**Project: Game Development with Framework**



Session: 2021 – 2024

**Submitted by:**

Yahya M.Mirza 2021-CS-11

**Supervised by:**

Dr. Muhammad Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Content**

1. Short description of Game-------------------------------------------------page 3
2. Game Character Description----------------------------------------------page 3-4
3. Rules and Interactions-----------------------------------------------------page 5
4. Goals of Game--------------------------------------------------------------page 5
5. Features of Framework----------------------------------------------------page 6
6. Class Diagram -------------------------------------------------------------page 7
7. Sequence Diagram-------------------------------------------------------page 8- 10
8. Example To Use Framework-------------------------------------------page 11-16
9. Complete Code-----------------------------------------------------------page#17-50

**Short Description of Game**

This game features the core 2d shooting and interaction/collision mechanics and allows you to enjoy the gesture of shooting.

The game has 2 levels, 1 with less difficulty while other with a more difficult enemy to kill. The game is developed keeping in mind all extendibility criteria using framework.

**Game Character Description**

**Main Player**

****

Main player the player controlled through key Codes, the player can shoot through key A

**Ammo/Score**

****

The ammo or call it score or coin it is just score collecting it increases the main score

**Gollem**

****

The golem enemy continuously shoots lasers from the lower tip

**Turret**

****

Emits the fire from its mouth continuously but it dies within 5 -6 bullets

**Spaceship**

****

Hovers above the stage and oscillates in a horizontal pattern

**Turret**

****

Hovers above the stage 2 and oscillates in a diagonal pattern

**Rules And Interaction**

**Stage 1:**

Don’t collide with turret or the turret bullets and the golem bullets or you will die.

**Stage 2:**

The diagonal enemy moves faster and is slightly harder to kill.

**Goals of the Game**

The goal of the game was to learn coding with framework and how to extend the functionality not within ourselves but within others too, secondly game was developed keeping in mind the core shooting mechanics so its environment is created for shooters but at very low level

The goal is to eliminate all the enemies of stage 1 and 2.

**Features of Framework**

**The Framework has 3 basic features**

1. **Movement**

* Horizontal oscillation
* Horizontal
* Freefall
* Random Appearance of the health and score.
* Keyboard Movement With progress-bar And Shooting

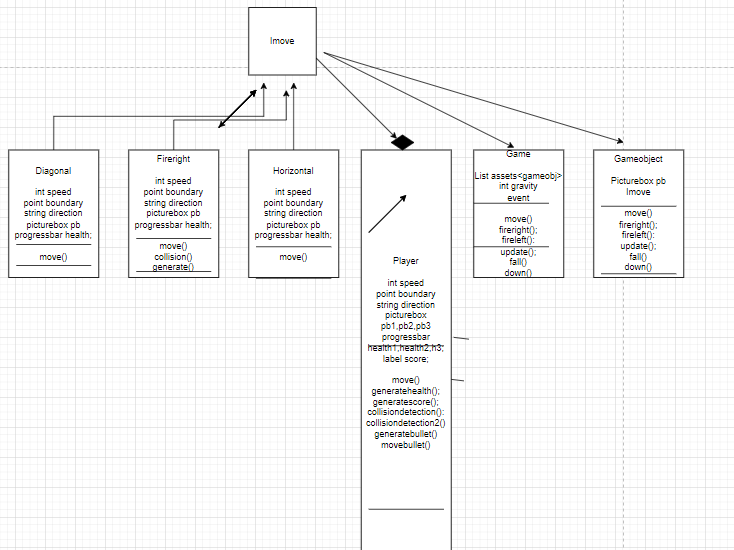
1. **Firing**

* Fire for player or the enemy.

