IUBAT-International University of Business Agriculture and Technology

A

Computer Graphics Project Report

On

"The Racer- A Car Racing Game"

Submitted in Partial fulfillment of the Requirements for the summer 2017 Semester of the Degree of Bachelor of Computer Science & Engineering by

Name: B.M.ASHIK MAHMUD

ID: 15103052

Section: A

Under the Guidance of Course Instructor

Mr. Krishna Das

Lecturer, Dept. of CSE

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING IUBAT University

IUBAT University

4 Embankment Drive Road, Sector-10, Uttara, 4 Abdullahpur Hwy, Dhaka 1230, Bangladesh.

ABSTRACT

A picture is worth a thousand words goes the ancient Chinese proverb. This has become a cliché in our society after the advent of inexpensive and simple techniques for producing pictures.

Computers have become a powerful medium for the rapid and economical production of pictures. There is virtually no area in which graphical displays cannot be used to some advantage. Graphics provide a so natural means of communicating with a computer that they have become widespread. The fields in which Computer Graphics find their uses are many. Some of them being User Interfaces, Computer Aided Design, Office automation, Desktop Publishing, Plotting of mathematical, scientific or industrial data, Simulation, Art, Presentations, Cartography, to name a few…Here, I have tried to make a simple car racing game. Which run in a street 2D road. There is also some car running on road. Racers need to pass through those car.

The Racer is a simple car game in a 2D plat form. A racer need to control a car by pressing the keyboard key Right and Left key to pass others car. And according to time their will add a game POINT and if the racer car touch with other car it will be Game Over. Simple, colorful.

ACKNOWLEDGMENT

While presenting this Graphics Project on “The Racer- a Car Racing Game”, I feel that we need to work or think different and do some cool things. That’s why I chose to make game that will be different and enjoyable.

I would like to express my heartfelt gratitude to our Course Instructor Mr. Krishna Das Lecturer, Dept. of CSE, for extending his support.

Firstly I thank God for showering his blessings on me. I am grateful to my institution IUBAT University for providing me a congenial atmosphere to carry out the project successfully.

I would also be indebted to my Parents and Elder brother Mamun for their continued moral and material support throughout the course of project and helping me in finalizing the presentation.

My heartfelt thanks to all those have contributed bits, bytes and words to accomplish this Project.

B.M.ASHIK MAHMUD

ID: 15103052

CONTENTS

INTRODUCTION

Although computer graphics is a vast field that encompasses almost any graphical aspect, we are mainly interested in the generation of images of 2 and 3-dimensional scenes. Computer imagery has applications for film special effects, simulation and training, games, medical imagery, flying logos, etc. Computer graphics relies on an internal model of the scene, that is, a mathematical representation suitable for graphical computations. Our race game is fun because of coalition effect. That's an awesome program.

The Racer is a simple car game in a 2D plat form. A racer need to control a car by pressing the keyboard key Right and Left key to pass others car. And according to time their will add a game POINT and if the racer car touch with other car it will be Game Over. Simple, colorful.

It takes a few time to open but after starting and running the car user can play and pass others car.

The highest point is 1000# then it will congratulate you.

LITERATURE SURVEY

CG (Computer graphics) started with the display of data on hardcopy plotters and cathode ray tube screens soon after the introduction of computer themselves. It includes the creation, storage, and manipulation of models and images of objects. These models include physical, mathematical, engineering, architectural, and even conceptual or abstract structures, natural phenomena, and so on. Computer Graphics today is largely interactive- the user controls the contents, structure, and appearance of objects and their displayed images by using input devices, such as keyboard, mouse or touch sensitive panel on the screen. Bitmap graphics is used for user-computer interaction. A Bitmap is an ones and zeros representation of points (pixels, short for picture elements') on the screen. Bitmap graphics provide easy-to-use and inexpensive graphics based applications.

The concept of desktop is a popular metaphor for organizing screen space. By means of a window manager, the user can create, position, and resize rectangular screen areas, called windows, that acted as virtual graphics terminals, each running an application. This allowed users to switch among multiple activities just by pointing at the desired window, typically with the mouse. Graphics provides one of the most natural means of communicating with the computer, since our highly developed 2D and 3D pattern – recognition abilities allow us to perceive and process pictorial data rapidly and efficiently. In many design, implementation, and construction processes, the information pictures can give is virtually indispensable.

