Department of Information Technology

National Institute of Technology Karnataka, Surathkal

Computer Graphics (IT254)

Mini Project

Title- 2D Car Racing Game

Submitted by

Supriya Sahoo 15IT145

Akriti Kumari 15IT107

Ankita Bhalavi 15IT108

Vandana Baidya 15IT250

Submitted to

Ms. R Priyadarshini
(Course Instructor)

April 2017

CERTIFICATE

This is to certify that the project entitled **"**2D Car Racing Game**"** has been presented by Supriya Sahoo, Akriti Kumari, Ankita Bhalavi, Vandana Baidya, students of second year, B.Tech (IT), Department of Information Technology, National Institute of Technology Karnataka , Surathkal, on April 6, 2017, during the even semester of the academic year 2016 - 2017, in partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology at NITK, Surathkal.

Place: NITK, Surathkal

Date: 06-04-2017 (Signature of Course instructor)

ACKNOWLEDGEMENT

This project would not have been possible without the kind support and help of many individuals and organizations. We would like to extend our sincere thanks to all of them. We are highly indebted to the Department of Information Technology, NITK for their guidance and constant supervision as well as for providing necessary information and resources for the project and also for their support in completing the project. We would like to express our gratitude towards our teacher R. Priyadarshini for her kind co-operation and encouragement which helped us in completion of this project. Our thanks and appreciations also go to our group members in developing the project and people who have willingly helped us out with their abilities.

CONTENTS

SYNOPSIS

Game theory is an interesting field of exploration. Many works have been already done in this field. By this mini project in 2D car racing , we aim to further test our knowledge in game design and implementing various graphical interfaces.

We are going to implement a 2D car racing game in C using openGL interface. And the platform will be codeblocks. We planned to use various basic graphics functions to establish our goal to implement a visual appealing ,single player simple 2D game. Since we are yet to specialize in the field of computer graphics,this game is our first attempt to convert ideas into a working application.

INTRODUCTION

2D car racing game is a simple 2D game developed using openGL. In this game, there is a car and which moves forward in a road using up arrow key. The sides of the road on which the car is moving can be changed using keyboard side arrow keys and its speed can be decreased using down arrow key. The score keeps on increasing till the car moves forward. When the car encounters other obstacles present in the road, the game ends. At the end the score is displayed.

PROBLEM STATEMENT

This project is about Car Racing Game where we developed a “2D Car Racing Game” in which creation of a motion graphic piece for a racing game through openGL functions have done with 2D effects. Also there is a track as in real formula one race game while in this, we have simple road.

OBJECTIVE

- The aim of this mini project is to implement the car racing game in 2D.
- Develop an idea to design some graphical applications.
- This graphics is developed as a model which shows how to combine color, lighting effects, transformations of an object, viewing, increasing and decreasing the objects in real time with OpenGL.
- This project can be used to study the various functionalities of computer graphics.

METHODOLOGY

The methodology of this project includes all the user defined functions used in the code for creating the vehicles and movement of these vehicles. These are:

1.drawText():

This function has three arguments which are score stored in an array, x co-ordinate position and y co-ordinate position where the function should be displayed. It displays the text for score using **glRasterPos2f(xpos , ypos)** function.

2. drawRoad():

This function draws a black colored rectangle of 200x500 dimension which displays the road. The road contains the dividers, the racing car and opponent cars.

3. drawDivider():

This function when called draws 10 white colored 10x36 rectangles using translation at the middle of road which represents the divider of road.

4. drawVehicle():

This function is used to draw the vehicle which has four parts : tires, middle body , upper body and lower body. Tires are the points and the body parts are quadrilaterals of different shapes and sizes. The middle body is 50x40 while upper and lower body are trapezium of (46,38)x20 .

5. drawOVehicle():

This function draws the opponent vehicles and detects the collision. The opponent vehicle has same appearance as the racing car only differs in color.

