Sardar Vallabhbhai National Institute Of Technology

Computer Graphics Project Report

Dodge Ball

13/06/2020

Group No :- 10

Herat Makwana(U17CO015)

Jiten Vagadia(U17CO030)

```
/**
* Computer Graphics Project
* 3d Version of Project
*
* Subject : Save the ball going towards your head
*
*
* Herat Makwana(U17CO015)
* Jiten Vagadiya(U17CO030)
*/
#ifdef __WIN32__
#include<windows.h>
#endif
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;

float xcenter = 0.0, ycenter = 0.0, zcenter = 0.0;
bool gone = false;
bool xdir = true, ydir = true, zdir = true;

// float ballAt[3] = {1.5, 0.0, -5.8};
float plateAt[3] = {0.0, 0.0, -3.0};
int score = 0;
int livesleft = 5;
int speed = 0.02;
```

```
void RenderString(float x, float y, void *font, int player)
{
   glColor3f(0.0, 0.0, 1.0);
   glRasterPos2f(x, y);
   stringstream ss;
   ss<<"Score ";
   ss<<score;
   string str = ss.str();
   for (unsigned int i = 0; i < str.length(); i++)
       glutBitmapCharacter(font, str[i]);
   glRasterPos2f(x,y-0.2);
   stringstream ss2;
   ss2<<"Lives Left ";
   ss2<<livesleft;
   string str2 = ss2.str();
   for (unsigned int i = 0; i < str2.length(); i++)
       glutBitmapCharacter(font, str2[i]);
}
void RenderStringOver(float x, float y, void *font, int player)
{
   glColor3f(1.0, 0.0, 0.0);
   glRasterPos2f(x, y);
   string str = "Game Over :(";
   for (unsigned int i = 0; i < str.length(); i++)
       glutBitmapCharacter(font, str[i]);
```

```
}


void display()
{
   if(livesleft){
   glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
   glLoadIdentity();
   glTranslatef(-2.0, -1.5, 0.0);
   glBegin(GL_QUADS);
   glColor3f(0.4, 0.4, 0.4);
   glVertex3f(0.0, 0.0, -3.0);
   glVertex3f(4.0, 0.0, -3.0);
   glVertex3f(4.0, 0.0, -6.0);
   glVertex3f(0.0, 0.0, -6.0);
   glEnd();
   glBegin(GL_QUADS);
   glColor3f(0.6, 0.6, 0.6);
   glVertex3f(0.0, 0.0, -6.0);
   glVertex3f(4.0, 0.0, -6.0);
   glVertex3f(4.0, 3.0, -6.0);
   glVertex3f(0.0, 3.0, -6.0);
   glEnd();
   glBegin(GL_QUADS);
   glColor3f(0.7, 0.3, 0.3);
   glVertex3f(0.0, 0.0, -3.0);
   glVertex3f(0.0, 0.0, -6.0);
   glVertex3f(0.0, 3.0, -6.0);
```

```
glVertex3f(0.0, 3.0, -3.0);

glEnd();

glBegin(GL_QUADS);

glColor3f(0.7, 0.3, 0.3);

glVertex3f(4.0, 0.0, -3.0);

glVertex3f(4.0, 0.0, -6.0);

glVertex3f(4.0, 3.0, -6.0);

glVertex3f(4.0, 3.0, -3.0);

glEnd();

glColor3f(1.0, 0.0, 0.0);

glLoadIdentity();

if (gone)

{

    glTranslatef(xcenter, ycenter, zcenter);

    glutSolidSphere(0.2, 30, 36);

}

glLoadIdentity();

glTranslatef(plateAt[0], plateAt[1], plateAt[2]);

glBegin(GL_QUADS);

glColor3f(0.3, 0.7, 0.7);

glVertex3f(-0.3, -0.3, 0.0);

glVertex3f(-0.3, 0.3, 0.0);

glVertex3f(0.3, 0.3, 0.0);

glVertex3f(0.3, -0.3, 0.0);

glEnd();

glLoadIdentity();

glTranslatef(-3.0,1.5,-3.0);
```

```
   RenderString(0.0f, 0.0f, GLUT_BITMAP_TIMES_ROMAN_24, 1);
   glutSwapBuffers();
   }
   else{
       gone = false;
       glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);
       glLoadIdentity();
       glTranslatef(-3.0,1.5,-3.0);
       RenderString(0.0f, 0.0f, GLUT_BITMAP_TIMES_ROMAN_24, 1);
       glTranslatef(3.0,-1.5,0.5);
       RenderStringOver(0.0f, 0.0f, GLUT_BITMAP_TIMES_ROMAN_24, 1);
       glutSwapBuffers();
   }
}
void reshape(int w, int h)
{
   glViewport(0, 0, w, h);
   glMatrixMode(GL_PROJECTION);
   glLoadIdentity();
   gluPerspective(60, (float)w / (float)h, 1.0, 10.0);
