLvoid drawCircle(GLdouble xc, GLdouble yc, GLdouble rad)///function for drawing circle
30 {
31     GLfloat i;
32     glPointSize(3);
33     glBegin(GL_POLYGON);
34
35     for (i = 0; i <= 7; i += .01)
36         glVertex2f(xc + rad * cos(i), yc + rad * sin(i));
37     glEnd();
38 }
39 GLvoid tree_l(GLdouble x, GLdouble y)///function for drawing left side tree
40 {
41     glBegin(GL_QUADS);
42     glColor3f(.75, 0, 0);
43     glVertex2f(x, y);
44     glVertex2f(x - 10, y + 5);
45     glVertex2f(x - 10, y + 8);
46     glVertex2f(x, y + 3);
47     glEnd();
48
49     glColor3f(0, 1, 0);

```

```

49     glColor3f(0, 1, 0);
50     drawCircle(x - 10, y + 5, 5);
51     drawCircle(x - 10, y + 11, 5);
52     drawCircle(x - 5, y + 8, 5);
53 }
54
55 ///level 1 case
56 GLvoid tree_r(GLdouble x, GLdouble y)///function for drawing right side tree
57 {
58     glBegin(GL_QUADS);
59     glColor3f(.75, 0, 0);
60     glVertex2f(x, y);
61     glVertex2f(x + 10, y + 5);
62     glVertex2f(x + 10, y + 8);
63     glVertex2f(x, y + 3);
64     glEnd();
65     glColor3f(0, 1, 0);
66     drawCircle(x + 10, y + 5, 5);
67     drawCircle(x + 10, y + 11, 5);
68     drawCircle(x + 5, y + 8, 5);
69 }
70

```

```

71
72     ///level 2 case
73     GLvoid tree_l2(GLdouble x, GLdouble y)///function for drawing left side tree
74     {
75         glBegin(GL_QUADS);
76         glColor3f(0, 0, 0);
77         glVertex2f(x, y);
78         glVertex2f(x - 10, y + 5);
79         glVertex2f(x - 10, y + 8);
80         glVertex2f(x, y + 3);
81         glEnd();
82
83         glColor3f(0.75, 0.75, 0);
84         drawCircle(x - 10, y + 5, 5);
85         drawCircle(x - 10, y + 11, 5);
86         drawCircle(x - 5, y + 8, 5);
87     }

```

```

88
89     GLvoid tree_r2(GLdouble x, GLdouble y)///function for drawing right side tree
90     {
91         glBegin(GL_QUADS);
92         glColor3f(0, 0, 0);
93         glVertex2f(x, y);
94         glVertex2f(x + 10, y + 5);
95         glVertex2f(x + 10, y + 8);
96         glVertex2f(x, y + 3);
97         glEnd();
98         glColor3f(0.75, 0.75, 0);
99         drawCircle(x + 10, y + 5, 5);
00         drawCircle(x + 10, y + 11, 5);
01         drawCircle(x + 5, y + 8, 5);
02     }

```

```

103
104     /// level 3 Case
105     GLvoid tree_l3(GLdouble x, GLdouble y)///function for drawing left side tree
106     {
107         glBegin(GL_QUADS);
108         glColor3f(0, 0, 1);
109         glVertex2f(x, y);
110         glVertex2f(x - 10, y + 5);
111         glVertex2f(x - 10, y + 8);
112         glVertex2f(x, y + 3);
113         glEnd();
114
115         glColor3f(1, 1, 0);
116         drawCircle(x - 10, y + 5, 5);
117         drawCircle(x - 10, y + 11, 5);
118         drawCircle(x - 5, y + 8, 5);
119     }

```



```

121  GLvoid tree_r3(GLdouble x, GLdouble y)///function for drawing right side tree
122  {
123      glBegin(GL_QUADS);
124      glColor3f(0, 0, 1);
125      glVertex2f(x, y);
126      glVertex2f(x + 10, y + 5);
127      glVertex2f(x + 10, y + 8);
128      glVertex2f(x, y + 3);
129      glEnd();
130
131      glColor3f(1, 1, 0);
132      drawCircle(x + 10, y + 5, 5);
133      drawCircle(x + 10, y + 11, 5);
134      drawCircle(x + 5, y + 8, 5);
135  }
136

```

