Computer Graphics (UCS505)

Project on Flappy Bird

Submitted by:

Shreyas Sharma	101903713
Tanish Sharma	101903714
Raghav Gupta	101903716

Group No. 1
B.E. Third Year – COE

Submitted to:

Ms. Kudratdeep Aulakh

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

Patiala – 147001

Table of Contents

Introduction

This project explores the use of physical laws (gravitational law) for game development. The game in its exactness follows rules similar to that of the original game known as flappy bird.

The rules and regulation of the game are very basic and easy to understand. The game scenario puts the player in control of a hovering dot in the screen with moving obstacles of pipelines.

The goal of the game is to get through the opening in the obstacles without colliding into them. The game is implemented in C++. The case study discusses the use of simpler motion and modification to build the abstraction.

In addition, this report will also discuss the strategies used to stimulate the crowd pattern in the leisure environment. The game is in single player mode.

The user can interact with the game using keyboard. The position of the obstacles will also be discussed. The game is a 2D game with the purpose to keep the dot moving forward.

For the game to feel like Flappy Bird, the focus is given on the gameplay. The two essential elements that the game needs to get right are the “bird” and the “pipes”. Specifically, the bird has to move like a Flappy Bird, and our pipes must generate and move exactly as their Green counterparts.

We made this project so that we could give our best to show what we have learned. The main objectives of this project are:

1. To play the famous game flappy bird in computer.
2. To make it user friendly.
3. To provide an easy interface.
4. To entertain people in their leisure time.

Computer Graphics Concepts Used

Vector graphics and Raster graphics

Vector graphics are applied in bird movement and Raster graphics are applied in intro screen. Vector graphics are computer graphics images that are defined in terms of points on a Cartesian plane, which are connected by lines and curves to form polygons and other shapes. Vector graphics have the unique advantage over raster graphics in that the points, lines, and curves may be scaled up or down to any resolution with no aliasing. The points determine the direction of the vector path; each path may have various properties including values for stroke color, shape, curve, thickness, and fill.

Mathematical Structures

Spaces, points, vectors; dusts, curves, surfaces, solids. **gl Vertex** glVertex commands are used within glBegin/glEnd pairs to specify point, line, and polygon vertices. The current color, normal, texture coordinates, and fog coordinate are associated with the vertex when glVertex is called. When only x and y are specified, z defaults to 0 and w defaults to 1. When x, y, and z are specified, w defaults to 1.

Simulation and animation

Simulation & Animation studies methods and algorithms for the computation used in producing images intended to create the perception of motion.

Rendering

Realism, physical modeling, ray tracing, radiosity, visible surface determination, transparency, translucency, reflection, refraction, shadows, shading, surface and texture mapping.

glutTimerFunc & glutPostRedisplay & glutBitmapCharacter

glutTimerFunc registers the timer callback func to be triggered in at least msecs milliseconds. **glutPostRedisplay** marks the normal plane of current window as needing to be redisplayed.

glutPostWindowRedisplay works the specified window as needing to be redisplayed **glutBitmapCharacter** - renders a bitmap character using OpenGL.

Graphical User Interface

The use of pictures, images, icons, pop-up menus, graphical objects helps in creating a user friendly environment where working is easy and pleasant, using computer graphics we can create such an atmosphere where everything can be automated and anyone can get the desired action performed in an easy fashion

Implicit surfaces are a versatile representation of closed manifolds for modeling, simulation and rendering. They are defined as the isocontour of a volumetric scalar function, which allows intuitive handling of complex topology.

User Defined Functions

Birdmovementmouse - to regulate the gameplay and movement of the bird using the mouse.

Birdmovementkey - to regulate the gameplay and movement of the bird using the keyboard.

Renderbitmapstring - to give the values of initial introduction screen

Drawscore - to keep updating the score as and when the game progresses.

Introscreen - to display the introduction screen.

Gameoverscreen - to display the score screen when the game is over.

Menu - to display menu.

Display - to display the introduction screen, the gameover screen and other functions.

Resetgame - resets the game to its initial state.

Drawpipe - draws the obstacles in the game, their specific axes and the position to where they are initialized.