It detects the collision when racing car and opponent vehicle has same x co-ordinate (horizontal colliding) . At the same time when racing car collides opponent vehicles from behind or when opponent vehicle collides with the racing car.

If the collision is not detected then more opponent cars are drawn otherwise collision value is returned true.

6. ovpos():

This function returns the position for the opponent vehicles which is random.

7. specialKey():

This function is used to enable navigation keys for movement of car. The left and right keys are used for left and right movement of the car respectively. While up and down navigation keys are used for the acceleration and deceleration of the racing car respectively.

8. display():

This is the main display function where all the other functions are called. The score value is increased by one until the car has not collided. If the collision has return value true then exit status is returned.

SYSTEM SPECIFICATION

Operating system : Windows 7 or above

Platform : Codeblocks

Interface : OpenGL

WORKS DONE

Basically it is based on the logic that the y co-ordinates of opponent cars are decreased constantly and we can change the x co-ordinate of racing car using left and right navigation keys.

The acceleration of the racing car is shown by moving the dividers backward (decreasing y co-ordinates) constantly. Similarly, deceleration of the car is shown by constantly moving the dividers in positive y direction.

The left and right movement of car is achieved using specialKey() function. When left key is pressed then the x co-ordinate of car changes to 200 i.e left of divider and when right key is pressed then the x co-ordinate of car changes to 300 i.e right of divider.

The movement of the cars is shown by changing the y co-ordinates of dividers constantly which is achieved using glutPostReDisplay and gutIdleFunc.

The game is over when any type of collision occurs.

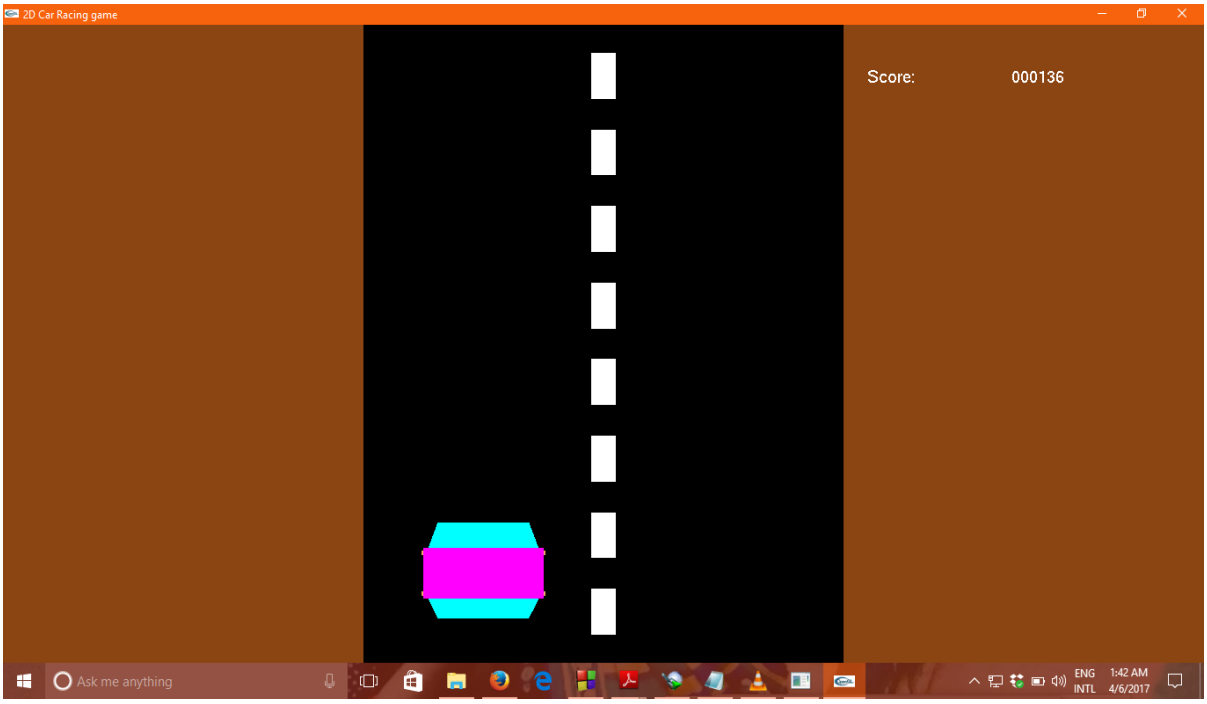

FIG1:game area showing racing car on the road with the score.