```

137
138  GLvoid draw_all(GLdouble x, GLdouble y)///function for drawing everything except car
139  {
140      tree_l(x + 20, y + 0);///left side tree
141      tree_l(x + 20, y + 10);
142      tree_l(x + 20, y + 30);
143      tree_l(x + 20, y + 50);
144      tree_l(x + 20, y + 60);
145      tree_l(x + 20, y + 70);
146      tree_l(x + 20, y + 90);
147
148      tree_r(x + 80, y + 0);///right side tree
149      tree_r(x + 80, y + 10);
150      tree_r(x + 80, y + 30);
151      tree_r(x + 80, y + 50);
152      tree_r(x + 80, y + 60);
153      tree_r(x + 80, y + 70);
154      tree_r(x + 80, y + 90);

```

```
156     glColor3f(0.561, 0.737, 0.561);
157     glBegin(GL_POLYGON); //main road
158     glVertex2f(x + 30, y + 0);
159     glVertex2f(x + 70, y + 0);
160     glVertex2f(x + 70, y + 100);
161     glVertex2f(x + 30, y + 100);
162     glEnd();
163
164     glColor3f(1, 1, 0);
165     glBegin(GL_POLYGON); //yellow line left
166     glVertex2f(x + 30, y + 0);
167     glVertex2f(x + 32, y + 0);
168     glVertex2f(x + 32, y + 100);
169     glVertex2f(x + 30, y + 100);
170     glEnd();
171
172     glColor3f(1, 1, 0);
173     glBegin(GL_POLYGON); //yellow line right
174     glVertex2f(x + 70, y + 0);
175     glVertex2f(x + 68, y + 0);
176     glVertex2f(x + 68, y + 100);
177     glVertex2f(x + 70, y + 100);
178     glEnd();
```

```
179
180     glColor3f(0.741, 0.718, 0.420);
181     glBegin(GL_POLYGON); //left footpath
182     glVertex2f(x + 30, y + 0);
183     glVertex2f(x + 25, y + 0);
184     glVertex2f(x + 25, y + 100);
185     glVertex2f(x + 30, y + 100);
186     glEnd();
187
188     glColor3f(0.741, 0.718, 0.420);
189     glBegin(GL_POLYGON); //right footpath
190     glVertex2f(x + 70, y + 0);
191     glVertex2f(x + 75, y + 0);
192     glVertex2f(x + 75, y + 100);
193     glVertex2f(x + 70, y + 100);
194     glEnd();
195
196     glColor3f(1, 1, 1);
197     glBegin(GL_POLYGON); //zebra lines starts
198     glVertex2f(x + 49, y + 100);
199     glVertex2f(x + 49, y + 90);
200     glVertex2f(x + 51, y + 90);
201     glVertex2f(x + 51, y + 100);
202     glEnd();
```

```

203
204     glColor3f(1, 1, 1);
205     glBegin(GL_POLYGON);
206     glVertex2f(x + 49, y + 80);
207     glVertex2f(x + 49, y + 70);
208     glVertex2f(x + 51, y + 70);
209     glVertex2f(x + 51, y + 80);
210     glEnd();
211
212     glColor3f(1, 1, 1);
213     glBegin(GL_POLYGON);
214     glVertex2f(x + 49, y + 60);
215     glVertex2f(x + 49, y + 50);
216     glVertex2f(x + 51, y + 50);
217     glVertex2f(x + 51, y + 60);
218     glEnd();
219
220     glColor3f(1, 1, 1);
221     glBegin(GL_POLYGON);
222     glVertex2f(x + 49, y + 40);
223     glVertex2f(x + 49, y + 30);
224     glVertex2f(x + 51, y + 30);
225     glVertex2f(x + 51, y + 40);
226     glEnd();

```