ObstacleCreate - to check whether the bird has crossed the particular pipe or not and increase the score corresponding to it.

drawBird - to draw the bird using general OPENGL functionalities and initialise its state and starting axes.

Birdcollision - to fix the final state and return the collision when the bird comes in contact with the pipespawner 4.

Birdfly - to help the bird move longitudinally by moving its axes, position and keep a check of the obstacle in front to smoothen the bird's ride.

Birdfall - receive the value from birdcollision function when the bird comes in contact with the pipespawner, fix the final state and reset the game.

Birdphysics - to see to the movement of the bird, its flaps, its wings and the angle of those wings.

C++ Code

```
#include <stdio.h>
#include <stdlib.h>
#include <GL/glut.h>

// Pipe
int obstacle[3] = {300, 500, 700};
int move_obstacle[3] = {0, 100, 0};
int collision[3][4][2] = {{{0, 0}, {0, 0}, {0, 0}, {0, 0}}, {{0, 0}, {0, 0}, {0, 0}, {0, 0}}, {{0, 0}, {0, 0}, {0, 0}, {0, 0}}};

// Bird
int wingRotateAngle = 0;
int position = 0;
int dist_moved = 0;
int out_of_bound[2][2] = {{-180, 10}, {-180, -10}};
// Game logic

int intro = 0;
int gameOver = 0;
int key_pressed = 0;
int score = 0;
int highscore = 0;
int reset = 0;
void resetGame()
```

```
{
    if (reset == 1)
    {
        gameOver = 0;
        reset = 0;
        score = 0;
        position = 0;
        dist_moved = 0;
        // Reseting Pipe Position
        obstacle[0] = 300;
        obstacle[1] = 500;
        obstacle[2] = 700;
        move_obstacle[0] = 0;

        move_obstacle[1] = 100;
        move_obstacle[2] = 0;
        glutPostRedisplay();
    }
}

void drawPipe(int index, int xaxis, int yaxis)
{
    // Collision Box
    collision[index][0][0] = xaxis;      // starting x axis
    collision[index][0][1] = yaxis + 50; //  top starting y axis
    collision[index][1][0] = xaxis + 50; //
```

```
collision[index][1][1] = yaxis + 50;
collision[index][2][0] = xaxis + 50;
collision[index][2][1] = yaxis – 50;
collision[index][3][0] = xaxis;
collision[index][3][1] = yaxis – 50;
// Top pipe
glPushMatrix();
glColor3f(0, 1, 0);
glBegin(GL_QUADS);
glVertex2i(xaxis, 800);
glVertex2i(xaxis + 50, 800);
glVertex2i(xaxis + 50, 50 + yaxis);
glVertex2i(xaxis, 50 + yaxis);
// Pipe head
glColor3f(0.137255, 0.556863, 0.137255);
glVertex2i(xaxis – 3, 70 + yaxis);
glVertex2i(xaxis + 50 + 3, 70 + yaxis);
glVertex2i(xaxis + 50 + 3, 50 + yaxis);
glVertex2i(xaxis – 3, 50 + yaxis);
glEnd();
// Bottom pipe
glColor3f(0, 1, 0);
glBegin(GL_QUADS);
glVertex2i(xaxis, –800);
glVertex2i(xaxis + 50, –800);
glVertex2i(xaxis + 50, –50 + yaxis);