Computer graphics is the creation and manipulation of pictures with the aid of computers. It is divided into two broad classes:

- Non-Interactive Graphics.
- Interactive Graphics.

2.1.1 Non-Interactive graphics –

This is a type of graphics where observer has no control over the pictures produced on the screen. It is also called as Passive graphics.

2.1.2 Interactive Graphics-

This is the type of computer graphics in which the user can control the pictures produced. It involves two-way communication between user and computer. The computer upon receiving signal from the input device can modify the displayed picture appropriately. To the user it appears that the picture changes instantaneously in response to his commands. The following fig. shows the basic graphics system:

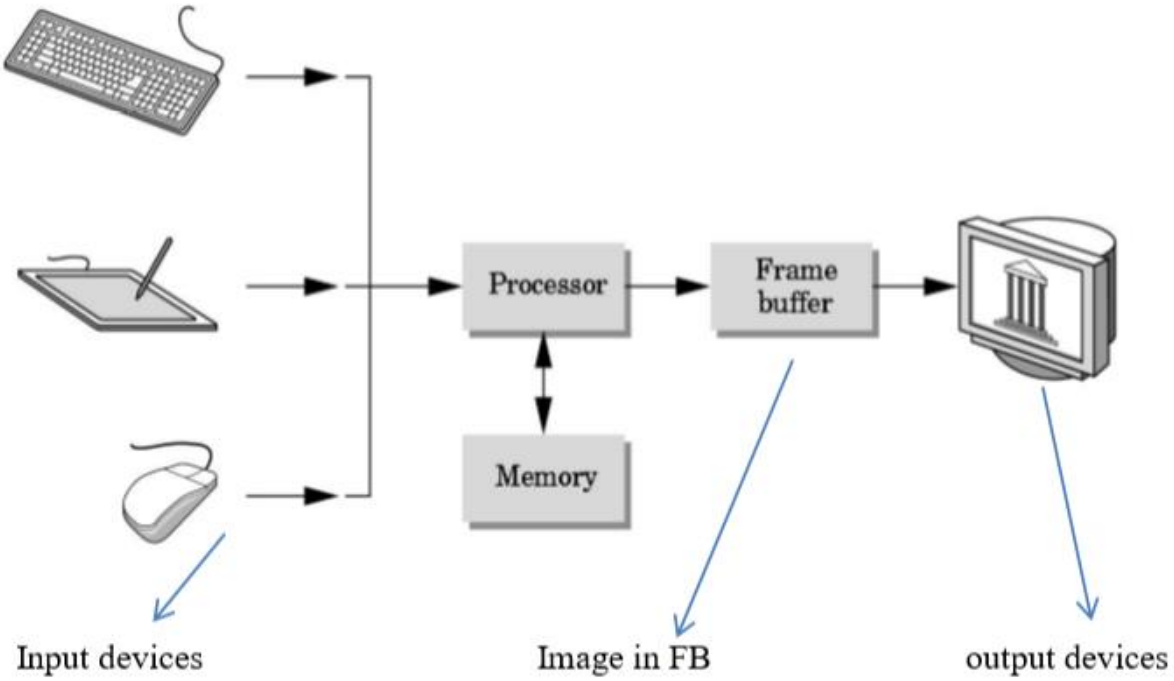

Fig 2.1: Basic Graphics System.

2.2 About OpenGL -

OpenGL is an open specification for an applications program interface for defining 2D and 3D objects. The specification is cross-language, cross-platform API for writing applications that produce 2D and 3D computer graphics. It renders 3D objects to the screen, providing the same set of instructions on different computers and graphics adapters. Thus it allows us to write an application that can create the same effects in any operating system using any OpenGL-adhering graphics adapter.

In Computer graphics, a 3-dimensional primitive can be anything from a single point to an 'n' sided polygon. From the software standpoint, primitives utilize the basic 3dimensional rasterization algorithms such as Brenham's line drawing algorithm, polygon scan line fill, texture mapping and so forth.