```

228     glColor3f(1, 1, 1);
229     glBegin(GL_POLYGON); //zebra lines finishes
230     glVertex2f(x + 49, y + 20);
231     glVertex2f(x + 49, y + 10);
232     glVertex2f(x + 51, y + 10);
233     glVertex2f(x + 51, y + 20);
234     glEnd();
235
236 }
237

```

```

238  GLvoid draw_all_level2(GLdouble x, GLdouble y)///function for drawing everything except car
239  {
240      tree_l2(x + 20, y + 0);///left side tree
241      tree_l2(x + 20, y + 10);
242      tree_l2(x + 20, y + 30);
243      tree_l2(x + 20, y + 50);
244      tree_l2(x + 20, y + 60);
245      tree_l2(x + 20, y + 70);
246      tree_l2(x + 20, y + 90);
247
248      tree_r2(x + 80, y + 0);///right side tree
249      tree_r2(x + 80, y + 10);
250      tree_r2(x + 80, y + 30);
251      tree_r2(x + 80, y + 50);
252      tree_r2(x + 80, y + 60);
253      tree_r2(x + 80, y + 70);
254      tree_r2(x + 80, y + 90);
255
256      glColor3f(0.561, 0.561, 0.561);
257      glBegin(GL_POLYGON);///main road
258      glVertex2f(x + 30, y + 0);
259      glVertex2f(x + 70, y + 0);
260      glVertex2f(x + 70, y + 100);
261      glVertex2f(x + 30, y + 100);
262      glEnd();

```

```

264      glColor3f(1, 1, 0);
265      glBegin(GL_POLYGON);///yellow line left
266      glVertex2f(x + 30, y + 0);
267      glVertex2f(x + 32, y + 0);
268      glVertex2f(x + 32, y + 100);
269      glVertex2f(x + 30, y + 100);
270      glEnd();
271
272      glColor3f(1, 1, 0);
273      glBegin(GL_POLYGON);///yellow line right
274      glVertex2f(x + 70, y + 0);
275      glVertex2f(x + 68, y + 0);
276      glVertex2f(x + 68, y + 100);
277      glVertex2f(x + 70, y + 100);
278      glEnd();

```

```
280     glColor3f(0.741, 0.718, 0.420);
281     glBegin(GL_POLYGON); //left footpath
282     glVertex2f(x + 30, y + 0);
283     glVertex2f(x + 25, y + 0);
284     glVertex2f(x + 25, y + 100);
285     glVertex2f(x + 30, y + 100);
286     glEnd();
287
288     glColor3f(0.741, 0.718, 0.420);
289     glBegin(GL_POLYGON); //right footpath
290     glVertex2f(x + 70, y + 0);
291     glVertex2f(x + 75, y + 0);
292     glVertex2f(x + 75, y + 100);
293     glVertex2f(x + 70, y + 100);
294     glEnd();
295
296     glColor3f(1, 1, 1);
297     glBegin(GL_POLYGON); //zebra lines starts
298     glVertex2f(x + 49, y + 100);
299     glVertex2f(x + 49, y + 90);
300     glVertex2f(x + 51, y + 90);
301     glVertex2f(x + 51, y + 100);
302     glEnd();
```

```

304     glColor3f(1, 1, 1);
305     glBegin(GL_POLYGON);
306     glVertex2f(x + 49, y + 80);
307     glVertex2f(x + 49, y + 70);
308     glVertex2f(x + 51, y + 70);
309     glVertex2f(x + 51, y + 80);
310     glEnd();
311
312     glColor3f(1, 1, 1);
313     glBegin(GL_POLYGON);
314     glVertex2f(x + 49, y + 60);
315     glVertex2f(x + 49, y + 50);
316     glVertex2f(x + 51, y + 50);
317     glVertex2f(x + 51, y + 60);
318     glEnd();
319
320     glColor3f(1, 1, 1);
321     glBegin(GL_POLYGON);
322     glVertex2f(x + 49, y + 40);
323     glVertex2f(x + 49, y + 30);
324     glVertex2f(x + 51, y + 30);
325     glVertex2f(x + 51, y + 40);
326     glEnd();

```