   glMatrixMode(GL_MODELVIEW);
}
void initialize()
{
   glClearColor(0.0, 0.0, 0.0, 1.0);
   glEnable(GL_DEPTH_TEST);
   glDepthFunc(GL_LEQUAL);
```

```
}


void timer(int)
{
    glutPostRedisplay();
    glutTimerFunc(1000 / 60, timer, 0);

    if (xcenter <= -1.8 || xcenter >= 1.8)
    {
        xdir = xdir ? false : true;
    }
    if (ycenter <= -1.3 || ycenter >= 1.3)
    {
        ydir = ydir ? false : true;
    }
    if (zcenter <= -5.8)
    {
        zdir = zdir ? false : true;



        score++;
    }
    if (zcenter >= -3.2 && xcenter < plateAt[0] + 0.3 && xcenter >
plateAt[0] - 0.3 && ycenter < plateAt[1] + 0.3 && ycenter >
plateAt[1] - 0.3)
    {
        zdir = zdir ? false : true;
    }
    else if (zcenter > -3.2)
    {
        if (gone && livesleft!=0)
        {
            livesleft--;

            gone = false;
        }
    }

    xdir ? xcenter += 0.02 : xcenter -= 0.02;
    ydir ? ycenter += 0.02 : ycenter -= 0.02;
    zdir ? zcenter += 0.02 : zcenter -= 0.02;
}
void processNormalKeys(unsigned char key, int x, int y)
{
    switch (key)
    {
    case ' ':
        if (!gone && livesleft!=0)
        {
            gone = true;
            xcenter = 0.0;
            ycenter = 0.0;
            zcenter = -5.8;
```

```
           xdir = true;

           ydir = true;

           zdir = false;

       }

       break;

   case 'a':

       plateAt[0] = max(-1.7, plateAt[0] - 0.2);

       break;

   case 'd':

       plateAt[0] = min(1.7, plateAt[0] + 0.2);

       break;

   case 'w':

       plateAt[1] = min(1.2, plateAt[1] + 0.2);

       break;

   case 's':

       plateAt[1] = max(-1.2, plateAt[1] - 0.2);

       break;

   }

}


void processSpecialKeys(int key, int x, int y)

{

   switch (key)

   {

   case GLUT_KEY_LEFT:

       plateAt[0] = max(-1.7, plateAt[0] - 0.2);

       break;
```

```
    case GLUT_KEY_RIGHT:
        plateAt[0] = min(1.7, plateAt[0] + 0.2);
        break;
    case GLUT_KEY_UP:
        plateAt[1] = min(1.2, plateAt[1] + 0.2);
        break;
    case GLUT_KEY_DOWN:
        plateAt[1] = max(-1.2, plateAt[1] - 0.2);
        break;
    case GLUT_KEY_PAGE_UP:
        plateAt[1] = min(1.2, plateAt[1] + 0.2);
        plateAt[0] = min(1.7, plateAt[0] + 0.2);
        break;
    case GLUT_KEY_PAGE_DOWN:
        plateAt[1] = max(-1.2, plateAt[1] - 0.2);
        plateAt[0] = min(1.7, plateAt[0] + 0.2);
        break;
    case GLUT_KEY_HOME:
        plateAt[1] = min(1.2, plateAt[1] + 0.2);
        plateAt[0] = max(-1.7, plateAt[0] - 0.2);
        break;
    case GLUT_KEY_END:
        plateAt[1] = max(-1.2, plateAt[1] - 0.2);
        plateAt[0] = max(-1.7, plateAt[0] - 0.2);
        break;
    }
}
```

```
int main(int argc, char *argv[])
{

   glutInit(&argc, argv);
   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA | GLUT_DEPTH);
   glutInitWindowSize(1280, 720);
   glutCreateWindow("3D Ball");

   initialize();
   glutDisplayFunc(display);
   glutReshapeFunc(reshape);
   glutTimerFunc(10, timer, 0);
   glutKeyboardFunc(processNormalKeys);
   glutSpecialFunc(processSpecialKeys);

   glutMainLoop();

   return 0;
}
```

ABOUT :

A Mini Project about computer graphics using OpenGL. This is a simple game about saving balls which come towards us. In this game there is one palette provided by using that we have to save ball which is coming towards us reflecting continuously.