**3-Collision**

* Any collision according to game can be extended

**Class Diagram**

****

**Sequence Diagram**

**Add game object**

Imove

Stage1

Ikeyboard

game

Program

MainPlayer()

New movement

New movement

onobjectadded

**Move**

Imove

Stage1

Ikeyboard

game

Program

Mainiskeypressed(e.keycode)

iskeypressed(e.keycode)

move

onobjectadded

**Move update**

Imove

Stage1

Ikeyboard

game

Program

Mainmove()

Movement(point l)

Movement(prg\_bar)

point

**Collision**

Stage1

game

Program

MainPlayer(collision)

detectcollision

**Fire**

Imove

Stage1

fireright

Program

Mainfireright(parameter)

firemoves

onobjectadded.invoke

**Example To Use Framework**

1. **Movement**

IMovement is an interface which can extend any type of simple movement

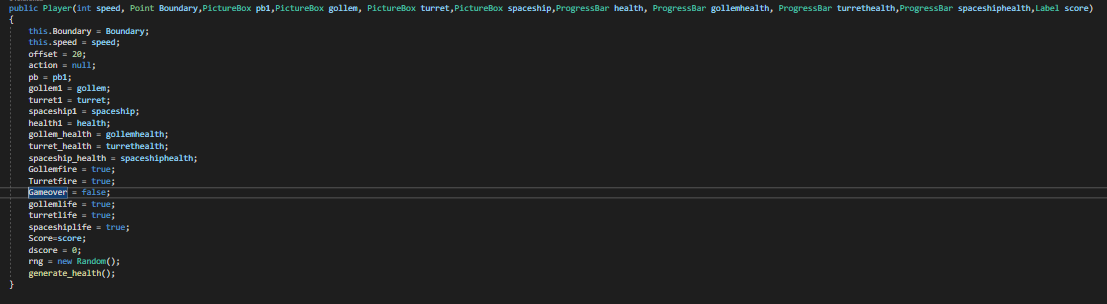
Such as vertical, horizontal and diagonal etc.

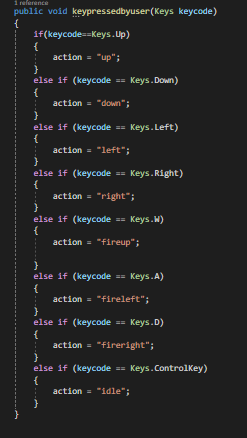


It implements IMovement and also it can be called as



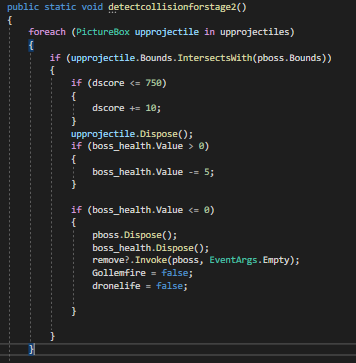
The Movement With KeyBoard can also be created such as using IMovement for simple use but it you want to make your progress bar to also move with your player or you need to change the image of the player then you just need to pass it within the constructor





**2-Collision**

You just pass the enemy picture boxes and the health bar in the player class and it will detect the collision.

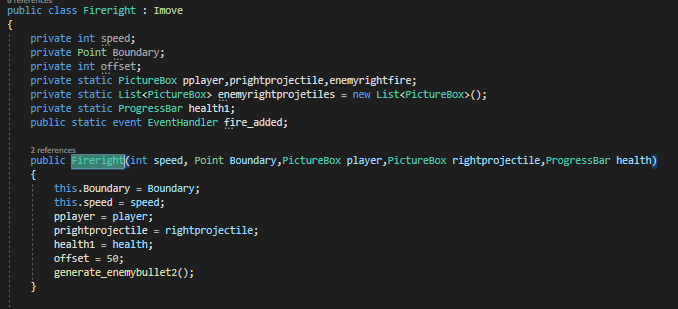


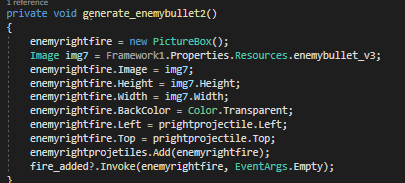
Called as

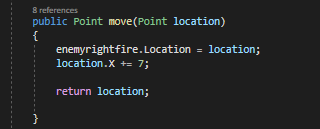


**3-Fire generation**

The fire has a separate class called the fire right which generates the fires from the right side of a certain location which is passed in the constructor of the fire right class. The projectile picturebox is also passed.