```

328     glColor3f(1, 1, 1);
329     glBegin(GL_POLYGON); //zebra lines finishes
330     glVertex2f(x + 49, y + 20);
331     glVertex2f(x + 49, y + 10);
332     glVertex2f(x + 51, y + 10);
333     glVertex2f(x + 51, y + 20);
334     glEnd();
335
336 }

```

```
338  GLvoid draw_all_level3(GLdouble x, GLdouble y)///function for drawing everything except car
339  {
340      tree_l3(x + 20, y + 0);///left side tree
341      tree_l3(x + 20, y + 10);
342      tree_l3(x + 20, y + 30);
343      tree_l3(x + 20, y + 50);
344      tree_l3(x + 20, y + 60);
345      tree_l3(x + 20, y + 70);
346      tree_l3(x + 20, y + 90);
347
348      tree_r3(x + 80, y + 0);///right side tree
349      tree_r3(x + 80, y + 10);
350      tree_r3(x + 80, y + 30);
351      tree_r3(x + 80, y + 50);
352      tree_r3(x + 80, y + 60);
353      tree_r3(x + 80, y + 70);
354      tree_r3(x + 80, y + 90);
355
356      glColor3f(0.2, 0.2, 0.2);
357      glBegin(GL_POLYGON);///main road
358      glVertex2f(x + 30, y + 0);
359      glVertex2f(x + 70, y + 0);
360      glVertex2f(x + 70, y + 100);
361      glVertex2f(x + 30, y + 100);
362      glEnd();
```



```

364     glColor3f(1, 1, 0);
365     glBegin(GL_POLYGON); //yellow line left
366     glVertex2f(x + 30, y + 0);
367     glVertex2f(x + 32, y + 0);
368     glVertex2f(x + 32, y + 100);
369     glVertex2f(x + 30, y + 100);
370     glEnd();
371
372     glColor3f(1, 1, 0);
373     glBegin(GL_POLYGON); //yellow line right
374     glVertex2f(x + 70, y + 0);
375     glVertex2f(x + 68, y + 0);
376     glVertex2f(x + 68, y + 100);
377     glVertex2f(x + 70, y + 100);
378     glEnd();
379
380     glColor3f(0.741, 0.718, 0.420);
381     glBegin(GL_POLYGON); //left footpath
382     glVertex2f(x + 30, y + 0);
383     glVertex2f(x + 25, y + 0);
384     glVertex2f(x + 25, y + 100);
385     glVertex2f(x + 30, y + 100);
386     glEnd();

```

```
388     glColor3f(0.741, 0.718, 0.420);
389     glBegin(GL_POLYGON); //right footpath
390     glVertex2f(x + 70, y + 0);
391     glVertex2f(x + 75, y + 0);
392     glVertex2f(x + 75, y + 100);
393     glVertex2f(x + 70, y + 100);
394     glEnd();
395
396     glColor3f(1, 1, 1);
397     glBegin(GL_POLYGON); //zebra lines starts
398     glVertex2f(x + 49, y + 100);
399     glVertex2f(x + 49, y + 90);
400     glVertex2f(x + 51, y + 90);
401     glVertex2f(x + 51, y + 100);
402     glEnd();
403
404     glColor3f(1, 1, 1);
405     glBegin(GL_POLYGON);
406     glVertex2f(x + 49, y + 80);
407     glVertex2f(x + 49, y + 70);
408     glVertex2f(x + 51, y + 70);
409     glVertex2f(x + 51, y + 80);
410     glEnd();
```

