**Project: Flappy Bird with Framework**



Session: 2021 – 2025

**Submitted by:**

Mukarram Ali 2021-CS-58

**Supervised by:**

Dr Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Content:**

* Short Description of Game**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**3
* Game Character Description\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
* Rules and Interaction\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
* Goal of the Game\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_3
* Features of Framework\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4
* Class Diagram\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_4
* Sequence Diagram\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_5
* Example Usage of Framework\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_6
* Complete Code\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7

**Description of Game:**

* Flappy Bird is an Arcade-Style game in which player controls the bird Faby, which moves persistently to the right.
* The Bird is passed through pair of pipes that have equally sized gaps placed at random heights. Bird automatically descends when the user press the down key and ascends when the user press up key.
* The bird will have to save himself from the Pipes, Enemies, Ground, Fires.

**Game Characters Description:**

* Bird (the player) he can move up and down and the plateform will move toward him. He also has the progress bar that will decrease when he collides with the fire.
* Pipes will move towards the Bird(Player) and the Bird live will decrease when he collides with the pipes.
* Enemies will move towards the player they also has the progress bar and they can also fire after a certain time. Their movement will be zigzag ,horizontal ,vertical.
* Coins in game will increase the score of the Bird.
* Heart in game will increase the the lives of Bird.
* Power Ups will increase the value of progress bar of bird.

**Rules and Interactions:**

* Bird (player) will reduce the life if he collides with the Pipes, Enemy, Ground.
* Bird (player) progress bar value will decrease when he collides with the fires.
* Bird (player) progress bar value will increase when he collides with the power ups.
* Bird (player) life will increase when he collides with heart.
* Bird (player) score will increase when he collides with the coins.
* Game will restart when all the lives are lost or the time is over.
* Next Level will open when Bird (player) collects the given number of coins in fixed time.
* Bird (player) will fire when he press the space bar.
* Enemy progress bar value will also decrease when he collides with the player fire and he will die when its progress bar value reaches zero.
* Metaphors will fall randomly from the top and act as an enemy.
* Enemies and pipes will appear randomly.

**Goal of Game:**

The goal of the game is to save yourself from the enemies, pipes, ground, fires and collect the coins to increase the score. You will win the game when you achieve the required score.

**Features of Framework:**

My Framework provides the three features:

1. Movement Framework

* By using this Framework, we can create multiple movements.
* Following are the movements: Horizontal, Vertical, KeyBoard Movement, Diagonal and ZigZag.