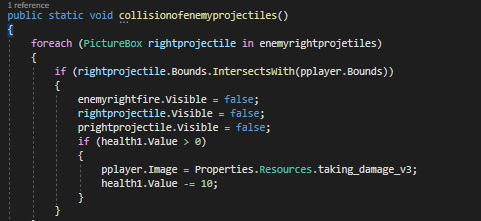






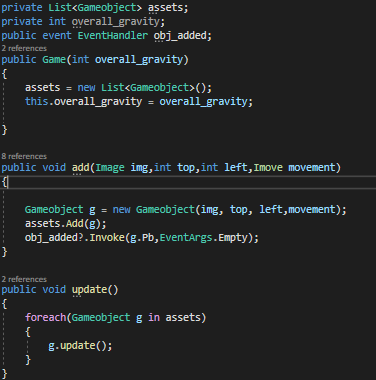
**4-Fire collision**

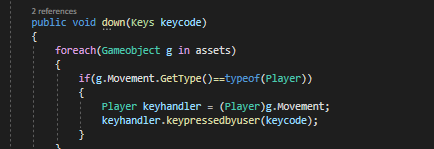
The fireright class has a static method to detect the collision



**5-Game Class**

The game class is one of the main classes, it keeps tracks of all the different game objects and their behaviors.it also provides the movement of the game objects.

****



**Complete Code**

**Stage 1**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Framework1.Core;

using Framework1.Movement;

namespace User

{

public partial class Form1 : Form

{

Game g;

Point boundary;

int firelatency,firecurrentlatency,downfirelatency,downfirecurrentlatency;

PictureBox rightprojectile,downprojectile;

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

rightprojectile = new PictureBox();

rightprojectile.Image = User.Properties.Resources.enemybullet\_v3;

rightprojectile.Top = 40;

rightprojectile.Left = 395;

downprojectile = new PictureBox();

downprojectile.Image = User.Properties.Resources.gollemprojectile;

downprojectile.Top = 435;

downprojectile.Left = 150;

firelatency = 25;

firecurrentlatency = 0;

downfirelatency = 25;

downfirecurrentlatency = 0;

g = new Game(10);

g.obj\_added += new EventHandler(addincontrol);

Player.fire\_added += new EventHandler(addincontrol);

Player.player\_added += new EventHandler(addincontrol);

Player.remove += new EventHandler(removefromcontrol);

Horizontal.obj\_added += new EventHandler(addincontrol);

Fireright.fire\_added += new EventHandler(addincontrol);

Firedown.fire\_added += new EventHandler(addincontrol);

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

g.add(spaceship\_pb.Image,spaceship\_pb.Top,spaceship\_pb.Left,new Horizontal(2,boundary,"right",spaceship\_pb,spaceship\_prg));

g.add(player.Image,player.Top,player.Left,new Player(6,boundary,player,gollem,turret,spaceship\_pb,player\_prg,gollem\_prgbar,turret\_prgbar,spaceship\_prg,score\_lbl));

g.add(rightprojectile.Image,rightprojectile.Top,rightprojectile.Left, new Fireright(7,boundary,player,rightprojectile, player\_prg));

g.add(downprojectile.Image, downprojectile.Top, downprojectile.Left,new Firedown(7, boundary, player, downprojectile, player\_prg));

}

private void pictureBox2\_Click(object sender, EventArgs e)

{

}

private void addincontrol(object sender, EventArgs e)

{

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

}

private void removefromcontrol(object sender, EventArgs e)

{

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

}

private void Form1\_KeyUp(object sender, KeyEventArgs e)

{

}

private void score\_lbl\_Click(object sender, EventArgs e)

{

}

private void pictureBox4\_Click(object sender, EventArgs e)

{

}

private void timer1\_Tick(object sender, EventArgs e)

{

firecurrentlatency++;

downfirecurrentlatency++;