```

411     glColor3f(1, 1, 1);
412     glBegin(GL_POLYGON);
413     glVertex2f(x + 49, y + 60);
414     glVertex2f(x + 49, y + 50);
415     glVertex2f(x + 51, y + 50);
416     glVertex2f(x + 51, y + 60);
417     glEnd();
418
419
420     glColor3f(1, 1, 1);
421     glBegin(GL_POLYGON);
422     glVertex2f(x + 49, y + 40);
423     glVertex2f(x + 49, y + 30);
424     glVertex2f(x + 51, y + 30);
425     glVertex2f(x + 51, y + 40);
426     glEnd();
427
428     glColor3f(1, 1, 1);
429     glBegin(GL_POLYGON); //zebra lines finishes
430     glVertex2f(x + 49, y + 20);
431     glVertex2f(x + 49, y + 10);
432     glVertex2f(x + 51, y + 10);
433     glVertex2f(x + 51, y + 20);
434     glEnd();
435 }
436
437
438 GLvoid obstracule(GLdouble x, GLdouble y) //function for drawing obstacle
439 {
440     glColor3f(0.545, 0.000, 0.000);
441     glBegin(GL_POLYGON); //obstracules
442     glVertex2f(x + 33, y + 50);
443     glVertex2f(x + 48, y + 50);
444     glVertex2f(x + 48, y + 53);
445     glVertex2f(x + 33, y + 53);
446     glEnd();
447 }
448
449 //obstacle green color for level 3
450 GLvoid obstracule3(GLdouble x, GLdouble y) //function for drawing obstacle
451 {
452     glColor3f(0.34, 1, 0);
453     glBegin(GL_POLYGON); //obstracules
454     glVertex2f(x + 33, y + 50);
455     glVertex2f(x + 48, y + 50);
456     glVertex2f(x + 48, y + 53);
457     glVertex2f(x + 33, y + 53);
458     glEnd();
459 }

```

```

460
461 GLvoid car(GLdouble x, GLdouble y)///function for drawing car
462 {
463     glColor3f(1, 0, 0);
464     glBegin(GL_POLYGON);///player car body
465     glVertex2f(x + 40, y + 5);
466     glVertex2f(x + 44, y + 5);
467     glVertex2f(x + 46, y + 8);
468     glVertex2f(x + 47, y + 24);
469     glVertex2f(x + 46, y + 28);
470     glVertex2f(x + 44, y + 32);
471     glVertex2f(x + 40, y + 32);
472     glVertex2f(x + 38, y + 28);
473     glVertex2f(x + 37, y + 24);
474     glVertex2f(x + 38, y + 8);
475     glVertex2f(x + 40, y + 5);
476     glEnd();
477
478     glColor3f(0, 0, 0);///car inside
479     glBegin(GL_POLYGON);
480     glVertex2f(x + 38, y + 8);
481     glVertex2f(x + 46, y + 8);
482     glVertex2f(x + 46, y + 24);
483     glVertex2f(x + 38, y + 24);
484     glVertex2f(x + 38, y + 8);
485     glEnd();
486
487     glColor3f(1, 0, 0);///car roof
488     glBegin(GL_POLYGON);
489     glVertex2f(x + 40, y + 10);
490     glVertex2f(x + 44, y + 10);
491     glVertex2f(x + 44, y + 20);
492     glVertex2f(x + 40, y + 20);
493     glVertex2f(x + 40, y + 10);
494     glEnd();
495
496     glColor3f(1, 0, 0);///up right roof connector
497     glBegin(GL_POLYGON);
498     glVertex2f(x + 44, y + 20);
499     glVertex2f(x + 44, y + 19.5);
500     glVertex2f(x + 46, y + 23.5);
501     glVertex2f(x + 46, y + 24);
502     glVertex2f(x + 44, y + 20);
503     glEnd();
504

```