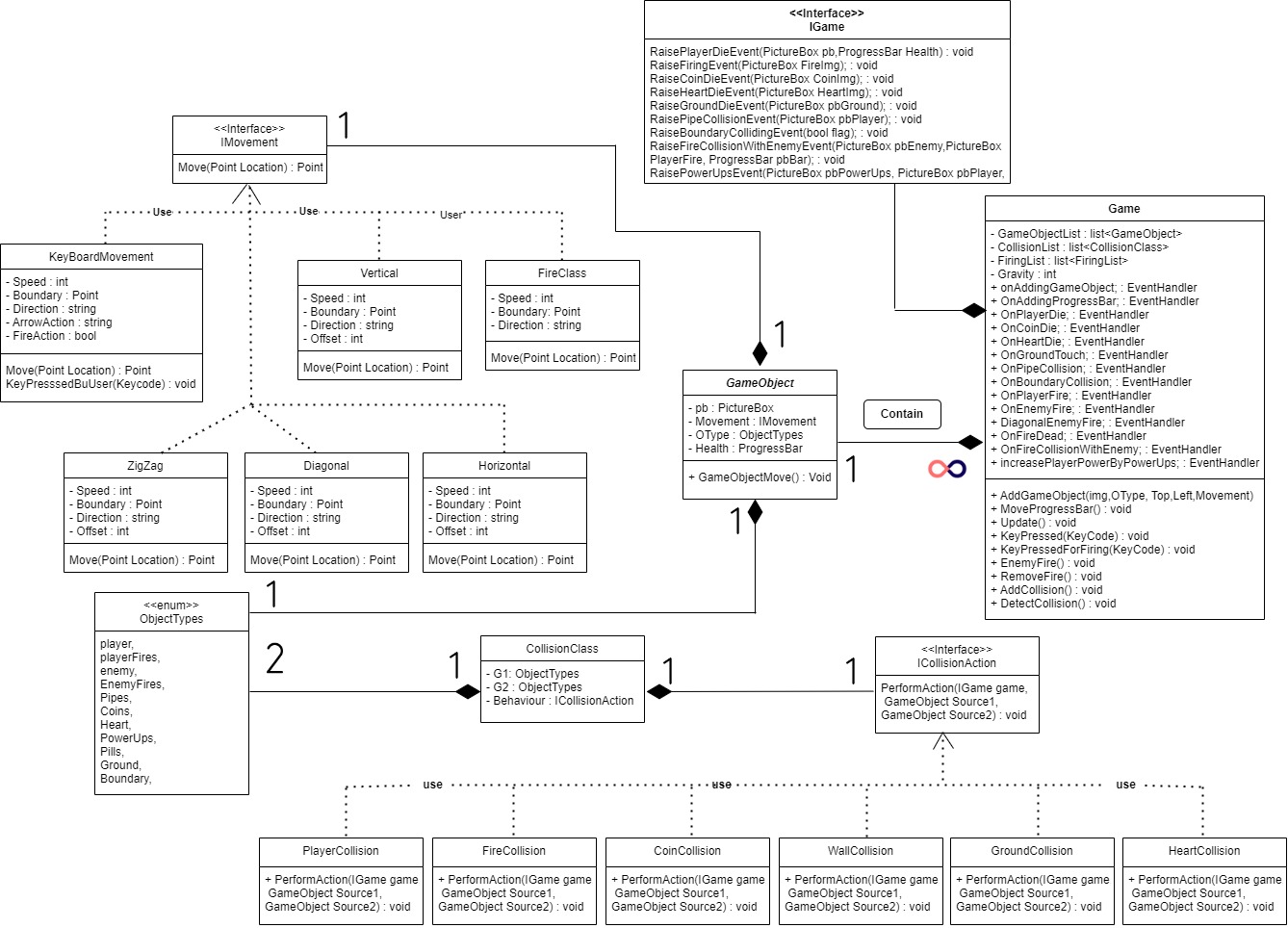
1. Collision Framework

* By using this Framework, we can handle multiple collisions.
* Following are the Collisions that we can handle: Boundary Collision, Coin Collision, Enemy Collision, Fire Collision, Ground Collision, Heart Collision, Player Collision, Power Ups Collision, Wall collision.

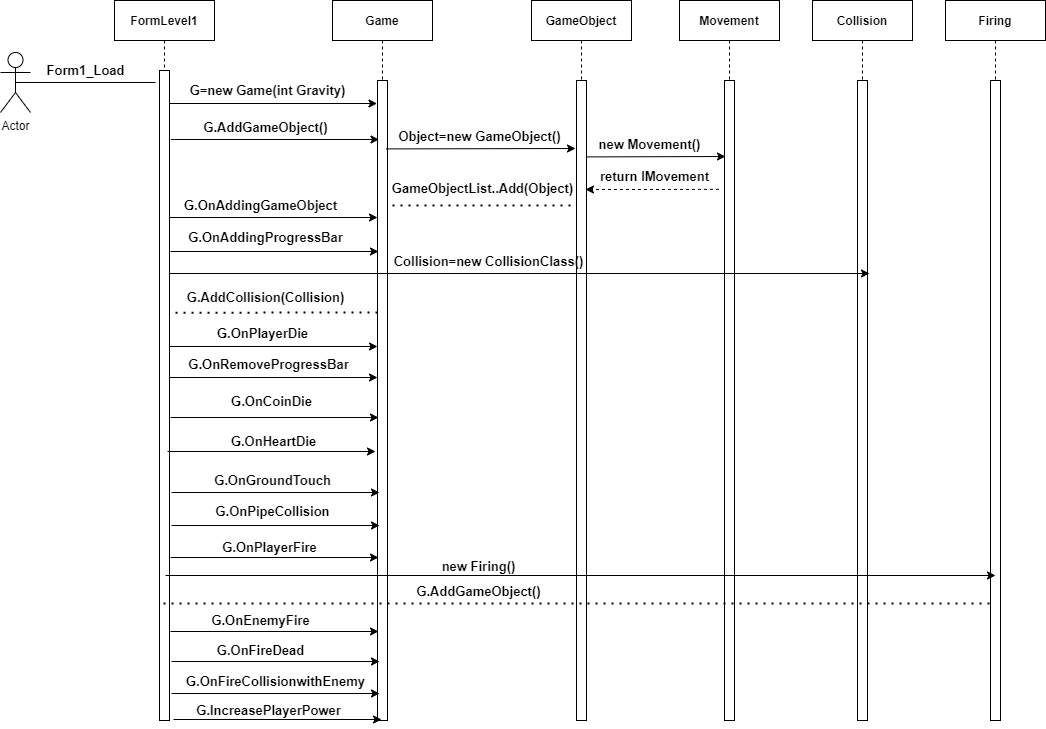
1. Firing Framework

* By using this Framework we can create the Player and Enemy Fires.

**Class Diagram:**

****

**Sequence Diagram:**



**Example Usage of Framework:**

* If a person wants to create a player that can be controlled by keys then he he has to create an object of the Game class and after that by using the AddGameObject Function in Game class and passing all the parameter and giving its movement class, the player can be created.The object will be created in Form class.Here is the sample example:

**Example:**

G.AddGameobject(Image img, ObectTypes OType.player, int Top, int Left, new KeyboardMovement(int PlayerSpeed, Point Boundary,string Direction));

* If a person wants to create a Enemy that can be controlled randomly then by using the Game object and passing all the parameter and giving its movement class, the Enemy can be created. The object will be created in Form class.Here is the sample example:

**Example:**

G.AddGameobject(Image img, ObectTypes OType.Enemy, int Top, int Left, new Movement(int EnemySpeed, Point Boundary,string Direction));

* If you want when a person press the space bar your player fire. For this purpose he has to handle the event when the user presses the space bar.In that event function we will have to create an object by and then add into the gameobjectlist by using the addGameObject function.The object will be created in the Form class. Here is the sample example:

**Example:**

G.AddGameobject(Image img, ObectTypes OType.PlayerFire, int Top, int Left, new FireClass(int EnemySpeed, Point Boundary,string Direction));

If you want that the enemy also fires after certain time then the procedure is same as the firing of the Player. The object will be created in the Form class. Here is the sample example:

**Example:**

G.AddGameobject(Image img, ObectTypes OType.EnemyFire, int Top, int Left, new FireClass(int EnemySpeed, Point Boundary,string Direction));

* If you want to detect the collision between the any two kind of objects like Player and Pipes.If the object like Pipes is on design time then we will have to pass it in the GameObjectList from the form by creating an object of GameObject.And after that we will have to create an object of CollisionClass and we will pass the parameter in it and then we will add it in Collision List by using the AddCollision function.Here is the sample example:

**Example:**

CollisionClass PlayerCollision=new CollisionClass(ObjectTypes.player, ObjectTypes.enemy, new PlayerCollision());

G.AddCollision(PlayerCollision);