Player.moveupprojectile();

Player.moveleftprojectile();

Player.moverightprojectile();

Player.detectcollision();

Fireright.moveturretenemyprojectile();

Fireright.collisionofenemyprojectiles();

Firedown.movegollemenemyprojectile();

Firedown.collisionofenemyprojectiles();

g.update();

g.fall();

g.wave();

if (firecurrentlatency == firelatency)

{

if (Player.Turretfire)

{

g.add(rightprojectile.Image, rightprojectile.Top, rightprojectile.Left, new Fireright(7, boundary, player, rightprojectile, player\_prg));

g.fireright();

firecurrentlatency = 0;

}

}

if (downfirecurrentlatency == downfirelatency)

{

if (Player.Gollemfire)

{

g.add(downprojectile.Image, downprojectile.Top, downprojectile.Left, new Firedown(7, boundary, player, downprojectile, player\_prg));

g.firedown();

downfirecurrentlatency = 0;

}

}

if(Player.Gameover)

{

stage1timer.Enabled = false;

this.Hide();

gameover f1 = new gameover();

f1.ShowDialog();

}

if (Player.Gollemlife == false && Player.Turretlife == false && Player.Spaceshiplife == false)

{

stage1timer.Enabled = false;

this.Hide();

Form2 f1 = new Form2();

f1.ShowDialog();

}

}

private void Form1\_KeyDown(object sender, KeyEventArgs e)

{

g.down(e.KeyCode);

}

}

}

**Stage 2**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Framework1.Core;

using Framework1.Movement;

namespace User

{

public partial class Form2 : Form

{

Game g;

Point boundary;

int firelatency, firecurrentlatency;

PictureBox boss,leftprojectile;

public Form2()

{

InitializeComponent();

}

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

{

shooter.Visible = false;

shooter\_prg.Visible = false;

leftprojectile = new PictureBox();

leftprojectile.Image = User.Properties.Resources.projectile\_left\_c;

leftprojectile.Top = 650;

leftprojectile.Left = 395;

boss = new PictureBox();

boss.Image = User.Properties.Resources.drone;

boss.Top = 20;

boss.Left = 10;

firelatency = 25;

firecurrentlatency = 0;

g = new Game(10);

g.obj\_added += new EventHandler(addincontrol);

Player.fire\_added += new EventHandler(addincontrol);

Player.player\_added += new EventHandler(addincontrol);

Diagonal.fire\_added += new EventHandler(addincontrol);

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

g.add(player.Image, player.Top, player.Left, new Player(6,boundary,player,shooter,boss,shooter\_prg,boss\_prg,player\_prg,score\_lbl));

g.add(boss.Image,boss.Top,boss.Left,new Diagonal(1,boundary,"left",boss,boss\_prg));

//g.add(leftprojectile.Image, leftprojectile.Top, leftprojectile.Left, new Fireleft(7, boundary, player, leftprojectile, player\_prg));

}

private void addincontrol(object sender, EventArgs e)

{

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

}

private void timer2\_Tick(object sender, EventArgs e)

{

}

private void Form2\_KeyDown(object sender, KeyEventArgs e)

{

g.down(e.KeyCode);

}

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

{

}

private void timer1\_Tick(object sender, EventArgs e)

{

firecurrentlatency++;

if (firecurrentlatency == firelatency)

{

if (Player.Turretfire)

{

// g.add(User.Properties.Resources.projectile\_left\_c, 650, 395, new Fireleft(7, boundary, player, leftprojectile, player\_prg));

// g.fireleft();

firecurrentlatency = 0;

}

}

Player.moveupprojectile();

Player.moveleftprojectile();

Player.moverightprojectile();

Player.detectcollisionforstage2();

// Fireleft.moveshooterenemyprojectile();

//Fireleft.collisionofshooterenemyprojectiles();

g.update();

g.fall();

g.wave();

if (Player.Gameover)

{

stage2timer.Enabled = false;

this.Hide();

gameover f1 = new gameover();

f1.ShowDialog();