```

505     glColor3f(1, 0, 0); //up left roof connector
506     glBegin(GL_POLYGON);
507     glVertex2f(x + 40, y + 20);
508     glVertex2f(x + 40, y + 19.5);
509     glVertex2f(x + 38, y + 23.5);
510     glVertex2f(x + 38, y + 24);
511     glVertex2f(x + 40, y + 20);
512     glEnd();
513
514     glColor3f(1, 0, 0); //bottom right roof connector
515     glBegin(GL_POLYGON);
516     glVertex2f(x + 44, y + 10);
517     glVertex2f(x + 44, y + 10.5);
518     glVertex2f(x + 46, y + 8.5);
519     glVertex2f(x + 46, y + 8);
520     glVertex2f(x + 44, y + 10);
521     glEnd();
522
523     glColor3f(1, 0, 0); //bottom left roof connector
524     glBegin(GL_POLYGON);
525     glVertex2f(x + 40, y + 10);
526     glVertex2f(x + 40, y + 10.5);
527     glVertex2f(x + 38, y + 8.5);
528     glVertex2f(x + 38, y + 8);
529     glVertex2f(x + 40, y + 10);
530     glEnd();
531 }

```

```

533 void display()
534 {
535
536     ///for clear all pixels
537     glClear(GL_COLOR_BUFFER_BIT);
538
539     ///1st window main drawing start from origin x=0 y=0
540     ///translate window's component that means changing position of component
541
542     glPushMatrix(); // push and pop the current matrix stack
543     glTranslated(tx, ty, 0);
544     draw_all(0, 0);
545     glPopMatrix(); ///end of 1st draw_all() function
546
547     ///2nd window drawing of all components x remain same but y increased by 100
548     ///that will draw all components outside of top window
549     glPushMatrix();
550     glTranslated(tx, ty, 0);
551     draw_all(0, 100);
552     glPopMatrix();///end of 2nd draw_all() function for animation
553
554     ///translating 1st(left side) obstacle (x axis = tx) & (y axis = y)
555     /// y axis need not any translation because
556     glPushMatrix();
557     glTranslated(tx, y, 0);
558     obstracule(0, 50);
559     glPopMatrix(); ///1st(left) obstacle translation ends
560
561     ///translating 2nd(right side) obstacle (x axis = tx) & (y axis = yy)
562     glPushMatrix();
563     glTranslated(tx, yy, 0);
564     obstracule(19, 130);
565     glPopMatrix(); ///2nd (right) obstacle translation ends
566
567     ///translating Car (x axis = cx) & (y axis = cy)
568     glPushMatrix();
569     glTranslated(cx, cy, 0);
570     car(0, 0);
571     glPopMatrix(); ///car translate ends
572
573     ///live score
574     score = score + 1;
575     glColor3f(1, 1, 1);
576     drawText("Score:", 41, 95);
577     _itoa_s(score, buffer, 10);
578     drawTextNum(buffer, 52, 95);

```

```
579
580     if (gameover == true)
581     {
582         drawTextRed("Game Over", 45, 55);
583         drawTextRed("Press UP Arrow Key to play again", 33, 50);
584         score = -1;
585         glutSwapBuffers(); //swaps the buffers of the current window if double buffered
586         drawTextRed("Published By :-", 45, 45);
587         drawTextRed("Parth_Vohra", 45, 40);
588         drawTextRed("Bipasha_Gupta", 45, 35);
589     }
590     ///end of live score
591     glFlush();
592 }
593
594
```



```

595 void display_level2()
596 {
597
598     ///for clear all pixels
599     glClear(GL_COLOR_BUFFER_BIT); //clear buffers to preset values
600
601     ///1st window main drawing start from origin x=0 y=0
602     ///translate window's component that means changing position of component
603
604     glPushMatrix();
605     glTranslated(tx, ty, 0);
606     draw_all_level2(0, 0);
607     glPopMatrix(); ///end of 1st draw_all() function
608
609     ///2nd window drawing of all components x remain same but y increased by 100
610     ///that will draw all components outside of top window
611     glPushMatrix();
612     glTranslated(tx, ty, 0);
613     draw_all_level2(0, 100);
614     glPopMatrix();///end of 2nd draw_all() function for animation
615
616     ///translating 1st(left side) obstacle (x axis = tx) & (y axis = y)
617     /// y axis need not any translation because
618     glPushMatrix();
619     glTranslated(tx, y, 0);
620     obstracule(0, 50);
621     glPopMatrix(); ///1st(left) obstacle translation ends
622
623     ///translating 2nd(right side) obstacle (x axis = tx) & (y axis = yy)
624     glPushMatrix();
625     glTranslated(tx, yy, 0);
626     obstracule(19, 130);
627     glPopMatrix(); ///2nd (right) obstacle translation ends
628
629     ///translating Car (x axis = cx) & (y axis = cy)
630     glPushMatrix();
631     glTranslated(cx, cy, 0);
632     car(0, 0);
633     glPopMatrix(); ///car translate ends
634
635     ///live score
636     score = score + 1;
637     glColor3f(1, 1, 1);
638     drawText("Score:", 41, 95);
639     _itoa_s(score, buffer, 10);
640     drawTextNum(buffer, 52, 95);
641

```