**Complete Code:**

**Form1:**

public partial class Flappy\_Bird\_Game : Form

{

Game g;

Random rand;

int EnemyLeft, EnemyTop, EnemySpeed, count, Score, PipeSpeed,CoinSpeed,Left,top,FireSpeed, count2=0;

Image EnemyHorizontalImg, EnemyZigZagImg;

PictureBox pbPlayer;

PictureBox EnemyLeftRight,EnemyZigZagBlue,EnemyZigZagYellow;

PictureBox PipeBottom1, PipeBottom2, PipeTop1, PipeTop2;

int TotalLives = 3;

List<GameObject> CoinsList = new List<GameObject>();

Point EnemyZigZagBluePoint = new Point();

Point EnemyZigZagYellowPoint = new Point();

Point EnemyLeftRightPoint = new Point();

Point Pipe1 = new Point();

Point Pipe2 = new Point();

Point Pipe3 = new Point();

Point Pipe4 = new Point();

}

public Flappy\_Bird\_Game()

{

InitializeComponent();

}

private void Form2\_Load(object sender, EventArgs e)

{

RestartAgain();

}

private void RestartAgain()

{

rand = new Random();

EnemySpeed = 5; count = 0; PipeSpeed = 5 ; CoinSpeed = 5; FireSpeed=6;

g = new Game(8);

g.onAddingGameObject += new EventHandler(AddIntoControls);

g.OnAddingProgressBar += new EventHandler(AddProgressBar);

g.OnPlayerDie += new EventHandler(RemovePlayer);

g.OnRemoveProgressBar += new EventHandler(RemoveProgressBar);

g.OnAddingFires += new EventHandler(AddFiresInControl);

g.OnCoinDie += new EventHandler(RemoveCoinFromControls);

g.OnHeartDie += new EventHandler(RemoveHeartFromControl);

g.OnGroundTouch += new EventHandler(OnGoundTouchEvent);

g.OnPipeCollision += new EventHandler(PipeCollisionEvent);

g.OnBoundaryCollision += new EventHandler(BoundaryCollisionEvent);

g.OnPlayerFire += new EventHandler(OnPlayerFire);

g.OnEnemyFire += new EventHandler(OnEnemyFire);

g.DiagonalEnemyFire += new EventHandler(DiagonalEnemyFire);

g.OnFireDead += new EventHandler(RemoveFire);

g.OnFireCollisionWithEnemy += new EventHandler(ReducePowerOfProgressBar);

g.IncreasePlayerPowerByPowerUps += new EventHandler(IncreasePowerPlayerEvent);

//Fire(Playerimg, ObjectTypes.player, new HorizontalFire(FireSpeed, Direction));

Point Boundary = new Point(this.Width, this.Height);

Image PlayerImg = Game1.Properties.Resources.FlappyDown;

pbPlayer= g.AddGameObject(PlayerImg,ObjectTypes.player, 20, 20, new KeyBoardMovement(5, Boundary, "Down"));

EnemyHorizontalImg = Game1.Properties.Resources.flyMan\_still\_stand;

EnemyLeft = rand.Next(800,950);

EnemyTop = rand.Next(20, EnemyHorizontalImg.Height + 20);

EnemyLeftRightPoint.X = EnemyLeft;

EnemyLeftRightPoint.Y = EnemyTop;

EnemyLeftRight= g.AddGameObject(EnemyHorizontalImg,ObjectTypes.enemy, EnemyTop, EnemyLeft, new Horizontal(EnemySpeed, Boundary, "Left"));

EnemyZigZagImg = Game1.Properties.Resources.narwhal\_\_1\_\_\_1\_;

EnemyLeft = rand.Next(960,1100);

EnemyTop = rand.Next(20, 150);

EnemyZigZagBluePoint.X = EnemyLeft; EnemyZigZagBluePoint.Y = EnemyTop;

EnemyZigZagBlue = g.AddGameObject(EnemyZigZagImg,ObjectTypes.enemy, EnemyTop, EnemyLeft, new ZigZag(EnemySpeed, Boundary, "Up"));

EnemyZigZagImg = Game1.Properties.Resources.wing;

EnemyLeft = rand.Next(1110,1250);

EnemyTop = rand.Next(150, 300);

EnemyZigZagYellowPoint.X = EnemyLeft; EnemyZigZagYellowPoint.Y = EnemyTop;

EnemyZigZagYellow = g.AddGameObject(EnemyZigZagImg,ObjectTypes.enemy, EnemyTop, EnemyLeft, new ZigZag(EnemySpeed, Boundary, "Up"));

GameObject Ground = new GameObject(pbGround, ObjectTypes.Ground, new Horizontal(0, Boundary, "Left"));

g.AddCoinInGameObjectList(Ground);

//---------------Collision With Pipes-------------------

Image pipeImgBottom1 = Game1.Properties.Resources.PipeResized2;

PipeBottom1= g.AddGameObject(pipeImgBottom1, ObjectTypes.Pipes, 278, 391, new Horizontal(PipeSpeed, Boundary, "Left"));

Image pipeImgBottom2 = Game1.Properties.Resources.PipeResized3;

PipeBottom2 = g.AddGameObject(pipeImgBottom2, ObjectTypes.Pipes, 226, 645, new Horizontal(PipeSpeed, Boundary, "Left"));

Image pipeImgTop1 = Game1.Properties.Resources.PipeTop1;

PipeTop1 = g.AddGameObject(pipeImgTop1, ObjectTypes.Pipes, 0, 419, new Horizontal(PipeSpeed, Boundary, "Left"));

Image PipeImgTop2 = Game1.Properties.Resources.PipeTop2;