HOW TO USE :

Normal Keys to Move Palette :

- **UP_ARROW || W**
- **DOWN_ARROW || S**
- **LEFT_ARROW || A**
- **RIGHT_ARROW || D**

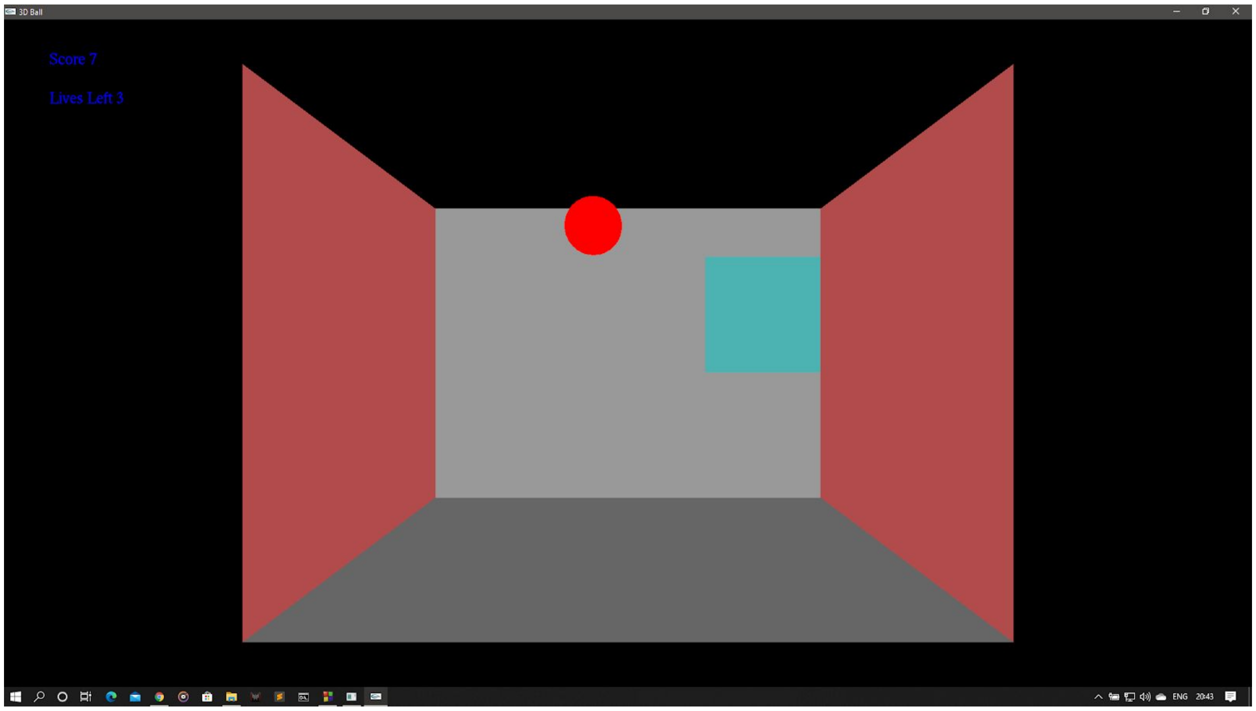
Special Keys to Move Palette :