}

if (Player.Dronelife == false)

{

stage2timer.Enabled = false;

this.Hide();

gameover f1 = new gameover();

f1.ShowDialog();

}

}

}

}

**Horizontal Class**

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace Framework1.Movement

{

public class Horizontal:Imove

{

private int speed;

private Point Boundary;

private string direction;

private PictureBox pb;

private int offset;

private static ProgressBar spaceship\_health;

public static event EventHandler obj\_added;

private Random rng;

public Horizontal(int speed,Point Boundary,string direction,PictureBox pb,ProgressBar spaceshiphealth)

{

this.Boundary = Boundary;

this.speed = speed;

this.direction = direction;

this.pb = pb;

spaceship\_health=spaceshiphealth;

offset =140;

}

public Point move(Point location)

{

// pb.Location = location;

location.Y -= 100;

if (direction == "left")

{

pb.Left -= speed;

spaceship\_health.Left -= speed;

}

else if (direction == "right")

{

pb.Left += speed;

spaceship\_health.Left += speed;

}

if (pb.Left <=0)

{

pb.Image = Properties.Resources.Ship6;

direction = "right";

}

else if((pb.Left+ pb.Width) + 10 >= Boundary.X)

{

pb.Image = Properties.Resources.Ship6inverted;

direction = "left";

}

return location;

}

}

}

**Fireright Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

using Framework1.Core;

namespace Framework1.Movement

{

public class Fireright : Imove

{

private int speed;

private Point Boundary;

private int offset;

private static PictureBox pplayer,prightprojectile,enemyrightfire;

private static List<PictureBox> enemyrightprojetiles = new List<PictureBox>();

private static ProgressBar health1;

public static event EventHandler fire\_added;

public Fireright(int speed, Point Boundary,PictureBox player,PictureBox rightprojectile,ProgressBar health)

{

this.Boundary = Boundary;

this.speed = speed;

pplayer = player;

prightprojectile = rightprojectile;

health1 = health;

offset = 50;

generate\_enemybullet2();

}

public static void collisionofenemyprojectiles()

{

foreach (PictureBox rightprojectile in enemyrightprojetiles)

{

if (rightprojectile.Bounds.IntersectsWith(pplayer.Bounds))

{

enemyrightfire.Visible = false;

rightprojectile.Visible = false;

prightprojectile.Visible = false;

if (health1.Value > 0)

{

pplayer.Image = Properties.Resources.taking\_damage\_v3;

health1.Value -= 10;

}

}

}

}

private void generate\_enemybullet2()

{

enemyrightfire = new PictureBox();

Image img7 = Framework1.Properties.Resources.enemybullet\_v3;

enemyrightfire.Image = img7;

enemyrightfire.Height = img7.Height;

enemyrightfire.Width = img7.Width;

enemyrightfire.BackColor = Color.Transparent;

enemyrightfire.Left = prightprojectile.Left;

enemyrightfire.Top = prightprojectile.Top;

enemyrightprojetiles.Add(enemyrightfire);

fire\_added?.Invoke(enemyrightfire, EventArgs.Empty);

}

public static void moveturretenemyprojectile()

{

// if (turretfire)

// {

foreach (PictureBox enemyrightfire in enemyrightprojetiles)

{

fire\_added?.Invoke(enemyrightfire, EventArgs.Empty);

enemyrightfire.Left += 7;

}

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

{

if (enemyrightprojetiles[i].Bottom <= 0)

{

enemyrightprojetiles.Remove(enemyrightprojetiles[i]);

}

}

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

{

if ( enemyrightprojetiles[i].Visible == false)

{

enemyrightprojetiles.Remove(enemyrightprojetiles[i]);

}

}

//}

}

public Point move(Point location)

{

enemyrightfire.Location = location;

location.X += 7;

return location;