PipeTop2 = g.AddGameObject(PipeImgTop2, ObjectTypes.Pipes, 0,694, new Horizontal(PipeSpeed, Boundary, "Left"));

//---------------End Collision With Pipes-------------------

//---------------Collision With Player and Enemies-------------------

CollisionClass PlayerCollision = new CollisionClass(ObjectTypes.player, ObjectTypes.enemy, new PlayerCollision());

//CollisionClass EnemyCollision = new CollisionClass(ObjectTypes.player, ObjectTypes.enemy, new EnemyCollision());

CollisionClass WallsCollision = new CollisionClass(ObjectTypes.player, ObjectTypes.Pipes, new WallCollision());

foreach (Control c in this.Controls)

{

if ((string)c.Tag == "Coin")

{

GameObject coin = new GameObject((PictureBox)c, ObjectTypes.Coins, new Horizontal(CoinSpeed, Boundary, "Left"));

g.AddCoinInGameObjectList(coin);

}

}

CollisionClass CoinsCollision = new CollisionClass(ObjectTypes.player, ObjectTypes.Coins, new CoinCollision());

foreach(Control c in this.Controls)

{

if((string)c.Tag=="Heart")

{

GameObject Heart = new GameObject((PictureBox)c,ObjectTypes.Heart,new Horizontal(CoinSpeed,Boundary,"Left"));

g.AddCoinInGameObjectList(Heart);

}

}

CollisionClass PowerUpsCollision = new CollisionClass(ObjectTypes.player,ObjectTypes.PowerUps,new PowerUpsCollision());

foreach(Control c in this.Controls)

{

if((string)c.Tag=="PowerUps")

{

GameObject PowerUps = new GameObject((PictureBox)c,ObjectTypes.PowerUps,new Horizontal(CoinSpeed,Boundary,"Left"));

g.AddCoinInGameObjectList(PowerUps);

}

}

CollisionClass HeartCollision = new CollisionClass(ObjectTypes.player,ObjectTypes.Heart,new HeartCollision());

CollisionClass GroundCollision = new CollisionClass(ObjectTypes.player,ObjectTypes.Ground,new GroundCollision());

CollisionClass BoundayCollision = new CollisionClass(ObjectTypes.player, ObjectTypes.Boundary, new BoundaryCollision(Boundary));

CollisionClass FireCollisionWithEnemy = new CollisionClass(ObjectTypes.enemy,ObjectTypes.playerFires,new FireCollision());

CollisionClass PlayerCollisionWithEnemyFire = new CollisionClass(ObjectTypes.player,ObjectTypes.EnemyFires,new PlayerFireCollision());

g.AddCollision(PlayerCollision);

//g.AddCollision(EnemyCollision);

g.AddCollision(WallsCollision);

g.AddCollision(CoinsCollision);

g.AddCollision(HeartCollision);

g.AddCollision(GroundCollision);

g.AddCollision(BoundayCollision);

g.AddCollision(FireCollisionWithEnemy);

g.AddCollision(PlayerCollisionWithEnemyFire);

g.AddCollision(PowerUpsCollision);

//---------------End Collision With Player and Enemies------------------

private void IncreasePowerPlayerEvent(object sender, EventArgs e)

{

ProgressBar bar = (ProgressBar)sender;

if (bar.Value < 80)

{

bar.Value += 20;

}

}

private void DiagonalEnemyFire(object sender, EventArgs e)

{

PictureBox Enemy = (PictureBox)sender;

Image FireImg = Game1.Properties.Resources.tank\_explosion3;

Point Boundary = new Point(this.Width, this.Height);

int Top = Enemy.Location.Y;

int Left = Enemy.Location.X;

g.AddGameObject(FireImg, ObjectTypes.EnemyFires, Top, Left, new FireClass(FireSpeed, Boundary, "LeftDiagonal"));

}

private void ReducePowerOfProgressBar(object sender, EventArgs e)

{

ProgressBar p =(ProgressBar)sender;

p.Value -= 10;

}

private void RemoveFire(object sender, EventArgs e)

{

this.Controls.Remove((PictureBox)sender);

}

private void OnEnemyFire(object sender, EventArgs e)

{

PictureBox Enemy = (PictureBox)sender;

Image FireImg = Game1.Properties.Resources.laserBlue03\_\_1\_;

Point Boundary = new Point(this.Width, this.Height);

int Top = Enemy.Location.Y;

int Left = Enemy.Location.X;

g.AddGameObject(FireImg,ObjectTypes.EnemyFires,Top,Left,new FireClass(FireSpeed, Boundary,"Left"));

}

private void OnPlayerFire(object sender, EventArgs e)

{

PictureBox pbPlayer = (PictureBox)sender;

Image FireImg = Game1.Properties.Resources.laserRed07\_\_1\_;

Point Boundary = new Point(this.Width, this.Height);

int Top = pbPlayer.Location.Y;

int Left = pbPlayer.Location.X;

g.AddGameObject(FireImg,ObjectTypes.playerFires,Top,Left,new FireClass(FireSpeed, Boundary,"Right"));

}

private void BoundaryCollisionEvent(object sender, EventArgs e)

{

bool flag = (bool)sender;

if(flag==true)

{

TotalLives--;

Thread.Sleep(1000);

RestartGame();

}

}

private void PipeCollisionEvent(object sender, EventArgs e)

{

if(TotalLives>0)

{

TotalLives--;

Thread.Sleep(1000);

RestartGame();

}

}

private void OnGoundTouchEvent(object sender, EventArgs e)

{

if (TotalLives > 0)

{

TotalLives--;

Thread.Sleep(1000);

RestartGame();

}

}

private void RemoveHeartFromControl(object sender, EventArgs e)