FIG2: game area showing collision of racing car with the opponent car.

FUTURE WORKS

We can convert this simple 2D car racing game into 3D model and also make it multiplayer in the future. There are additional features we would like to implement, to both improve performance and the visual quality of the game. Examples of features that could be added are, a height mapped terrain, as this would decrease the time it takes to load the map. Another example is adding a cube map to the car, as reflections would make the car look more like a part of the environment. These are just some of the features that would make a positive impact to the game and can hopefully be implemented in the near future.

RESULT AND DISCUSSION

The result of the 1 month development time, is a racing game that is both playable and visually appealing. There are many features to this result; all contributing to the final look and feel of the game. The most impactful features are: animation and scoring table. As these features provide a sense of speed, logical movement and something to look at while tackling the challenge of the race. The other implementations are as important, but are hard to distinguish from each other in terms of effect, as they together contribute to the visual quality of the rendered image.

CONCLUSION

The resulting game is in our opinion well realized, in perspective to our set goals. Of course we would like to have added more content and features, but due to time constraints and team size we had to prioritize essential components that we thought would contribute the most to the game. Therefore, game development in a small and rather inexperienced team, over a short period of time is nothing we would recommend. However, if the game idea is clear, and if the development is done with respect to the physics library, rather than the opposite; there is still a chance of creating a visually pleasing, as well as an entertaining game. Probably the greatest insight of the development team, is that proper research is crucial, with respect to time, as well as the final result. For instance, by taking the time to analyze the various rendering algorithms, we might have ended up with a deferred renderer in an earlier stage of the project. Thus, adding more visual effects might have been possible. However, doing proper time estimation for the implementation is still considered difficult, due to lack of experience.

REFERENCE

- Computer Graphics C Version by Donald Hearn & M Pauline Baker II Edition.
- https://www.tutorialspoint.com/listtutorial/Introduction-to-OpenGL/2339
- https://www.khronos.org/opengl/wiki/Skeletal_Animation
- https://en.wikipedia.org/wiki/OpenGL