}

**Fireleft Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace Framework1.Movement

{

public class Fireleft:Imove

{

private int speed;

private Point Boundary;

private int offset;

private static PictureBox pplayer, pleftprojectile,enemyleftfire;

private static List<PictureBox> enemyleftprojetiles = new List<PictureBox>();

private static ProgressBar health1;

public static event EventHandler fire\_added;

public Fireleft(int speed, Point Boundary,PictureBox player,PictureBox leftprojectile,ProgressBar playerhealth)

{

this.Boundary = Boundary;

this.speed = speed;

offset = 50;

pplayer = player;

pleftprojectile = leftprojectile;

health1 = playerhealth;

generate\_enemybullet2();

}

public static void collisionofshooterenemyprojectiles()

{

foreach (PictureBox enemyleftfire in enemyleftprojetiles)

{

if (enemyleftfire.Bounds.IntersectsWith(pplayer.Bounds))

{

enemyleftfire.Dispose();

// leftprojectile.Dispose();

pleftprojectile.Dispose();

if (health1.Value > 0)

{

pplayer.Image = Properties.Resources.taking\_damage\_v3;

health1.Value -= 1;

}

}

}

}

private void generate\_enemybullet2()

{

enemyleftfire = new PictureBox();

Image img7 = Framework1.Properties.Resources.enemybullet\_v3;

enemyleftfire.Image = img7;

enemyleftfire.Height = img7.Height;

enemyleftfire.Width = img7.Width;

enemyleftfire.BackColor = Color.Transparent;

enemyleftfire.Left = pleftprojectile.Left;

enemyleftfire.Top = pleftprojectile.Top;

enemyleftprojetiles.Add(enemyleftfire);

fire\_added?.Invoke(enemyleftfire, EventArgs.Empty);

}

public static void moveshooterenemyprojectile()

{

// if (turretfire)

// {

foreach (PictureBox enemyleftfire in enemyleftprojetiles)

{

fire\_added?.Invoke(enemyleftfire, EventArgs.Empty);

enemyleftfire.Left -= 7;

}

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

{

if (enemyleftprojetiles[i].Bottom <= 0)

{

enemyleftprojetiles.Remove(enemyleftprojetiles[i]);

}

}

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

{

if (enemyleftprojetiles[i].Visible == false)

{

enemyleftprojetiles.Remove(enemyleftprojetiles[i]);

}

}

//}

}

public Point move(Point location)

{

location.X -= 7;

return location;

}

}

}

**Player Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

using Framework1.Core;

namespace Framework1.Movement

{

public class Player:Imove

{

private int speed;

private static int dscore;

private Point Boundary;

private int offset;

private string action;

private static List<PictureBox> upprojectiles = new List<PictureBox>();

private static List<PictureBox> leftprojectiles = new List<PictureBox>();

private static List<PictureBox> rightprojectiles = new List<PictureBox>();

private PictureBox upfire, leftfire, rightfire;

private static PictureBox gollem1, turret1, spaceship1, pb,health,pplayer,pshooter,pboss,coin;

private static ProgressBar health1,gollem\_health,turret\_health,spaceship\_health,shooter\_health,boss\_health,player\_health;

public static event EventHandler fire\_added,player\_added,remove;

private static bool gollemfire, turretfire,gameover;

private static bool gollemlife, turretlife, spaceshiplife, dronelife;

private static Label Score;

private Random rng;

public static bool Gollemfire { get => gollemfire; set => gollemfire = value; }

public static bool Turretfire { get => turretfire; set => turretfire = value; }

public static bool Gameover { get => gameover; set => gameover = value; }

public static bool Gollemlife { get => gollemlife; set => gollemlife = value; }

public static bool Turretlife { get => turretlife; set => turretlife = value; }

public static bool Spaceshiplife { get => spaceshiplife; set => spaceshiplife = value; }

public static bool Dronelife { get => dronelife; set => dronelife = value; }

public Player(int speed, Point Boundary,PictureBox pb1,PictureBox gollem, PictureBox turret,PictureBox spaceship,ProgressBar health, ProgressBar gollemhealth, ProgressBar turrethealth,ProgressBar spaceshiphealth,Label score)