{

TotalLives++;

this.Controls.Remove((PictureBox)sender);

}

private void RemoveCoinFromControls(object sender, EventArgs e)

{

Score++;

this.Controls.Remove((PictureBox)sender);

}

private void AddFiresInControl(object sender, EventArgs e)

{

this.Controls.Add((PictureBox)sender);

}

private void RemoveProgressBar(object sender, EventArgs e)

{

ProgressBar p = (ProgressBar)sender;

if (p.Value <= 0)

{

this.Controls.Remove((ProgressBar)sender);

}

}

private void AddProgressBar(object sender, EventArgs e)

{

this.Controls.Add((ProgressBar)sender);

}

private void RemovePlayer(object sender, EventArgs e)

{

if (TotalLives >= 0)

{

TotalLives--;

Thread.Sleep(1000);

RestartGame();

//this.Controls.Remove((PictureBox)sender);

}

}

private void AddIntoControls(object sender, EventArgs e)

{

this.Controls.Add((PictureBox)sender); }

private void ChangePlayerImages()

{

count2++;

if (count2 == 3)

{

pbPlayer.Image = Game1.Properties.Resources.FlappyDown;

}

if (count2 == 6)

{

pbPlayer.Image = Game1.Properties.Resources.FlappyBirdMid;

}

if (count2 == 9)

{

pbPlayer.Image = Game1.Properties.Resources.FlappyBirdUp;

count2 = 0;

}

}

private void ChangePicsEnemyYellow()

{

count++;

if (count == 5)

{ EnemyZigZagYellow.Image = Game1.Properties.Resources.wing\_\_1\_; }

if (count == 10)

{ EnemyZigZagYellow.Image = Game1.Properties.Resources.wingMan1\_\_1\_; }

if (count == 15)

{ EnemyZigZagYellow.Image = Game1.Properties.Resources.wingMan2\_\_1\_; }

if (count == 20)

{ EnemyZigZagYellow.Image = Game1.Properties.Resources.wingMan4\_\_1\_; }

if (count == 25)

{ EnemyZigZagYellow.Image = Game1.Properties.Resources.wingMan5\_\_1\_; count = 0; }

}

private void LivesRemainimg\_Click(object sender, EventArgs e)

{

}

private void ChangePicsEnemyHorizontal()

{

count1++;

if(count1==5)

{

EnemyLeftRight.Image = Game1.Properties.Resources.flyMan\_fly1;

}

if(count1==10)

{

EnemyLeftRight.Image = Game1.Properties.Resources.flyMan\_jump2;

}

if(count1==15)

{

EnemyLeftRight.Image = Game1.Properties.Resources.flyMan\_still\_fly;

}

if(count1==20)

{

EnemyLeftRight.Image = Game1.Properties.Resources.flyMan\_still\_stand;

count1 = 0;

}

}

private void pictureBox1\_Click\_1(object sender, EventArgs e)

{

}

private void GameTimerEvent(object sender, EventArgs e)

{

g.Update();

ChangePlayerImages();

ChangePicsEnemyYellow();

ChangePicsEnemyHorizontal();

lblScore.Text = "Score: " + Score + "/20 ";

LivesRemainimg.Text = "Lives: " + TotalLives;

pbPlayer.BringToFront();

//DetectCollisionWithBoundary();

pbCactus.Left -= PipeSpeed;

pbMushRoom.Left -= PipeSpeed;

pbCloud.Left -= PipeSpeed;

RepeatPipes();

if (Score > 6)

{

PipeSpeed = 10;

}

WinGame();

EndGame();

}

private void WinGame()

{

if(Score>20)

{

GameTimer.Enabled = false;

Timer.Enabled = false;

lblScore.Text += "Game Won";

Image img = Game1.Properties.Resources.download\_\_2\_;

EndGameForm moreForm = new EndGameForm(img);

DialogResult Result = moreForm.ShowDialog();

if (Result == DialogResult.Yes)

{

this.Close();

Flappy\_Bird\_Game g = new Flappy\_Bird\_Game();

g.Show();

}

if (Result == DialogResult.No)

{

this.Hide();

Form1 f = new Form1();

f.Show();

}

}

}

private void Timer\_Tick(object sender, EventArgs e)

{

lblTime.Text = "Time Left: "+time-- + " sec";

}

public void EndGame()

{

if (TotalLives <= 0 || time<=0)

{

GameTimer.Stop();

Timer.Stop();

lblScore.Text += " Game Over";

Image img = Game1.Properties.Resources.download\_\_1\_;

EndGameForm moreForm = new EndGameForm(img);

DialogResult Result= moreForm.ShowDialog();

if(Result==DialogResult.Yes)

{

Flappy\_Bird\_Game g = new Flappy\_Bird\_Game();

g.Show();

RestartAgain();

}

if (Result == DialogResult.No)

{

this.Hide();

Form1 f = new Form1();

f.Show();

}

}

}

public void RestartGame()

{

Point player = new Point();

player.X = 20;

player.Y = 20;

pbPlayer.Location = player;

EnemyLeftRight.Location = EnemyLeftRightPoint;

EnemyZigZagBlue.Location = EnemyZigZagBluePoint;

EnemyZigZagYellow.Location = EnemyZigZagYellowPoint;

Pipe1.X = 391; Pipe1.Y = 278;

PipeBottom1.Location = Pipe1;

Pipe2.X = 419; Pipe2.Y = 0;

PipeTop1.Location = Pipe2;

Pipe3.X = 622; Pipe3.Y = 226;

PipeBottom2.Location = Pipe3;

Pipe4.X = 674; Pipe4.Y = 0;

PipeTop2.Location = Pipe4;

}

public void RepeatPipes()