```

```
    glVertex2i(xaxis, -50 + yaxis);
    // Pipe Head
    glColor3f(0.137255, 0.556863, 0.137255);

    glVertex2i(xaxis - 3, -70 + yaxis);
    glVertex2i(xaxis + 50 + 3, -70 + yaxis);
    glVertex2i(xaxis + 50 + 3, -50 + yaxis);
    glVertex2i(xaxis - 3, -50 + yaxis);
    glEnd();
    glPopMatrix();
}

void obstacle_create(int unused)
{
    if (obstacle[0] == -300 && gameOver != 1)
    {
        obstacle[0] = 300;
        move_obstacle[0] = (rand() % 20) * 10;
        score++;
    }

    else
        obstacle[0] -= 1;

    if (obstacle[1] == -300 && gameOver != 1)
    {
```

```
        obstacle[1] = 300;
        move_obstacle[1] = (rand() % 20) * 10;
        score++;
    }
    else
        obstacle[1] -= 1;

    if (obstacle[2] == -300 && gameOver != 1)
    {
        obstacle[2] = 300;
        move_obstacle[2] = (rand() % 20) * 10;
        score++;
    }
    else
        obstacle[2] -= 1;
    glutPostRedisplay();
    glutTimerFunc(10, obstacle_create, 0);
}

void drawBird()
{
    glPushMatrix();
    glTranslatef(0, position, 0);
    // Bird Y Axis
    out_of_bound[0][1] = 10 + position;
    out_of_bound[1][1] = -10 + position;
```

```
    glColor3f(1, 1, 0);
    glBegin(GL_QUADS);
    // Body
    glVertex2i(-200, 10);
    glVertex2i(-200 + 20, 10);
    glVertex2i(-200 + 20, -10);
    glVertex2i(-200, -10);
    glEnd();
    glColor3f(1, 0, 0);
    glBegin(GL_TRIANGLES);
    // Beak
    glVertex2i(-200 + 20, 3);
    glVertex2i(-200 + 25, 0);
    glVertex2i(-200 + 20, -3);
    glEnd();
    glRotatef(wingRotateAngle, 1, 0, 0);
    glBegin(GL_TRIANGLES);
    // Wing
    glVertex2i(-200 + 10, 5);
    glVertex2i(-200 + 5, -5);
    glVertex2i(-200 + 15, -5);
    glEnd();
    glPopMatrix();
}
```

```
int birdCollision()
{
    int i;

    for (i = 0; i < 3; i++)
    {
        if (collision[i][0][0] <= -180 && collision[i][1][0] >= -200)
        {
            if ((out_of_bound[0][1] >= collision[i][0][1] &&
                 out_of_bound[0][0] == -180) ||
                (out_of_bound[1][1] <= collision[i][2][1] &&
                 out_of_bound[1][0] == -180))
                return 1;
            else if ((out_of_bound[0][1] >= collision[i][0][1] &&
                      out_of_bound[0][0] == -200) ||
                     (out_of_bound[1][1] <= collision[i][2][1] &&
                      out_of_bound[1][0] == -200))
                return 1;
        }
    }
    return 0;
}

void birdFly(int unused)
{
    if (birdCollision() == 1)
```

```
    {
        gameOver = 1;
        reset = 1;
    }

    if (dist_moved != 0 && position != 300)
    {
        position++;
        dist_moved--;
    }

    else
    {
        key_pressed = 0;
        return;
    }

    glutPostRedisplay();
    glutTimerFunc(10, birdFly, unused);
}
void birdFall()
{
    if (birdCollision() == 1)
    {
        gameOver = 1;
        reset = 1;
```

```
    }

    if (key_pressed == 1)
        return;

    if (position != -300 && key_pressed == 0)
        position--;

    if (position == -300 || position == 300)
    {
        gameOver = 1;
        reset = 1;
    }

    glutPostRedisplay();
}
void birdPhysics(int unused)
{
    if (wingRotateAngle == 180)
        wingRotateAngle = 0;
    else
        wingRotateAngle = 180;
    birdFall();

    glutTimerFunc(10, birdPhysics, 0);
}
```

```
void birdMovementMouse(int btn, int state, int x, int y)
{
    if (btn == GLUT_LEFT_BUTTON && state == GLUT_DOWN)
    {
        dist_moved = 30;
        key_pressed = 1;
        birdFly(0);
    }
}

void birdMovementKey(unsigned char btn, int x, int y)
{
    if (btn == ' ')
    {
        dist_moved = 30;
        key_pressed = 1;
        birdFly(0);
    }

    if (btn == ' ' && intro == 0)
    {
        intro = 1;
        reset = 1;
        resetGame();
    }
```

```
    if (btn == ' ' && gameOver == 1)
        resetGame();
}

void renderBitmapString(float x, float y, void *font, const char *string)
{
    const char *c;
    glRasterPos2f(x, y);
    for (c = string; *c != '\0'; c++)
        glutBitmapCharacter(font, *c);
}

void drawScore()
{
    char buffer[15] = {'\0'};
    sprintf_s(buffer, "Score : %d", score);
    glColor3f(0, 0, 0);

    renderBitmapString(-290, 270, GLUT_BITMAP_TIMES_ROMAN_24,
buffer);
}
void introScreen()
{
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0, 0, 0, 1);
```

```
    glColor3f(0, 0, 1);
    renderBitmapString(-170, 265, GLUT_BITMAP_TIMES_ROMAN_24,
                       "Thapar Institute of Engineering and Technology");
    glColor3f(1, 0, 0);
    renderBitmapString(-80, 180, GLUT_BITMAP_TIMES_ROMAN_24,
                       "FLAPPY BIRD");
    glColor3f(1, 0.5, 0);
    renderBitmapString(-290, 10, GLUT_BITMAP_TIMES_ROMAN_24,
                       "By:");
    glColor3f(0.5, 0, 0.5);
    renderBitmapString(-260, -40, GLUT_BITMAP_TIMES_ROMAN_24,
                       "Shreyas Sharma (101903713)");
    renderBitmapString(-260, -70, GLUT_BITMAP_TIMES_ROMAN_24,
                       "Tanish Sharma (101903714)");
    renderBitmapString(-260, -100,
                       GLUT_BITMAP_TIMES_ROMAN_24, "Raghav Gupta
(101903716)");
    glColor3f(1, 0.5, 0);
    renderBitmapString(-80, -275, GLUT_BITMAP_TIMES_ROMAN_24,
                       "Press Space to Start");
    glFlush();
}

void gameOverScreen()
{
    char buffer[15] = {'\0'};
```

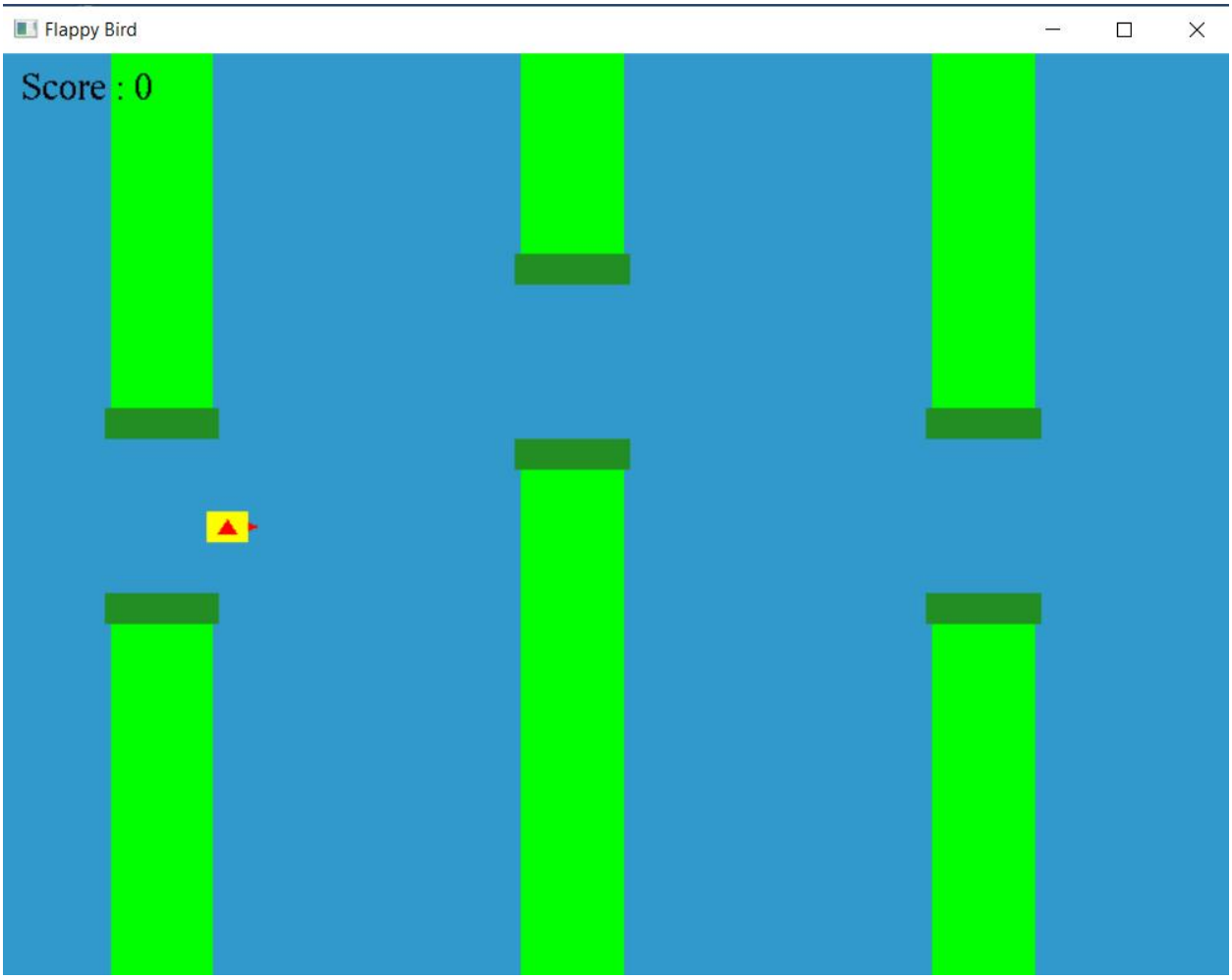
```
    glClear(GL_COLOR_BUFFER_BIT);
    glClearColor(0, 0, 0, 1);
    glColor3f(1, 0, 0);
    renderBitmapString(-40, 90, GLUT_BITMAP_TIMES_ROMAN_24,
"Game Over");

    glColor3f(1, 0.5, 0);

    if (highscore < score)
        highscore = score;

    sprintf_s(buffer, "Score : %d", score);
    renderBitmapString(-30, -30, GLUT_BITMAP_TIMES_ROMAN_24,
buffer);
    sprintf_s(buffer, "HighScore : %d", highscore);
    renderBitmapString(-45, -90, GLUT_BITMAP_TIMES_ROMAN_24,
buffer);
    glColor3f(1, 0.1, 1);
    renderBitmapString(-80, -275, GLUT_BITMAP_TIMES_ROMAN_24,
"Press Space to Restart");
    glFlush();
}

void menu(int option)
{
    if (option == 1)
```

```
        resetGame();
    if (option == 2)
        exit(0);
}

void display()
{
    if (intro == 0)
        introScreen();
    else if (gameOver == 1)
        gameOverScreen();
    else
    {
        glClear(GL_COLOR_BUFFER_BIT);
        glClearColor(0.196078, 0.6, 0.8, 1);
        glLoadIdentity();
        drawPipe(0, obstacle[0], move_obstacle[0]);
        drawPipe(1, obstacle[1], move_obstacle[1]);
        drawPipe(2, obstacle[2], move_obstacle[2]);
        drawScore();
        drawBird();
        glFlush();
    }
}

void init()
```

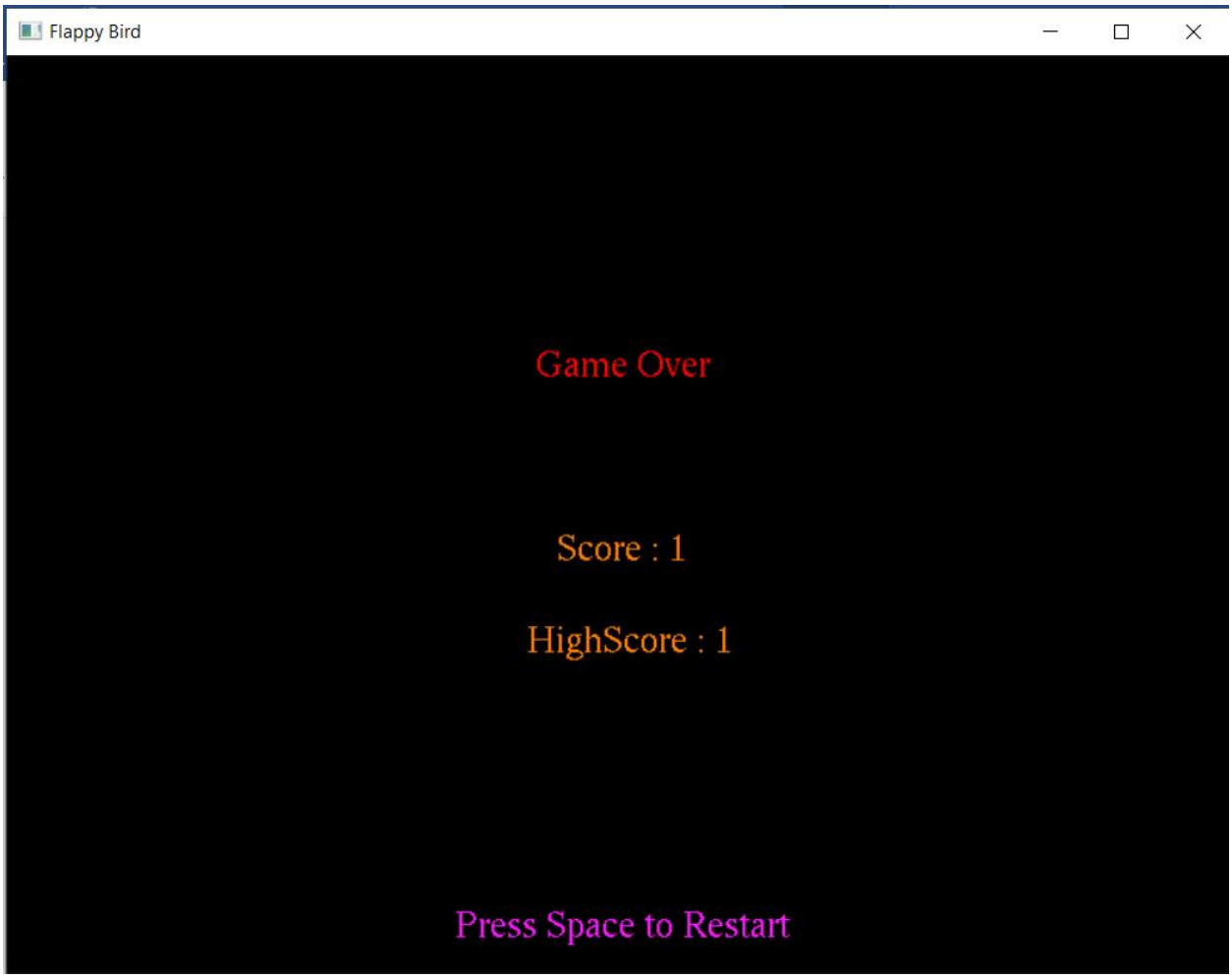
```
{
    glMatrixMode(GL_PROJECTION);
    gluOrtho2D(-300, 300, -300, 300);
    glMatrixMode(GL_MODELVIEW);
}

int main(int argc, char **argv)
{
    glutInit(&argc, argv);
    glutInitDisplayMode(GLUT_RGB);
    glutInitWindowSize(800, 600);
    glutInitWindowPosition(40, 40);
    glutCreateWindow("Flappy Bird");
    glutDisplayFunc(display);
    glutMouseFunc(birdMovementMouse);
    glutKeyboardFunc(birdMovementKey);
    glutCreateMenu(menu);
    glutAddMenuEntry("Restart", 1);
    glutAddMenuEntry("Exit", 2);
    glutAttachMenu(GLUT_RIGHT_BUTTON);
    init();
    obstacle_create(0);
    birdPhysics(0);
    glutMainLoop();
}
```

Output/Screenshots

Welcome Screen



Game Screen


Flappy Bird
Game Over
Score : 1
HighScore : 1
Press Space to Restart

End Screen