OpenGL is a low-level, procedural API, requiring the programmer to dictate the exact steps required to render a scene. OpenGL's low-level design requires programmers to have a good knowledge of the graphics pipeline, but also gives a certain amount of freedom to implement novel rendering algorithms.

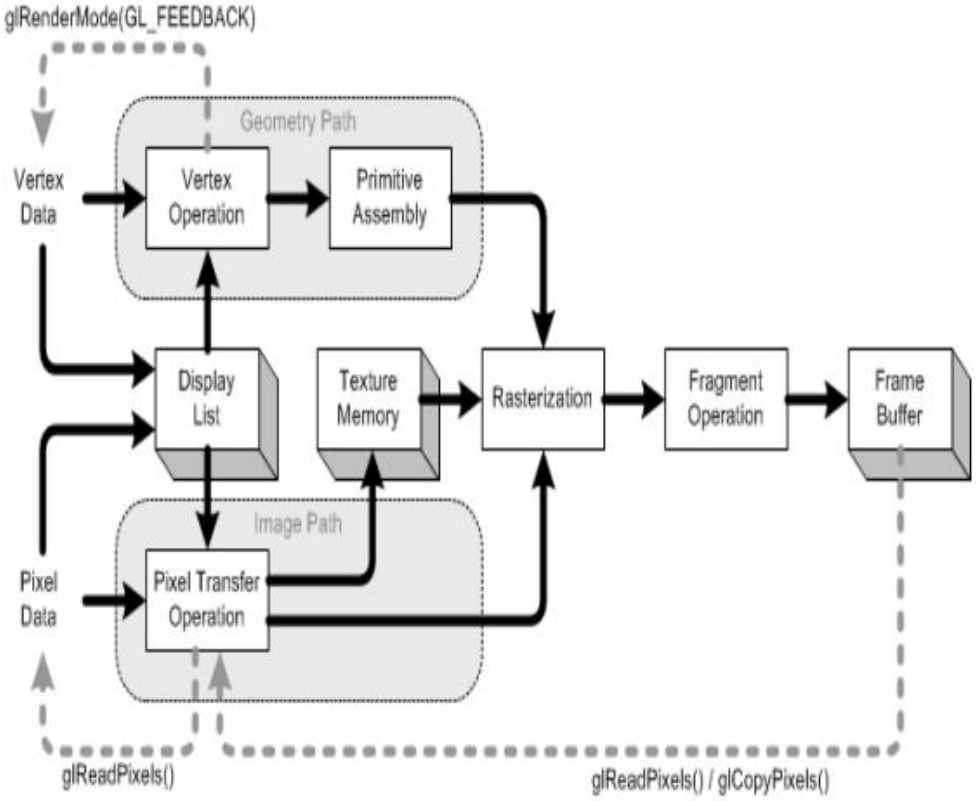

Fig 2.2: The OpenGL rendering pipeline.

2.3 Advantages of OpenGL-

- With different 3D accelerators, by presenting the programmer to hide the complexities of interfacing with a single, uniform API.
- To hide the differing capabilities of hardware platforms, by requiring that all implementations support the full OpenGL feature set (using software emulation if necessary).

Chapter 3

REQUIREMENT SPECIFICATION

3.1 Hardware requirements- Pentium 3 or higher processor, 128 MB or more

RAM, a standard keyboard, an Ubuntu Linux compatible mouse and a VGA monitor. If the user wants to save the created files a secondary storage medium can be used.

3.2 Software requirements- The graphics package has been designed for Windows any version. With the OpenGL library with CodeBlog.

3.3 Development platform- Windows 10,CodeBlog, OpenGL library.

3.4 Language used in coding- C++ language.

3.5 Functional Requirements-

The whole project is divided into many small parts known as functions and these functions would take care of implementing particular parts of this project which makes the programming easy and effective. Each function takes the responsibility of designing the assigned parts hence we combined all the functions at a particular stage to get the final required design. The functions that are used to implement The racer game are:

- void drawOVehicle()- This function draw the other cars from the center of the scan.
- Void drawText() - This function draw all the text. Like core, Game over, Congratulate.
- void drawTextNum() -This function draw the point.
- Void ovpos () - This function select the path of the car. Left and Right side of the road.
- Void drawRoad () - This function will draw the road.
- Void drawDivider() - This function will draw the road divider.
- Void drawVehicle () - This function draw the Racers car.
- void display() - This will call all the functions and repeat every frame.