{

if (pbCactus.Left <= -50)

{

pbCactus.Left = 825;

}

if (pbMushRoom.Left <= -50)

{

pbMushRoom.Left = 825;

}

if (pbCloud.Left <= -50)

{

pbCloud.Left = 825;

}

}

private void GameKeyisDown(object sender, KeyEventArgs e)

{

g.KeyPressed(e.KeyCode);

g.KeyPressedForFiring(e.KeyCode);

}

**GameObject Class:**

public class GameObject

{

private PictureBox pb;

private IMovement Movement;

private ObjectTypes OType;

private ProgressBar Health;

public PictureBox Pb { get => pb; set => pb = value; }

public IMovement Movement1 { get => Movement; set => Movement = value; }

public ObjectTypes OType1 { get => OType; set => OType = value; }

public ProgressBar Health1 { get => Health; set => Health = value; }

public GameObject(Image img,ObjectTypes OType, int Top, int Left, IMovement Movement)

{

pb = new PictureBox();

Health1 = new ProgressBar();

pb.Image = img;

pb.Height = img.Height;

pb.Width = img.Width;

pb.BackColor = Color.Transparent;

pb.Top = Top;

pb.Left = Left;

Health.Value = 100;

Health.Step = 10;

Health.Height = 9;

Health.Width = 80;

Health1.Left = pb.Left;

Health1.Top = pb.Top -( pb.Height + 50);

this.OType1 = OType;

this.Movement1 = Movement;

}

public GameObject(PictureBox pb,ObjectTypes Otype,IMovement Movement)

{

this.pb = pb;

this.OType = Otype;

this.Movement = Movement;

}

public GameObject(PictureBox pb)

{

this.pb = pb;

}

public void GameObjectMove()

{

pb.Location = Movement1.Move(pb.Location/\*,pb.Image\*/);

}}

**IGame Interface:**

public interface IGame

{

void RaisePlayerDieEvent(PictureBox pb,ProgressBar Health);

void RaiseFiringEvent(PictureBox FireImg);

void RaiseCoinDieEvent(PictureBox CoinImg);

void RaiseHeartDieEvent(PictureBox HeartImg);

void RaiseGroundDieEvent(PictureBox pbGround);

void RaisePipeCollisionEvent(PictureBox pbPlayer);

void RaiseBoundaryCollidingEvent(bool flag);

void RaiseFireCollisionWithEnemyEvent(PictureBox pbEnemy,PictureBox PlayerFire, ProgressBar pbBar);

void RaisePowerUpsEvent(PictureBox pbPowerUps, PictureBox pbPlayer, ProgressBar BarPlayer);

}

**Game Class:**

public class Game:IGame

{

private List<GameObject> GameObjectsList;

private List<CollisionClass> CollisionList;

private List<FireClass> FiringList;

private int Gravity;

public event EventHandler onAddingGameObject;

public event EventHandler OnAddingProgressBar;

public event EventHandler OnPlayerDie;

public event EventHandler OnRemoveProgressBar;

public event EventHandler OnAddingFires;

public event EventHandler OnCoinDie;

public event EventHandler OnHeartDie;

public event EventHandler OnGroundTouch;

public event EventHandler OnPipeCollision;

public event EventHandler OnBoundaryCollision;

public event EventHandler OnPlayerFire;

public event EventHandler OnEnemyFire;

public event EventHandler DiagonalEnemyFire;

public event EventHandler RightDiagonalEnemyFire;

public event EventHandler OnFireDead;

public event EventHandler OnFireCollisionWithEnemy;

public event EventHandler IncreasePlayerPowerByPowerUps;

public Game(int Gravity)

{

GameObjectsList = new List<GameObject>();

//ProgressBarList = new List<ProgressBar>();

CollisionList = new List<CollisionClass>();

FiringList = new List<FireClass>();

this.Gravity = Gravity;

}

public PictureBox AddGameObject(Image img,ObjectTypes OType, int Top, int Left, IMovement Movement)

{

GameObject go = new GameObject(img,OType, Top, Left, Movement);

GameObjectsList.Add(go);

onAddingGameObject?.Invoke(go.Pb, EventArgs.Empty);

if(OType==ObjectTypes.enemy || OType==ObjectTypes.player)

{

OnAddingProgressBar?.Invoke(go.Health1, EventArgs.Empty);

}

return go.Pb;

}

public void AddGameObject(GameObject g)

{

GameObjectsList.Add(g);

}

public void MoveProgressBar()

{

int Left, Top;

Point p = new Point();

foreach(GameObject g in GameObjectsList)

{

if (g.OType1 == ObjectTypes.player || g.OType1 == ObjectTypes.enemy)

{

Left = g.Pb.Location.X;

Top = g.Pb.Location.Y;

p.X = Left;

p.Y = Top + g.Pb.Height + 5;

g.Health1.Location = p;

}

}

}

int Count = 0, count1 = 0;

public void Update()

{

DetectCollision();

MoveProgressBar();

foreach (GameObject g in GameObjectsList)

{

g.GameObjectMove();

}

EnemyFire();

RemoveFire();

}

public void KeyPressed(Keys KeyCode)

{

foreach (GameObject g in GameObjectsList)

{

if (g.Movement1.GetType() == typeof(KeyBoardMovement))

{

KeyBoardMovement keyBoardHandle = (KeyBoardMovement)g.Movement1;

keyBoardHandle.KeyPresssedBuUser(KeyCode);

}

}

}

public void KeyPressedForFiring(Keys KeyCode )

{

if (KeyCode == Keys.Space)

{

foreach(GameObject g in GameObjectsList)

{

if(g.OType1==ObjectTypes.player)

{

OnPlayerFire?.Invoke(g.Pb, EventArgs.Empty);

break;