{

this.Boundary = Boundary;

this.speed = speed;

offset = 20;

action = null;

pb = pb1;

gollem1 = gollem;

turret1 = turret;

spaceship1 = spaceship;

health1 = health;

gollem\_health = gollemhealth;

turret\_health = turrethealth;

spaceship\_health = spaceshiphealth;

Gollemfire = true;

Turretfire = true;

Gameover = false;

gollemlife = true;

turretlife = true;

spaceshiplife = true;

Score=score;

dscore = 0;

rng = new Random();

generate\_health();

}

public Player(int speed, Point Boundary, PictureBox player, PictureBox shooter, PictureBox boss,ProgressBar shooterhealth, ProgressBar bosshealth, ProgressBar playerhealth,Label score)

{

this.Boundary = Boundary;

this.speed = speed;

offset = 20;

action = null;

pb = player;

pshooter = shooter;

pboss = boss;

shooter\_health = shooterhealth;

boss\_health = bosshealth;

health1 = playerhealth;

Turretfire = true;

dronelife = true;

Gameover = false;

Score = score;

dscore = 0;

rng = new Random();

generate\_score();

}

public static void detectcollisionforstage2()

{

foreach (PictureBox upprojectile in upprojectiles)

{

if (upprojectile.Bounds.IntersectsWith(pboss.Bounds))

{

if (dscore <= 750)

{

dscore += 10;

}

upprojectile.Dispose();

if (boss\_health.Value > 0)

{

boss\_health.Value -= 5;

}

if (boss\_health.Value <= 0)

{

pboss.Dispose();

boss\_health.Dispose();

remove?.Invoke(pboss, EventArgs.Empty);

Gollemfire = false;

dronelife = false;

}

}

}

if (pb.Bounds.IntersectsWith(coin.Bounds))

{

if (dscore <= 750)

{

dscore += 10;

}

coin.Dispose();

remove?.Invoke(coin, EventArgs.Empty);

}

if (coin.Bounds.IntersectsWith(pshooter.Bounds))

{

coin.Dispose();

remove?.Invoke(coin, EventArgs.Empty);

}

if (pb.Bounds.IntersectsWith(pshooter.Bounds))

{

if (health1.Value > 0)

{

pb.Image = Properties.Resources.taking\_damage\_v3;

health1.Value -= 1;

}

}

foreach (PictureBox rightprojectile in rightprojectiles)

{

if (rightprojectile.Bounds.IntersectsWith(pshooter.Bounds))

{

if (dscore <= 750)

{

dscore += 10;

}

rightprojectile.Visible = false;

if (shooter\_health.Value > 0)

{

shooter\_health.Value -= 5;

}

if (shooter\_health.Value <= 0)

{

pshooter.Dispose();

shooter\_health.Dispose();

remove?.Invoke(pshooter, EventArgs.Empty);

Turretfire = false;

}

}

}

Score.Text = dscore.ToString();

if(health1.Value<=0)

{

Gameover = true;

}

}

public static void detectcollision()

{

foreach (PictureBox upprojectile in upprojectiles)

{

if (upprojectile.Bounds.IntersectsWith(gollem1.Bounds))

{

if (dscore <= 750)

{

dscore += 10;

}

upprojectile.Dispose();

if (gollem\_health.Value > 0)

{

gollem\_health.Value -= 5;

}

if (gollem\_health.Value <= 40)

{

gollem1.Image = Properties.Resources.gollemhurt\_\_2\_;

}

if (gollem\_health.Value <= 0)

{

gollem1.Dispose();

gollem\_health.Dispose();

remove?.Invoke(gollem1, EventArgs.Empty);

Gollemfire = false;

gollemlife = false;

}

}

if (upprojectile.Bounds.IntersectsWith(spaceship1.Bounds))

{

if (dscore <= 750)

{

dscore += 10;

}

upprojectile.Visible = false;

if (spaceship\_health.Value > 0)

{

spaceship\_health.Value -= 5;

}

if (spaceship\_health.Value <= 0)