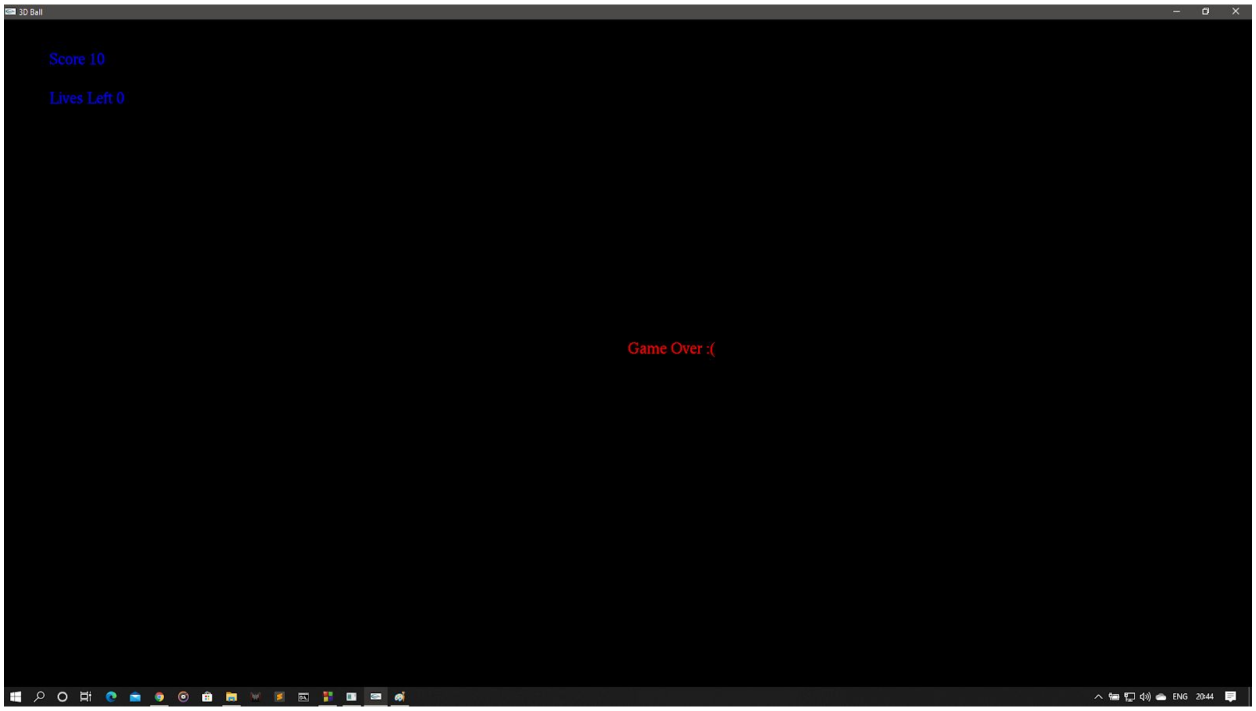
- **PAGE_UP(UP & RIGHT)**
- **PAGE_DOWN(DOWN & RIGHT)**
- **HOME(UP & LEFT)**
- **END(DOWN & LEFT)**

To Throw the ball

- **SPACEBAR**

Dodge Ball


3D Ball
Score 7
Lives Left 3
ENG
20:43


3D Ball
Score 10
Lives Left 0
Game Over :(
ENG
20:44

```
/**
* Computer Graphics project
* 2d Version Of Project
*
* Subject : Save the ball going towards your head
*
*
* Herat Makwana(U17CO015)
* Jiten Vagadiya(U17CO030)
*/
#ifdef __WIN32__
#include<windows.h>
#endif
#include <GL/glut.h>
#include <bits/stdc++.h>
using namespace std;
#define PI 3.142857
#define MAX_SCORE 5
GLfloat twicePI = 2.0f * PI;

//Initialization Variables
struct Ball
```

```
{
   int xcenter, ycenter;
   bool ballboard;
   bool gone;
   bool xdir, ydir;
   Ball() : xcenter(0), ycenter(0), xdir(0), ydir(0), gone(false),
ballboard(0) {}
};

Ball ball;
int x, y;
int nextfree = 0;
int boardblue = 0;
int boardred = 0;
int redScore = 0;
int blueScore = 0;
bool isThereBall = false;
int lastThrown = 1;
int winid = 0;
int speed = 3;
int hits = 0;

// Initialization window for it`s characteristics
void myInit(void)
{
   glClearColor(0.0, 0.0, 0.0, 1.0);
   glColor3f(1.0f, 0.0f, 0.0f);


   glPointSize(1.0);

   glMatrixMode(GL_PROJECTION);

   glLoadIdentity();


   gluOrtho2D(-620.0, 620.0, -340.0, 340.0);

}


// Keyboard Event function

void keyboard(int key, int x, int y)

{

   switch (key)

   {

   case GLUT_KEY_LEFT:

       boardblue = max(-519, boardblue - 50);

       break;

   case GLUT_KEY_RIGHT:

       boardblue = min(520, boardblue + 50);

       break;

   case GLUT_KEY_F1:

       boardred = max(-519, boardred - 50);

       break;

   case GLUT_KEY_F3:

       boardred = min(520, boardred + 50);

       break;

   case GLUT_KEY_UP:
       if (!isThereBall && lastThrown == 1)

       {

           isThereBall = true;

           ball.xcenter = boardblue;

           ball.ycenter = -280;

           ball.ballboard = 0;

           ball.gone = true;

           ball.xdir = true;

           ball.ydir = true;

           lastThrown = 0;

           hits=0;

       }

       break;

   case GLUT_KEY_DOWN:

       if (!isThereBall && lastThrown == 1)

       {

           isThereBall = true;

           ball.xcenter = boardblue;

           ball.ycenter = -280;

           ball.ballboard = 0;

           ball.gone = true;

           ball.xdir = false;

           ball.ydir = true;

           lastThrown = 0;

           hits=0;

       }

       break;
```