}

}

}

}

public void AddFiresInFiringList(FireClass f)

{

FiringList.Add(f);

}

public void EnemyFire()

{

Count++;

if(Count==60)

{

for(int i=0;i<GameObjectsList.Count;i++)

{

if(GameObjectsList[i].OType1 == ObjectTypes.enemy)

{

DiagonalEnemyFire?.Invoke(GameObjectsList[i].Pb, EventArgs.Empty);

break;

}

}

}

if (Count == 70)

{

for(int i=0;i<GameObjectsList.Count;i++)

{

if(GameObjectsList[i].OType1==ObjectTypes.enemy)

{

count1++;

if (count1 >1 && count1<5)

{

OnEnemyFire?.Invoke(GameObjectsList[i].Pb, EventArgs.Empty);

}

if(count1>=5)

{

RightDiagonalEnemyFire?.Invoke(GameObjectsList[i].Pb, EventArgs.Empty);

}

}

}

Count = 0;

count1 = 0;

}

}

public void RemoveFire()

{

foreach(GameObject g in GameObjectsList)

{

if(g.OType1==ObjectTypes.EnemyFires || g.OType1==ObjectTypes.playerFires)

{

if(g.Pb.Location.X<0)

{

GameObjectsList.Remove(g);

}

else if(g.Pb.Location.X>1000)

{

GameObjectsList.Remove(g);

}

else if(g.Pb.Location.Y>600)

{

GameObjectsList.Remove(g);

}

OnFireDead?.Invoke(g.Pb, EventArgs.Empty);

break;

}

}

}

public void AddCollision(CollisionClass c)

{

CollisionList.Add(c);

}

public void RaisePlayerDieEvent(PictureBox PlayerGameObject,ProgressBar Health)

{

OnPlayerDie?.Invoke(PlayerGameObject, EventArgs.Empty);

OnRemoveProgressBar?.Invoke(Health, EventArgs.Empty);

}

public void DetectCollision()

{

int valu = 10;

for(int i=0;i<GameObjectsList.Count;i++)

{

for(int j=0;j<GameObjectsList.Count;j++)

{

if(GameObjectsList[i].Pb.Bounds.IntersectsWith(GameObjectsList[j].Pb.Bounds))

{ for(int k=0;k<CollisionList.Count;k++)

{

if(GameObjectsList[i].OType1==CollisionList[k].G1 && GameObjectsList[j].OType1==CollisionList[k].G2)

{

CollisionList[k].Behaviour1.PerformAction(this, GameObjectsList[i], GameObjectsList[j]);

break;

}

}

}

}

}

}

public void AddFires(FireClass f)

{

FiringList.Add(f);

//RaiseFiringEvent(f.Pbfire);

}

public void RaiseFiringEvent(PictureBox FireImg)

{

OnAddingFires?.Invoke(FireImg, EventArgs.Empty);

}

public void RaiseFireCollisionWithEnemyEvent(PictureBox pbEnemy,PictureBox pbPlayerFire,ProgressBar pbBarEnemy)

{

if(pbBarEnemy.Value<=0)

{

OnRemoveProgressBar?.Invoke(pbBarEnemy, EventArgs.Empty);

OnFireDead?.Invoke(pbEnemy, EventArgs.Empty);

Remove(pbEnemy);

}

if (pbBarEnemy.Value > 0)

{

OnFireCollisionWithEnemy?.Invoke(pbBarEnemy, EventArgs.Empty);

OnFireDead?.Invoke(pbPlayerFire, EventArgs.Empty);

Remove(pbPlayerFire);

}

}

public void Remove(PictureBox pb)

{

for(int i=0;i<GameObjectsList.Count;i++)

{

if(GameObjectsList[i].Pb==pb)

{

GameObjectsList.Remove(GameObjectsList[i]);

}

}

}

public void AddCoinInGameObjectList(GameObject g)

{

GameObjectsList.Add(g);

}

public void RaiseCoinDieEvent(PictureBox CoinImg)

{

OnCoinDie?.Invoke(CoinImg, EventArgs.Empty);

Remove(CoinImg);

}

public void RaiseHeartDieEvent(PictureBox HeartImg)

{

Remove(HeartImg);

OnHeartDie?.Invoke(HeartImg, EventArgs.Empty);

}

public void RaiseGroundDieEvent(PictureBox pbGround)

{

OnGroundTouch?.Invoke(pbGround, EventArgs.Empty);

}

public void RaisePipeCollisionEvent(PictureBox pbPlayer)

{

OnPipeCollision?.Invoke(pbPlayer, EventArgs.Empty);

}

public void RaiseBoundaryCollidingEvent(bool flag)

{

OnBoundaryCollision?.Invoke(flag, EventArgs.Empty);

}

public void RaisePowerUpsEvent(PictureBox pbPowerUps,PictureBox pbPlayer,ProgressBar BarPlayer)

{

if (BarPlayer.Value < 80)

{

IncreasePlayerPowerByPowerUps?.Invoke(BarPlayer, EventArgs.Empty);

OnFireDead?.Invoke(pbPowerUps, EventArgs.Empty);

}

}

}

**ObjectTypes enum:**

public enum ObjectTypes

{

player,

playerFires,

enemy,

EnemyFires,

Pipes,

Coins,

Heart,

PowerUps,

Pills,

Ground,

Boundary,

}

**Fire Class:**

public class FireClass:IMovement

{

private int Speed;

private Point Boundary;

private string Direction;

public FireClass(int Speed,Point Boundary,string Direction)

{

this.Speed = Speed;

this.Boundary = Boundary;

this.Direction = Direction;

}

public Point Move(Point Location)