```

641
642     if (gameover == true)
643     {
644         drawTextRed("Game Over", 45, 55);
645         drawTextRed("Press UP Arrow Key to play again", 33, 50);
646         score = -1;
647         glutSwapBuffers();
648     }
649     ///end of live score
650     glFlush();
651 }
652
653
654
655 void display_level3()
656 {
657     ///for clear all pixels
658     glClear(GL_COLOR_BUFFER_BIT);
659
660     ///1st window main drawing start from origin x=0 y=0
661     ///translate window's component that means changing position of component
662
663     glPushMatrix();
664     glTranslated(tx, ty, 0);
665     draw_all_level3(0, 0);
666     glPopMatrix(); ///end of 1st draw_all() function
667
668     ///2nd window drawing of all components x remain same but y increased by 100
669     ///that will draw all components outside of top window
670     glPushMatrix();
671     glTranslated(tx, ty, 0);
672     draw_all_level3(0, 100);
673     glPopMatrix(); ///end of 2nd draw_all() function for animation
674
675     ///translating 1st(left side) obstacle (x axis = tx) & (y axis = y)
676     /// y axis need not any translation because
677     glPushMatrix();
678     glTranslated(tx, y, 0);
679     obstracule3(0, 50);
680     glPopMatrix(); ///1st(left) obstacle translation ends
681
682

```

```

682
683     ///translating 2nd(right side) obstacle (x axis = tx) & (y axis = yy)
684     glPushMatrix();
685     glTranslated(tx, yy, 0);
686     obstracule3(19, 130);
687     glPopMatrix(); ///2nd (right) obstacle translation ends
688
689     ///translating Car (x axis = cx) & (y axis = cy)
690     glPushMatrix();
691     glTranslated(cx, cy, 0);
692     car(0, 0);
693     glPopMatrix(); ///car translate ends
694
695     ///live score
696     score = score + 1;
697     glColor3f(1, 1, 1);
698     drawText("Score:", 41, 95);
699     _itoa_s(score, buffer, 10);
700     drawTextNum(buffer, 52, 95);
701
702     if (gameover == true)
703     {
704         drawTextRed("Game Over", 45, 55);
705         drawTextRed("Press UP Arrow Key to play again", 33, 50);
706         score = -1;
707         glutSwapBuffers();
708     }
709
710     ///end of live score
711     glFlush();
712 }
713
714
715     ///draw text by passing parameter
716 void drawText(string ch, int xpos, int ypos)//draw the text for score and game over
717 {
718     int numofchar = ch.length();
719     int k;
720     k = 0;
721     glColor3f(1.0, 1.0, 1.0);
722     glRasterPos2f(xpos, ypos); //Specifies the raster position for pixel operations.
723     for (int i = 0; i <= numofchar - 2; i++)
724     {
725         glutBitmapCharacter(GLUT_BITMAP_TIMES_ROMAN_24, ch[i]); //font used here, may use other font also
726     }
727 }

```