APPENDIX

CODE: 2D Car Racing Game

```
#include <windows.h>
#ifdef __APPLE__
#include <GLUT/glut.h>
#else
#include <GL/glut.h>
#endif
#include <conio.h>
#include <stdlib.h>
#include<iostream>//for strlen
#include<stdlib.h>
 int i,q;
int score = 0;
int screen = 0;
bool collide = false;
char buffer[10];
 int vehicleX = 200, vehicleY = 70;
int ovehicleX[4], ovehicleY[4];
int divx = 250, divy = 4, movd;

 void drawText(char ch[],int xpos, int ypos)
{
   int numofchar = strlen(ch);
   glLoadIdentity();
   glRasterPos2f( xpos , ypos);
   for (i = 0; i <= numofchar - 1; i++)
   {
    glutBitmapCharacter(GLUT_BITMAP_HELVETICA_18, ch[i]);
```

```
    }
}
 void drawTextNum(char ch[],int numtext,int xpos, int ypos)
{
    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++;
    }
  }
}

void ovpos()
{
    glClearColor(0.545, 0.271, 0.075,0);
for(i = 0; i < 4; i++)
{
    if(rand() % 2 == 0)
    {
        ovehicleX[i] = 200;
    }
```

```
    else
    {
        ovehicleX[i] = 300;
    }
    ovehicleY[i] = 1000 - i * 160;
}
}

void drawRoad()
{
    glBegin(GL_QUADS);
        glColor3f(0,0,0);
        glVertex2f(250 - 100, 500);
        glVertex2f(250 - 100, 0);
        glVertex2f(250 + 100, 0);
        glVertex2f(250 + 100, 500);
    glEnd();
}

void drawDivider()
{
    glLoadIdentity();
    glTranslatef(0, movd, 0);
    for(i = 1; i <= 10; i++)
    {
    glColor3f(1,1,1);
    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();
    }
    glLoadIdentity();
}

void drawVehicle()
{
    glPointSize(10.0);
    glBegin(GL_POINTS);
        glColor3f(0.957, 0.643, 0.376);
        glVertex2f(vehicleX - 25, vehicleY + 16);
        glVertex2f(vehicleX + 25, vehicleY + 16);
        glVertex2f(vehicleX - 25, vehicleY - 16);
        glVertex2f(vehicleX + 25, vehicleY - 16);
    glEnd();
    glBegin(GL_QUADS);
        glColor3f(1,0,1);
        glVertex2f(vehicleX - 25, vehicleY + 20);
        glVertex2f(vehicleX - 25, vehicleY - 20);
        glVertex2f(vehicleX + 25, vehicleY - 20);
        glVertex2f(vehicleX + 25, vehicleY + 20);
    glEnd();
   glBegin(GL_QUADS);
        glColor3f(0,.999,1);
        glVertex2f(vehicleX - 23, vehicleY + 20);
        glVertex2f(vehicleX - 19, vehicleY + 40);
        glVertex2f(vehicleX + 19, vehicleY + 40);
        glVertex2f(vehicleX + 23, vehicleY + 20);
    glEnd();
    glBegin(GL_QUADS);
        glColor3f(0,.999,1);
```

```
        glVertex2f(vehicleX - 23, vehicleY - 20);
        glVertex2f(vehicleX - 19, vehicleY - 35);
        glVertex2f(vehicleX + 19, vehicleY - 35);
        glVertex2f(vehicleX + 23, vehicleY - 20);
    glEnd();
}

void drawOVehicle()
{
  for(i = 0; i < 1; i++)
    {
    glPointSize(10.0);
    glBegin(GL_POINTS);//tire
        glColor3f(0,1,1);
        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(0.566, 0.834, 0);
        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);
        glColor3f(1,0,0);
        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);
        glColor3f(1,0,0);
        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)
{
switch(key)
```

```
{
case GLUT_KEY_UP:
   for(i = 0; i <4; i++)
          {
           ovehicleY[i] = ovehicleY[i] - 10;
          }
          movd = movd - 20;
          break;
case GLUT_KEY_DOWN:
   for(i = 0; i <4; i++)
          {
           ovehicleY[i] = ovehicleY[i] + 10;
          }
          movd = movd + 20;
          break;
case GLUT_KEY_LEFT:vehicleX = 200;
          break;
case GLUT_KEY_RIGHT:vehicleX = 300;
          break;
 }

glutPostRedisplay();
}

void Normalkey(unsigned char key, int x, int y)
{
   switch(key)
   {
   case 27:exit(0);
   }
}
```

```
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 - 16;
   if(movd < - 60)
   {
      movd = 0;
   }
 score = score + 1;
   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(0,0,0);
        drawText("Game Over", 200,250);
       glutSwapBuffers();
        exit(0);
    }

}

int main(int argc, char **argv)
{
   glutInit(&argc,argv);
   glutInitDisplayMode(GLUT_RGB|GLUT_DOUBLE);
   glutInitWindowPosition(100,100);
   glutInitWindowSize(800,500);
   glutCreateWindow("2D Car Racing game");
   ovpos();
   init();
   glutDisplayFunc(display);
   glutSpecialFunc(Specialkey);
   glutKeyboardFunc(Normalkey);
   glutIdleFunc(display);
   glutMainLoop();
 }
```