{

if(Direction=="Right")

{

Location.X += Speed;

}

if(Direction=="Left")

{

Location.X -= Speed;

}

if(Direction=="Up")

{

Location.Y -= Speed;

}

if(Direction=="Down")

{

Location.Y += Speed;

}

if(Direction=="LeftDiagonal")

{

Location.X -= Speed;

Location.Y += Speed;

}

if(Direction=="RightDiagonal")

{

Location.X -= Speed;

Location.Y -= Speed;

}

return Location; }

**IMovement Interface:**

public interface IMovement

{

Point Move(Point Location/\*,Image img\*/);

}

**KeyboardMovement Class:**

public class KeyBoardMovement :IMovement

{

private int Speed;

private Point Boundary;

private string Direction;

private string ArrowAction;

private bool FireAction;

public KeyBoardMovement(int Speed, Point Boundary, string Direction)

{

this.Speed = Speed;

this.Boundary = Boundary;

this.Direction = Direction;

FireAction = false;

}

public void KeyPresssedBuUser(Keys KeyCode)

{

if (KeyCode == Keys.Up)

{

ArrowAction = "Up";

}

else if (KeyCode == Keys.Down)

{

ArrowAction = "Down";

}

}

public Point Move(Point Location)

{

if (ArrowAction == "Up")

{

Location.Y -= Speed;

}

if (ArrowAction == "Down")

{

Location.Y += Speed;

}

return Location;

}

}

**ZigZag Class:**

public class ZigZag:IMovement

{

private int Speed;

private Point Boundary;

private string Direction;

private int Offset;

public ZigZag(int Speed, Point Boundary, string Direction)

{

this.Speed = Speed;

this.Boundary = Boundary;

this.Direction = Direction;

Offset = 100;

}

public Point Move(Point Location)

{

if (Direction == "Up")

{

Location.X -= Speed;

Location.Y -= Speed;

}

if (Direction == "Down")

{

Location.X -= Speed;

Location.Y += Speed;

}

if (Location.Y >= 391)

{

Direction = "Up";

}

if (Location.Y <= 0)

{

Direction = "Down";

}

if (Location.X <= 0)

{

Location.X = 1800;

}

return Location;

}

}

**Horizontal Class:**

public class Horizontal :IMovement

{

private int Speed;

private Point Boundary;

private string Direction;

private int Offset;

int count = 0;

public Horizontal()

{

}

public Horizontal(int Speed, Point Boundary, string Direction)

{

this.Speed = Speed;

this.Boundary = Boundary;

this.Direction = Direction;

Offset = 100;

}

public Point Move(Point Location/\*,Image img\*/)

{

if (Direction == "Left")

{

int SetStartingLocation = 1800;

if (Location.X <= -50)

{

Location.X = SetStartingLocation;

}

Location.X -= Speed;

}

if(Direction=="Right")

{

Location.X += Speed;

}

return Location;

}

**Collision Class:**

public class CollisionClass

{

private ObjectTypes g1;

private ObjectTypes g2;

private ICollisionAction Behaviour;

public CollisionClass(ObjectTypes g1,ObjectTypes g2,ICollisionAction Action)

{

this.G1 = g1;

this.G2 = g2;

this.Behaviour1 = Action;

}

public ObjectTypes G1 { get => g1; set => g1 = value; }

public ObjectTypes G2 { get => g2; set => g2 = value; }

public ICollisionAction Behaviour1 { get => Behaviour; set => Behaviour = value; }

}

**IColissionAction Interface:**

public interface ICollisionAction

{

void PerformAction(IGame game, GameObject Source1, GameObject Source2);

}

**EnemyColission Class:**

public class EnemyCollision:ICollisionAction

{

public void PerformAction(IGame Game, GameObject Source1, GameObject Source2)

{

GameObject Enemy;

if(Source1.OType1==ObjectTypes.enemy)

{

Enemy = Source1;

}

else

{

Enemy = Source2;

}

Game.RaisePlayerDieEvent(Enemy.Pb,Enemy.Health1);

}

}

**Coin Colission:**

public class CoinCollision:ICollisionAction

{

public void PerformAction(IGame Game, GameObject Source1, GameObject Source2)

{

GameObject Coin;

if (Source1.OType1 == ObjectTypes.Coins)

{

Coin = Source1;

}

else

{

Coin = Source2;

}

Game.RaiseCoinDieEvent(Coin.Pb);

}

}

**GroundColission Class:**

public class GroundCollision:ICollisionAction

{

public void PerformAction(IGame Game, GameObject Source1, GameObject Source2)

{

GameObject Ground;

if (Source1.OType1 == ObjectTypes.Ground)

{

Ground = Source1;

}

else

{

Ground = Source2;

}

Game.RaiseGroundDieEvent(Ground.Pb);

}

}

**PowerUpsColission Class:**

public void PerformAction(IGame Game, GameObject Source1, GameObject Source2)

{

GameObject PowerUps;

GameObject Player;

if (Source1.OType1 == ObjectTypes.PowerUps)

{

PowerUps = Source1;

Player = Source2;

}

else

{

PowerUps = Source2;

Player = Source1;

}

Game.RaisePowerUpsEvent(PowerUps.Pb,Player.Pb,Player.Health1);

}

**WallColission Class:**

public class WallCollision:ICollisionAction

{

public void PerformAction(IGame Game, GameObject Source1, GameObject Source2)

{

GameObject PlayerCollidingwithWall;

if(Source1.OType1==ObjectTypes.Pipes)

{

PlayerCollidingwithWall = Source1;

}

else

{

PlayerCollidingwithWall = Source2;

}

Game.RaisePipeCollisionEvent(PlayerCollidingwithWall.Pb);

}

}