ALGORITHM DESIGN AND ANALYSIS

4.1 Pseudo Code::

- void display() - This will call all the functions and repeat every frame. It starts the program and the run all the functions by calling by fame to fame. Then
- Void drawRoad () - This function will draw the road.
- Void drawDivider() - This function will draw the road divider.
- void drawOVehicle()- This function draw the other cars from the center of the scan.
- Void drawText() - This function draw all the text. Like core, Game over, Congratulate.
- void drawTextNum() -This function draw the point.
- Void ovpos () - This function select the path of the car. Left and Right side of the road.
- Void drawVehicle () - This function draw the Racers car.

By one by one all the functions can and repeat then again again to the End.

4.2 Analysis:

The functions and their algorithms are as follows:

- Void keyboard():

  This function basically changes the color of the boat as we have implemented the suitable color codes for their respective colors. The colors that the boat can have are red, green, blue, yellow, cyan and magenta.

- Void main ():

  This function puts the whole program together. It says which function to execute first and which one at the end. However, here we have used int main () since eclipse expects the main to have a return value.

IMPLEMENTATION

The Racer- A Car Racing Game can be implemented using some of the OpenGL inbuilt functions along with some user defined functions. The inbuilt OpenGL functions that are used mentioned under the FUNCTIONS USED category. The user defined functions are mentioned under USER DEFINED FUNCTIONS category.

4.1 Functions Used –

- **Void glColor3f (float red, float green, float blue):**

  This function is used to mention the color in which the pixel should appear. The number 3 specifies the number of arguments that the function would take. The 'f' gives the data type float. The arguments are in the order RGB (Red, Green and Blue). The color of the pixel can be specified as the combination of these 3 primary colors.

- **Void glClearColor(int red, int green, int blue, int alpha):**

  This function is used to clear the color of the screen. The 4 values that are passed as arguments for this function are (RED, GREEN, BLUE, ALPHA) where the red green and blue components are taken to set the background color and alpha is a value that specifies depth of the window. It is used for 3D images.

- **void display() :**

  GlutDisplay, glutPostWindowRedisplay — marks the current or specified window as needing to be redisplayed.

- **int main(int argc, char **argv) :**

  GlutMainLoop enters the GLUT event processing loop. This routine should be called at most once in a GLUT program. Once called, this routine will never stop.

4.2 USER DEFINED FUNCTIONS:

- void drawOVehicle()- This function draw the other cars from the center of the scan.
- Void drawText() - This function draw all the text. Like core, Game over, Congratulate.
- void drawTextNum() -This function draw the point.
- Void ovpos () - This function select the path of the car. Left and Right side of the road.
- Void drawRoad () - This function will draw the road.
- Void drawDivider() - This function will draw the road divider.
- Void drawVehicle () - This function draw the Racers car.
- void display() - This will call all the functions and repeat every frame.

Chapter 6

SNAPSHOTS

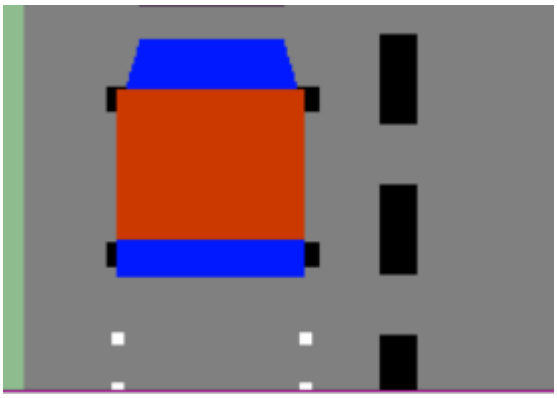

Figure 6.1: Racer Car

Figure 6.2: Opposite Racer Car

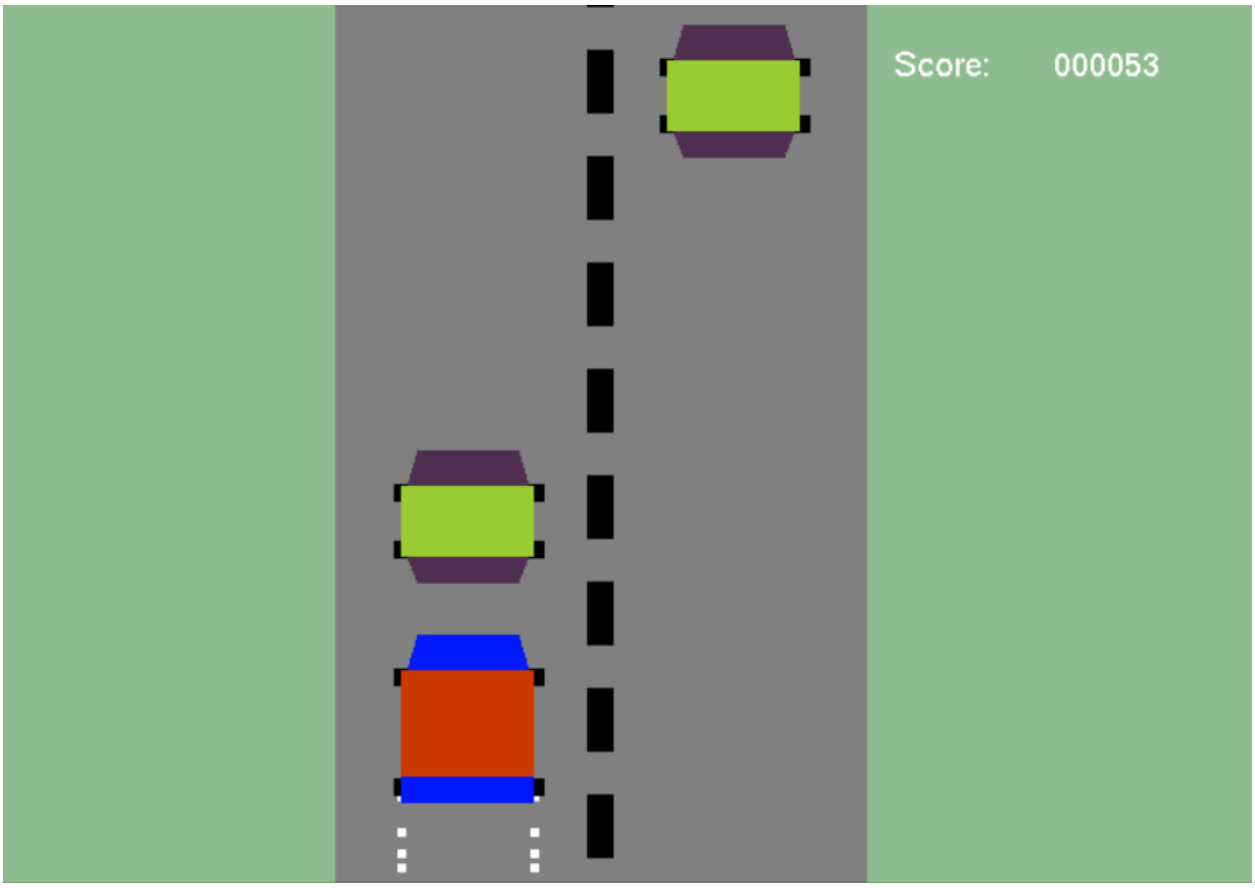

Figure 6.3: Starting of the Game

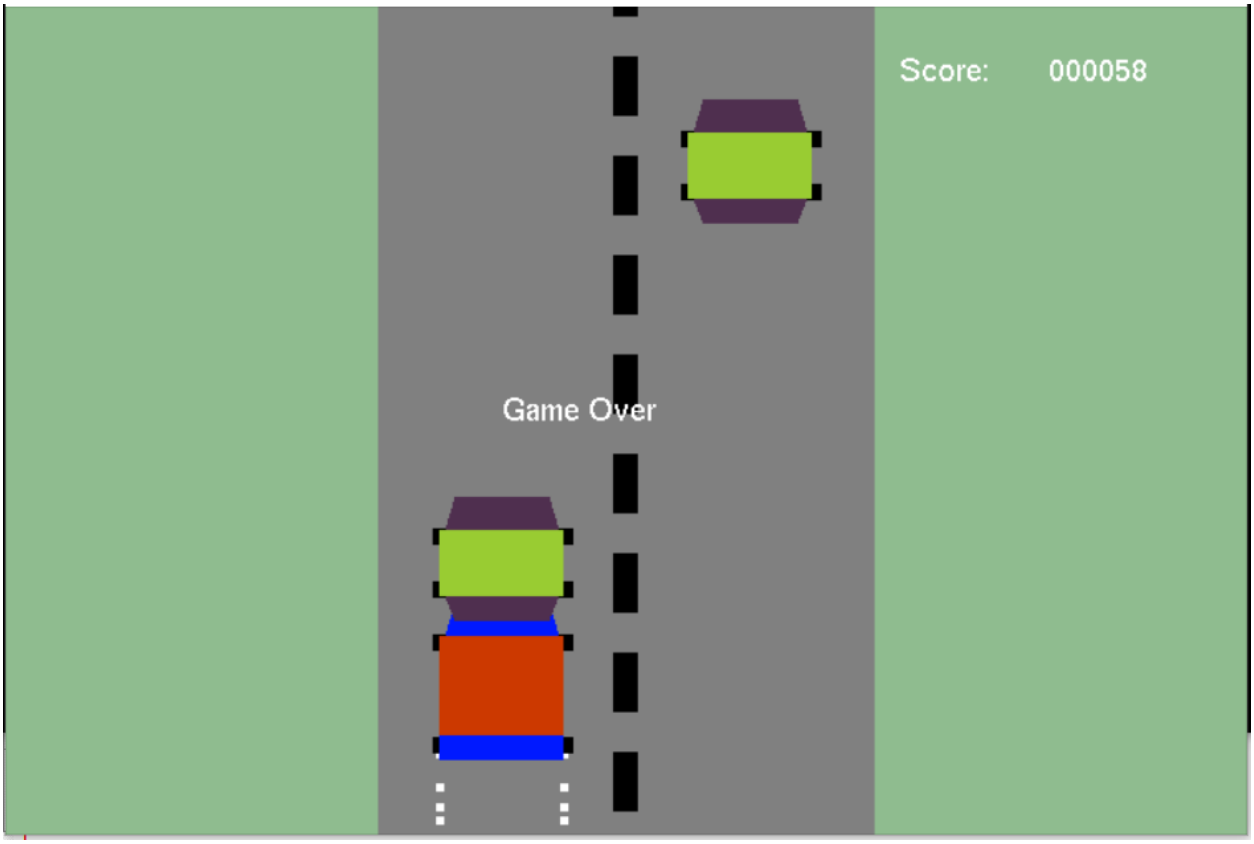

Figure 6.3: End of the Game

Chapter 7

CONCLUSIONS AND FUTURE SCOPE

7.1 General Constraints:

- Simple and colorful game.
- Easy to play.
- Needless to say, the computation of algorithms should also be robust and fast.
- It is built assuming that the standard output device (monitor) supports colors.

7.3 Further Enhancements:

The following are some of the features that can be included in the revised versions of this code are:

- Sounds of running car and end sound.
- Popup message.
- Replay game Option.
- Support for different types of vehicles all moving simultaneously on road.

Chapter 8

Web References:

- www.opengl.org
- www.google.com
- http://www.openglprojects.in/2015/03/2d-car-racing-game-opengl-projects-with-source-code.html#gsc.tab=0

APPENDICES

9.1 Source Code:

```
#include<windows.h>
#include<GL/glut.h>
#include<iostream>//for strlen
#include<conio.h>



int i,q;
int score = 0;//for score counting
int screen = 0;
bool collide = false;//check if car collide to make game over
char buffer[10];

int vehicleX = 200, vehicleY = 100;  // drawing point
int ovehicleX[4], ovehicleY[4];
int divx = 250, divy = 4, movd;
int ox = 250, oy = 250;



void drawText(char ch[],int xpos, int ypos)//draw the text for score and game over
{
   int numofchar = strlen(ch);
   glLoadIdentity (); //replace the current matrix with the identy matrix
   glRasterPos2f( xpos , ypos); // it define the screen position
   for (i = 0; i <= numofchar - 1; i++)
   {

   glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[i]);//font used here, may
use other font also
   }
```

```
}


void drawTextNum(char ch[],int numtext,int xpos, int ypos)//counting the score
{
   int len;
   int k;
   k = 0;
   len = numtext - strlen (ch);
   glLoadIdentity ();
   glRasterPos2f( xpos , ypos);
   for (i = 0; i <=numtext - 1; i++)
   {
   if ( i < len )
      glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18,'0');
   else
   {
      glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[k]);
      k++;
   }

   }
}

//Back Screen
void ovpos()
{
   glClearColor(  0.560784, 0.737255,0.560784,0);
for(i = 0; i < 4; i++)
{
   if(rand() % 2 == 0)
   {
      ovehicleX[i] = 200;
```

```
    }
    else
    {
        ovehicleX[i] = 300;
    }
    ovehicleY[i] = 1100 - i * 240;
}
}

//Road
void drawRoad()
{
    glBegin(GL_QUADS);
        glColor3f(0.50,0.50,0.50);
        glVertex2f(ox - 100, 500);
        glVertex2f(ox - 100, 0);
        glVertex2f(ox + 100, 0);
        glVertex2f(ox + 100, 500);
    glEnd();



}


void drawDivider()//black patch drawn in middle of road
{
    glLoadIdentity();
    glTranslatef(0, movd, 0); // multiply the current matrix by translation matrix x,y,z
    for(i = 1; i <= 10; i++)
    {
    glColor3f(0, 0, 0);
    glBegin(GL_QUADS);
        glVertex2f(divx  - 5, divy * 15 * i + 18);
```

```
        glVertex2f(divx  - 5, divy * 15 * i - 18);
        glVertex2f(divx  + 5, divy * 15 * i - 18);
        glVertex2f(divx  + 5, divy * 15 * i + 18);
    glEnd();
    }


    for(i = 1; i <= 3; i++){
    glPointSize(5.0);
    glBegin(GL_POINTS);//road dots
        glColor3f(1,1,1);
        glVertex2f(vehicleX  - 25, divy * 5 * i + 16);
        glVertex2f(vehicleX  + 25, divy * 5 * i + 16);
        glVertex2f(vehicleX  - 25, divy * 5 * i - 16);
        glVertex2f(vehicleX  + 25, divy * 5 * i - 16);
    glEnd();

    }
    glLoadIdentity();
}


void drawVehicle()//car for racing
{
    glPointSize(10.0);
    glBegin(GL_POINTS);//tire
        glColor3f(0,0,0);
        glVertex2f(vehicleX - 25, vehicleY + 16);//l up
        glVertex2f(vehicleX + 25, vehicleY + 16);//R up
        glVertex2f(vehicleX - 25, vehicleY - 46);//l down
        glVertex2f(vehicleX + 25, vehicleY - 46);//R down
    glEnd();
```

```
    glBegin(GL_QUADS);
        glColor3f( 0.8 ,0.2222 ,0);//middle body
        glVertex2f(vehicleX - 25, vehicleY + 20);//l up
        glVertex2f(vehicleX - 25, vehicleY - 50);//l down
        glVertex2f(vehicleX + 25, vehicleY - 50);//r down
        glVertex2f(vehicleX + 25, vehicleY + 20);//r up
    glEnd();

    glBegin(GL_QUADS);//up body
        glColor3f(0,0.1,1);
        glVertex2f(vehicleX - 23, vehicleY + 20);
        glVertex2f(vehicleX - 19, vehicleY + 40);
        glVertex2f(vehicleX + 19, vehicleY + 40);
        glVertex2f(vehicleX + 23, vehicleY + 20);