```

727 }
728 void drawTextRed(string ch, int xpos, int ypos)//draw the text for score and game over
729 {
730     int numofchar = ch.length();
731     int k;
732     k = 0;
733     glColor3f(1.0, 0.0, 0.0);
734     glRasterPos2f(xpos, ypos);
735     for (int i = 0; i <= numofchar - 1; i++)
736     {
737         glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[i]); //font used here, may use other font also
738     }
739 }
740 //draw score int type variable
741 void drawTextNum(string ch, int xpos, int ypos)//counting the score
742 {
743     int len;
744     int k;
745     k = 0;
746     len = ch.length();
747     glRasterPos2f(xpos, ypos);
748     for (int i = 0; i <= len - 1; i++)
749     {
750         glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[k++]);
751     }
752 }

```

```

753 //function for controlling all the things with obstacle except car.
754
755 void controlAllexceptCar()
756 {
757     //checking 1st obstacle touch the car or not.
758     //if y(1st obstacle y axis) less than -67 then 2nd obstacle y axis(yy) must be greater than -97 hote hobe
759     //otherwise car will stop if y less than -67.
760     if ((y <= -67 && yy >= -97) && (cx >= -5 && cx <= 5))
761     {
762         glutIdleFunc(NULL); //infinity loop will stop because of NULL value
763         gameover = true;
764     }
765     //checking 2nd obstacle touch the car or not.
766     else if ((yy <= -147 && yy >= -177) && (cx >= 10 && cx <= 17))
767     {
768         glutIdleFunc(NULL);
769         gameover = true;
770     }
771     //control 1st and 2nd window animation(moving)
772     //1st window goes down and 2nd window appearing(repeating again and again)
773     //when ty-(y axis of draw_all() function) -
774     //- less than -100 then it set the value of(ty) to 0 for repeating this moving
775     // 1st window ty=0 and 2nd window ty=0(where 1st window ty=100)
776     if (ty < -100) {
777         ty = 0;
778     }
779
780     else if (score < 500) {
781         glutDisplayFunc(display);
782         ty -= 0.10;
783         glutPostRedisplay();
784     }
785     else if (score < 1500) {
786         glutDisplayFunc(display_level2);
787         ty -= 2.5;
788         glutPostRedisplay();
789     }
790     else {
791         glutDisplayFunc(display_level3);
792         //decreasing value of ty that means windows goes down
793         //if the value is less than -100 then it will not Redisplay, go to if condition
794         ty -= 4.5000;
795         glutPostRedisplay();
796     }
797     //end of controlling 1st & 2nd window moving
798 }

```

```

800 //controlling 1st & 2nd obstacle
801 //if y axis(of 1st obstacle is less than -180(50+130) than y && yy will reset)
802 if (y < -180) {
803     yy = 0;
804     y = 0;
805 }
806 else {
807     y -= 1;
808     yy -= 1;
809     glutPostRedisplay();
810 }
811 //end of obstacle controlling
812
813 //end of controlAllexceptCar() function
814 }
815
816
817 void spe_key(int key, int x, int y)
818 { //controlling car with up left right
819
820     switch (key) {
821     case GLUT_KEY_UP:
822         gameover = false;
823         //set the global ideal callback
824         glutIdleFunc(controlAllexceptCar);
825         break;
826
827         //start controlling car moving
828
829         //left side move
830     case GLUT_KEY_LEFT:
831         if (cx > 0) {
832             cx -= 16;
833             glutPostRedisplay();
834         }
835         break;
836         //right side move
837     case GLUT_KEY_RIGHT:
838         if (cx < 16) {
839             cx += 16;
840             glutPostRedisplay();
841         }
842         break;
843         //End of car moving
844     default:
845         break;
846     }
847 }

```

```
849 int main(int argc, char* argv[])
850 {
851     glutInit(&argc, argv);
852     glutInitDisplayMode(GLUT_SINGLE | GLUT_RGB);
853     glutInitWindowSize(800, 700);
854     glutInitWindowPosition(300, 0);
855     glutCreateWindow("2-D RACING CAR");
856     init();
857     glutGetModifiers();
858     glutDisplayFunc(display);
859     glutSpecialFunc(spe_key);
860     glutMainLoop();
861
862     return 0;
863 }
864
```

6. Output:

Level 1:



Level 2:



Level 3:



When the game is over:

