Department of Computer Science & Information Technology

Object-Oriented Programming (OOP)

“Flappy Bird Game”

Project Report

Course Code: CT-260

SEC: B

**GROUP MEMBERS:**

* **EMAN SHAHID(CT-22058)**
* **SYED ALI MURTAZA RIZVI(CT-22093)**
* **TAQI HAIDER(CT-22092)**

**PROBLEM STATEMENT:**

This Code is a C++ Program that creates a simple game called “FLAPPY BIRD” using the “olcConsoleGameEngine” library.

**PROJECT DESCRIPTION:**

The program defines a class called “FLAPPY BIRD” that inherits from “olcConsoleGameEngine”. The Class contains several members variables that represent different aspects of the game, such as the bird’s position, the game’s score, and the obstacles that the bird must avoid.

**DESCRIPTION OF FUNCTIONALITIES**

We have used 2 user defined functions which are as under:

**1.OnUserCreate:**

The "OnUserCreate" function is a virtual function that is called once when the game starts. It initializes the game's variables and sets the initial position of the bird.

2.OnUserUpdate:

The "OnUserUpdate" function is another virtual function that is called repeatedly during the game loop. It contains the main game logic, including updating the bird's position and checking for collisions with obstacles.

INDIVIDUAL CONTRIBUTIONS:

1. **Syed Ali Murtaza Rizvi(CT-22093):**

Provided the main idea of game. Worked with the components of base class “olcConsoleGameEngine” and its main functions which are inherited by derived class.

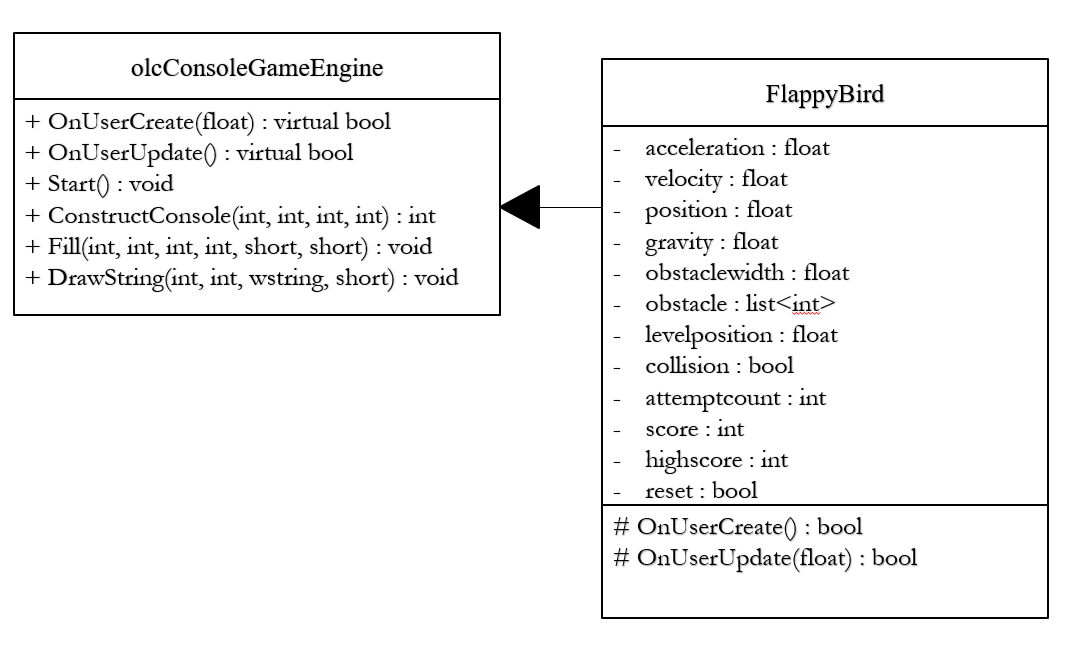
1. **Taqi Haider(CT-22092):**

Basically worked with the logic of game components such as bird’s movement and obstacles logic and also worked on project report.

1. **Eman Shahid(CT-22058):**

Implemented and ensured the use of pillars of Objected Oriented Programming and made code readable and less complex by removing unnecessary conditions and variables. Also worked on Project report.