    glEnd();

    glBegin(GL_QUADS);//down body
        glColor3f(0,0.1,1);
        glVertex2f(vehicleX - 25, vehicleY - 40);
        glVertex2f(vehicleX - 25, vehicleY - 55);
        glVertex2f(vehicleX + 25, vehicleY - 55);
        glVertex2f(vehicleX + 25, vehicleY - 40);
    glEnd();
 }

void drawOVehicle()//other cars
{

    for(i = 0; i < 3; i++)
    {
    glPointSize(10.0);
    glBegin(GL_POINTS);//tire
        glColor3f(0,0,0);
```

```
    glVertex2f(ovehicleX[i] - 25, ovehicleY[i] + 16);
    glVertex2f(ovehicleX[i] + 25, ovehicleY[i] + 16);
    glVertex2f(ovehicleX[i] - 25, ovehicleY[i] - 16);
    glVertex2f(ovehicleX[i] + 25, ovehicleY[i] - 16);
glEnd();

glBegin(GL_QUADS);
    glColor3f(0.6,0.8 ,0.196078  );//middle body
    glVertex2f(ovehicleX[i] - 25, ovehicleY[i] + 20);
    glVertex2f(ovehicleX[i] - 25, ovehicleY[i] - 20);
    glVertex2f(ovehicleX[i] + 25, ovehicleY[i] - 20);
    glVertex2f(ovehicleX[i] + 25, ovehicleY[i] + 20);
glEnd();

glBegin(GL_QUADS);//up body
    glColor3f(0.309804  ,0.184314 , 0.309804);
    glVertex2f(ovehicleX[i] - 23, ovehicleY[i] + 20);
    glVertex2f(ovehicleX[i] - 19, ovehicleY[i] + 40);
    glVertex2f(ovehicleX[i] + 19, ovehicleY[i] + 40);
    glVertex2f(ovehicleX[i] + 23, ovehicleY[i] + 20);
glEnd();

glBegin(GL_QUADS);//down body
    glColor3f(0.309804  ,0.184314 , 0.309804);
    glVertex2f(ovehicleX[i] - 23, ovehicleY[i] - 20);
    glVertex2f(ovehicleX[i] - 19, ovehicleY[i] - 35);
    glVertex2f(ovehicleX[i] + 19, ovehicleY[i] - 35);
    glVertex2f(ovehicleX[i] + 23, ovehicleY[i] - 20);
glEnd();

ovehicleY[i] = ovehicleY[i] - 8;
```

```
    if(ovehicleY[i] > vehicleY - 25 - 25 && ovehicleY[i] < vehicleY + 25 + 25 &&
ovehicleX[i] == vehicleX)
    {
        collide = true;
    }

    if(ovehicleY[i] < -25)
    {
        if(rand() % 2 == 0)
        {
            ovehicleX[i] = 200;
        }
        else
        {
            ovehicleX[i] = 300;
        }
        ovehicleY[i] = 600;
    }
    }
}

void Specialkey(int key, int x, int y)//allow to use navigation key for movement of car
{
switch(key)
{
case GLUT_KEY_LEFT:vehicleX = 200;
            break;
case GLUT_KEY_RIGHT:vehicleX = 300;
            break;

}
glutPostRedisplay();
}
```

```
void init()
{
glMatrixMode(GL_PROJECTION);
glLoadIdentity();
gluOrtho2D(0, 500, 0, 500);
glMatrixMode(GL_MODELVIEW);
}


void display()
{
   glClear(GL_COLOR_BUFFER_BIT);
   drawRoad();
   drawDivider();
   drawVehicle();
   drawOVehicle();
   movd = movd - 14;
   if(movd < - 60)
   {
      movd = 0;
   }
   score = score + 1;
   if(score>1000){          //Congratulations!
      glColor3f(1,1,1);
      drawText("Congratulations!", 200,250);
      glutSwapBuffers();
      getch();
   }

   glColor3f(1,1,1);
   drawText("Score:", 360,455);
   itoa (score, buffer, 10);
   drawTextNum(buffer, 6, 420,455);
```

```
    glutSwapBuffers();
    for(q = 0; q<= 10000000; q++){;}
    if(collide == true)
    {
        glColor3f(1,1,1);
        drawText("Game Over", 200,250);
        glutSwapBuffers();
        getch();
    }
}


int main(int argc, char **argv)
{
    glutInit(&argc,argv);
    glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
    glutInitWindowPosition(200,200);
    glutInitWindowSize(750,500);
    glutCreateWindow("The Racer");
    ovpos();
    init();
    glutDisplayFunc(display);
          glutSpecialFunc(Specialkey);
    glutIdleFunc(display);
    glutMainLoop();
    return 0;
}



////////////////////////////////////  The End  :D     ////////////////////////////
```