Dodge_Ball

```
case GLUT_KEY_F2:
    if (!isThereBall && lastThrown == 0)
    {
        isThereBall = true;
        ball.xcenter = boardred;
        ball.ycenter = 280;
        ball.ballboard = 1;
        ball.gone = true;
        ball.xdir = false;
        ball.ydir = false;
        lastThrown = 1;
        hits=0;
    }
    break;
case GLUT_KEY_F4:
    if (!isThereBall && lastThrown == 0)
    {
        isThereBall = true;
        ball.xcenter = boardred;
        ball.ycenter = 280;
        ball.ballboard = 1;
        ball.gone = true;
        ball.xdir = true;
        ball.ydir = false;
        lastThrown = 1;
        hits=0;
    }
```

```
        break;
    }
}
void RenderString(float x, float y, void *font,int player)
{

    player ==1 ? glColor3f(0.0, 0.0, 1.0) : glColor3f(1.0, 0.0, 0.0);
    glRasterPos2f(x, y);
    stringstream ss;
    ss << (player==1 ? "blue " : "red ");
    ss << (player==1 ? blueScore : redScore);
    string str = ss.str();
    string showstring = "Player "+ str;
    for(unsigned int i=0;i<showstring.length();i++)
        glutBitmapCharacter(font, showstring[i]);
}
void WinString(float x, float y, void *font,int player)
{

    player ==1 ? glColor3f(0.0, 0.0, 1.0) : glColor3f(1.0, 0.0, 0.0);
    glRasterPos2f(x, y);
    stringstream ss;
    ss << (player==1 ? "blue " : "red ");
    ss << "wins\nscore ";
    ss << (player==1 ? blueScore : redScore);
    string str = ss.str();
    string showstring = "Player "+ str;
```

```
   for(unsigned int i=0;i<showstring.length();i++)
       glutBitmapCharacter(font, showstring[i]);
}
//Display Function
void display(void)
{
   glClear(GL_COLOR_BUFFER_BIT);
   ball.ballboard == 0 ? glColor3f(0.0, 0.0, 1.0) : glColor3f(1.0, 0.0,
0.0);
   if (ball.gone)
   {
       int change = ball.ydir;
       glBegin(GL_POINTS);
       for (float j = 0; j < (2 * PI); j += 0.1)
       {
           x = ball.xcenter + 5 * cos(j);
           y = ball.ycenter + 5 * sin(j);
           if (x >= 620)
               ball.xdir = false;
           else if (x <= -620)
               ball.xdir = true;
           else if (y <= -295 && (boardblue + 100 > ball.xcenter) &&
(boardblue - 100 < ball.xcenter))
           {
               ball.ydir = true;

           }
```

```
            else if (y >= 295 && (boardred + 100 > ball.xcenter) &&
(boardred - 100 < ball.xcenter))
            {
                ball.ydir = false;


            }
            else if (y >= 340)
            {
                isThereBall = false;
                if(ball.gone){
                blueScore++;
                ball.gone = false;}
                if (blueScore == MAX_SCORE)
                {
                    printf("player 1 wins");
                    WinString(-100,50,GLUT_BITMAP_TIMES_ROMAN_24,1);
                    glutDestroyWindow(winid);
                    exit(0);
                }
            }
            else if (y <= -340)
            {
                isThereBall = false;
                if(ball.gone){
                ball.gone = false;
                redScore++;
                }
                if (redScore == MAX_SCORE)
                {
                    printf("player 2 wins");
                    WinString(-100,50,GLUT_BITMAP_TIMES_ROMAN_24,0);
                    glutDestroyWindow(winid);
                    exit(0);
                }
            }
            glVertex2i(x, y);
        }
        if(change!=ball.ydir){
            hits++;
        }
        glEnd();

    }
    glColor3f(0.0, 0.0, 1.0);
    glBegin(GL_LINE_LOOP);
    glVertex2i(boardblue + 100, -315);
    glVertex2i(boardblue + 100, -295);
    glVertex2i(boardblue - 100, -295);
    glVertex2i(boardblue - 100, -315);
    glEnd();
    glColor3f(1.0, 0.0, 0.0);
    glBegin(GL_LINE_LOOP);
    glVertex2i(boardred + 100, +315);
    glVertex2i(boardred + 100, +295);
```

```
   glVertex2i(boardred - 100, +295);

   glVertex2i(boardred - 100, +315);

   glEnd();


   RenderString(500.0f, -300.0f, GLUT_BITMAP_TIMES_ROMAN_24,1);

   RenderString(500.0f, 300.0f, GLUT_BITMAP_TIMES_ROMAN_24,0);


   glutSwapBuffers();

}

//Timer function

void timer(int)

{

   glutPostRedisplay();

   glutTimerFunc(1000 / 60, timer, 0);


   ball.xdir ? ball.xcenter += (speed + hits/3) : ball.xcenter -= (speed +
hits/3);

   ball.ydir ? ball.ycenter += (speed + hits/3) : ball.ycenter -= (speed +
hits/3);

}
```

```
//Main Function
int main(int argc, char **argv)
{

   glutInit(&argc, argv);

   glutInitDisplayMode(GLUT_DOUBLE | GLUT_RGBA);
   glutInitWindowSize(1500, 800);

   glutInitWindowPosition(0, 0);

   winid = glutCreateWindow("Glow Hockey");

   myInit();
   glutDisplayFunc(display);
   glutTimerFunc(10, timer, 0);
   glutSpecialFunc(keyboard);
   // glutKeyboardFunc(keyboard);
   glutMainLoop();
}
```