**UML**

****

**SOURCE:**

#include <iostream>

#include <string>

#include<list>

using namespace std;

#include "olcConsoleGameEngine.h"

class FlappyBird : public olcConsoleGameEngine {

public:

FlappyBird() {}

private:

//bird components

float acceleration = 0.0f;

float velocity = 0.0f;

float position = 0.0f;

float gravity = 100.0f;

//game logic components

float obstaclewidth;

list<int> obstacle;

float levelposition = 0.0f;

bool collision = false;

int attemptcount = 0;

int score = 0;

int highscore = 0;

bool reset = false;

protected:

bool OnUserCreate() {

obstacle = { 0,0,0,0 };

obstaclewidth = (float)ScreenWidth() / (float)obstacle.size() - 1;

reset = true;

return true;

}

bool OnUserUpdate(float fElapsedTime) {

//Game Logic

if (reset) {

score = 0;

obstacle = { 0,0,0,0 };

reset = false;

collision = false;

acceleration = 0.0f;

position = ScreenHeight() / 2.0f;

velocity = 0.0f;

attemptcount++;

}

if (collision) {

DrawString(ScreenWidth() / 2.5, ScreenHeight() / 2.5, L"TRY AGAIN");

if (m\_keys[VK\_SPACE].bReleased) {

reset = true;

Fill(0, 0, ScreenWidth(), ScreenHeight(), L' ');

}

}

else {

if (m\_keys[VK\_SPACE].bPressed) {

acceleration = 0.0f;

velocity = -gravity / 5.0f;//minus sign is to move up beacuse in graph subtracting in y axis will make line move up and since this console is in x y plane we implement this graph logic

score++;

if (score > highscore) {

highscore = score;

}

}

else {

acceleration += gravity \* fElapsedTime;//from law of motion :a=gt

if (acceleration > gravity) {

acceleration = gravity;

}

velocity += acceleration \* fElapsedTime;//from law of motion :v=at;

}

position += velocity \* fElapsedTime;//from law of motion :p=vt

levelposition += 14.0f \* fElapsedTime;//this determines how fast game moves

if (levelposition > obstaclewidth) {//obstacle width if screen width is 80 then it 26.00 which means after every two section it will do this

levelposition -= obstaclewidth;//this will move back to it starting position

obstacle.pop\_front();

int i = rand() % (ScreenHeight() - 35) + 20;//we can change this to change gap of obstacle

if (i <= 10) { i = 0; }

obstacle.push\_back(i);

}

Fill(0, 0, ScreenWidth(), ScreenHeight(), L' ');//this is to clear screen per frame or on buffer

int nobstacles = 0;

for (list<int>::iterator it = obstacle.begin(); it != obstacle.end(); ++it) {

int k = \*it;

if (k != 0) {

Fill(nobstacles \* obstaclewidth + 10 - levelposition, ScreenHeight() - k, nobstacles \* obstaclewidth + 15 - levelposition, ScreenHeight(), PIXEL\_SOLID, FG\_GREEN);

Fill(nobstacles \* obstaclewidth + 10 - levelposition, 0, nobstacles \* obstaclewidth + 15 - levelposition, ScreenHeight() - k - 15, PIXEL\_SOLID, FG\_GREEN);

}

nobstacles++;

}

int bird\_on\_X = (int)(ScreenWidth() / 3.0f);

// Collision Detection

collision = position < 2 || position > ScreenHeight() - 2 ||

m\_bufScreen[(int)(position + 0) \* ScreenWidth() + bird\_on\_X].Char.UnicodeChar != L' ' ||

m\_bufScreen[(int)(position + 1) \* ScreenWidth() + bird\_on\_X].Char.UnicodeChar != L' ' ||

m\_bufScreen[(int)(position + 0) \* ScreenWidth() + bird\_on\_X + 6].Char.UnicodeChar != L' ' ||//this is to check the face if only face is collided when moving down

m\_bufScreen[(int)(position + 1) \* ScreenWidth() + bird\_on\_X + 6].Char.UnicodeChar != L' ';//this is to check the face if only face is collided when moving up

//bird

if (velocity > 0) {

DrawString(bird\_on\_X, position + 0, L"////");

DrawString(bird\_on\_X, position + 1, L"<//////O>");

}

else {

DrawString(bird\_on\_X, position + 0, L"<//////O>");

DrawString(bird\_on\_X, position + 1, L"////");

}

DrawString(1, 1, L"Attempt: " + to\_wstring(attemptcount) + L" Score: " + to\_wstring(score) + L" High Score: " + to\_wstring(highscore));

}

return true;

}

};

int main() {

FlappyBird game;

game.ConstructConsole(160, 48, 16, 16);

game.Start();

}

**OUTPUT**