ABOUT :

A Mini Project about computer graphics using OpenGL. This is two player simple game about saving balls which come towards each player. In this game there is one palette provided each player by using that we have to save a ball which is coming towards each player reflecting continuously.

HOW TO USE :

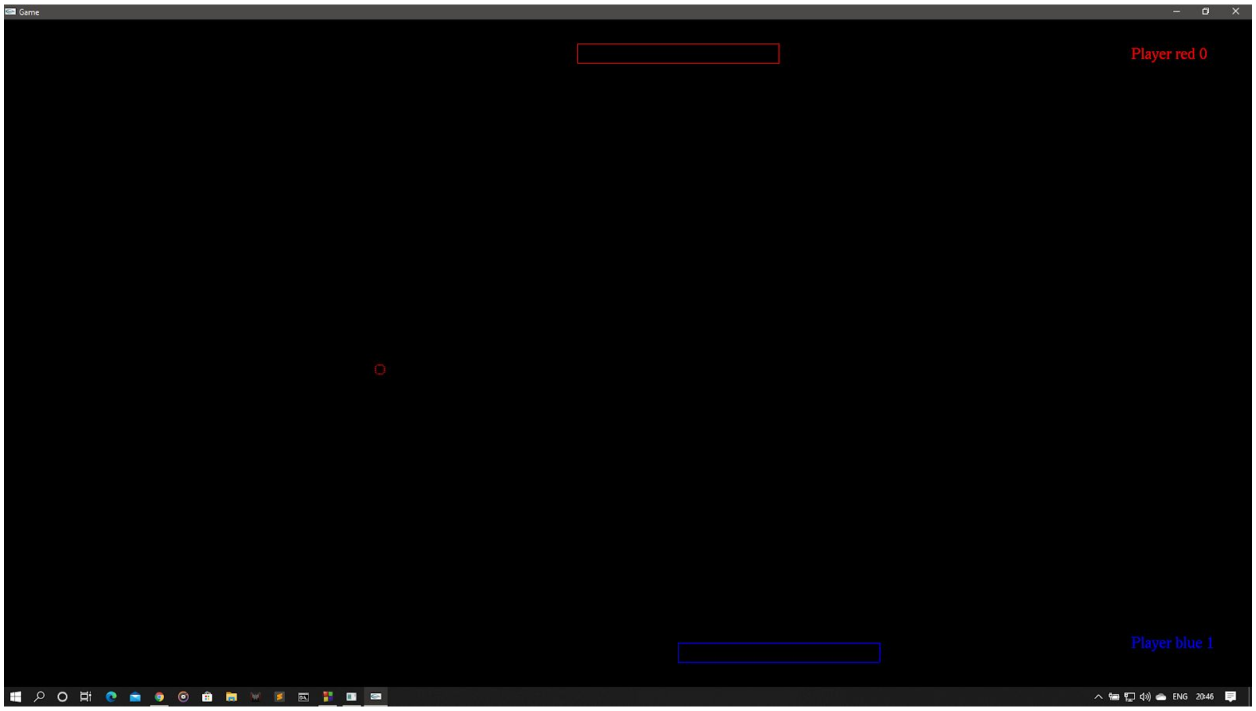
Player 1`s Keys To Move Palette :

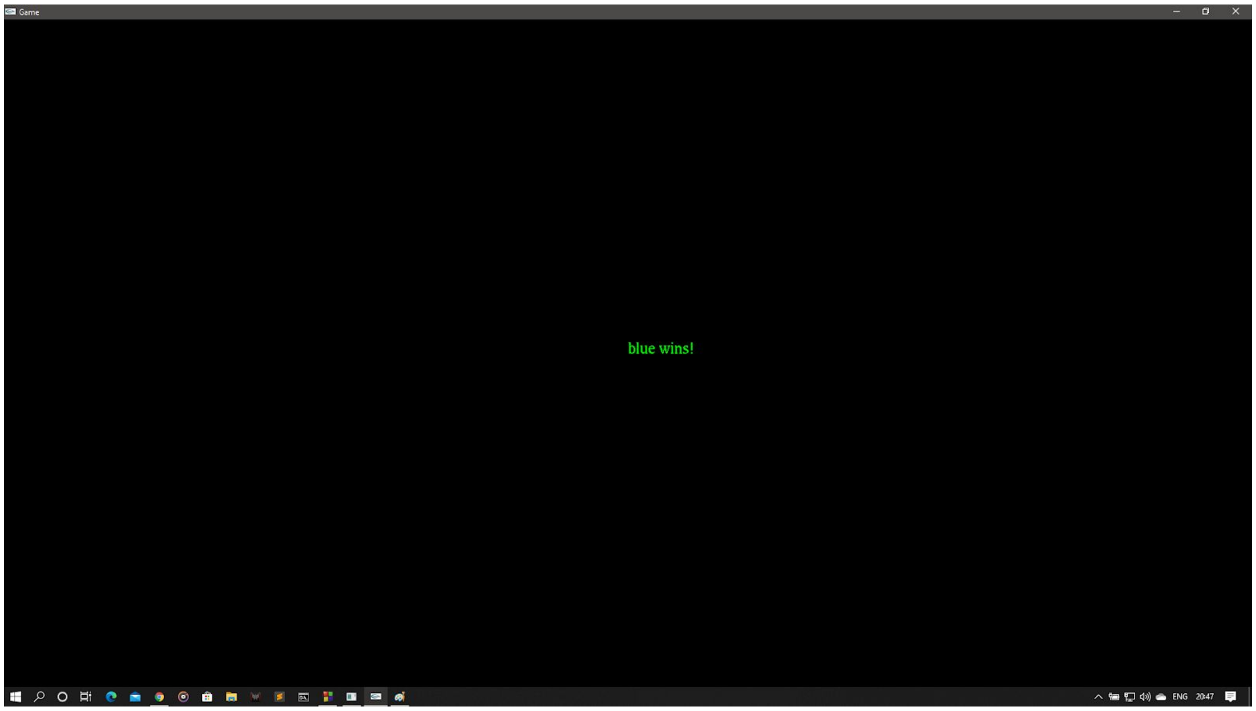
- **LEFT_ARROW**
- **RIGHT_ARROW**
- **UP_ARROW(to throw ball into right side)**
- **DOWN_ARROW(to throw ball into left side)**

Player 2`s Keys to Move Palette :

- **KEY_F1 (move left)**
- **KEY_F3 (move right)**
- **KEY_F2 (to throw ball into left side)**
- **KEY_F4 (to throw ball into right side)**

Dodge Ball


Game
Player red 0
Player blue 1
ENG
20:46


Game
blue wins!
ENG
20:47