{

spaceship1.Dispose();

spaceship\_health.Dispose();

remove?.Invoke(spaceship1, EventArgs.Empty);

spaceshiplife = false;

}

}

}

if (pb.Bounds.IntersectsWith(turret1.Bounds))

{

if (health1.Value > 0)

{

pb.Image = Properties.Resources.taking\_damage\_v3;

health1.Value -= 1;

}

}

if (pb.Bounds.IntersectsWith(health.Bounds))

{

if (health1.Value < 99)

{

health1.Value += 1;

}

health.Dispose();

remove?.Invoke(health, EventArgs.Empty);

}

if (health.Bounds.IntersectsWith(turret1.Bounds))

{

health.Dispose();

remove?.Invoke(health, EventArgs.Empty);

}

foreach (PictureBox leftprojectile in leftprojectiles)

{

if (leftprojectile.Bounds.IntersectsWith(turret1.Bounds))

{

if (dscore <= 750)

{

dscore += 10;

}

leftprojectile.Visible = false;

if (turret\_health.Value > 0)

{

turret\_health.Value -= 5;

}

if (turret\_health.Value <= 40)

{

turret1.Image = Properties.Resources.turretdmg;

}

if (turret\_health.Value <= 0)

{

turret1.Dispose();

turret\_health.Dispose();

remove?.Invoke(turret1, EventArgs.Empty);

Turretfire = false;

turretlife = false;

}

}

}

Score.Text = dscore.ToString();

if (health1.Value <= 0)

{

Gameover = true;

}

}

private void generate\_health()

{

health = new PictureBox();

Image img2 = Framework1.Properties.Resources.heart;

health.Image = img2;

health.Height = img2.Height;

health.Width = img2.Width;

health.BackColor = Color.Transparent;

health.Left = rng.Next(0, Boundary.X - img2.Width);

health.Top = pb.Top;

fire\_added?.Invoke(health,EventArgs.Empty);

}

private void generate\_score()

{

coin = new PictureBox();

Image img2 = Framework1.Properties.Resources.coin;

coin.Image = img2;

coin.Height = img2.Height;

coin.Width = img2.Width;

coin.BackColor = Color.Transparent;

coin.Left = rng.Next(0, Boundary.X - img2.Width);

coin.Top = pb.Top;

fire\_added?.Invoke(coin, EventArgs.Empty);

}

private void upperBullet()

{

upfire = new PictureBox();

Image img6 = Framework1.Properties.Resources.projectile\_up\_c;

upfire.Image = img6;

upfire.Height = img6.Height;

upfire.Width = img6.Width;

upfire.BackColor = Color.Transparent;

upfire.Left = pb.Left+42;

upfire.Top = pb.Top-20;

upprojectiles.Add(upfire);

fire\_added?.Invoke(upfire, EventArgs.Empty);

}

public static void moveupprojectile()

{

foreach (PictureBox upfire in upprojectiles)

{

fire\_added?.Invoke(upfire, EventArgs.Empty);

upfire.Top = upfire.Top - 15;

}

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

{

if (upprojectiles[i].Bottom <= 0)

{

upprojectiles.Remove(upprojectiles[i]);

}

}

}

private void leftbullet()

{

leftfire = new PictureBox();

Image img7 = Framework1.Properties.Resources.projectile\_left\_c;

leftfire.Image = img7;

leftfire.Height = img7.Height;

leftfire.Width = img7.Width;

leftfire.BackColor = Color.Transparent;

leftfire.Left = pb.Left + (-5);

leftfire.Top = pb.Top + 13;

leftprojectiles.Add(leftfire);

fire\_added?.Invoke(leftfire, EventArgs.Empty);

}

public static void moveleftprojectile()

{

foreach (PictureBox leftprojectile in leftprojectiles)

{

fire\_added?.Invoke(leftprojectile, EventArgs.Empty);

leftprojectile.Left = leftprojectile.Left - 15;

}

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

{

if (leftprojectiles[i].Right <= 0)

{

leftprojectiles.Remove(leftprojectiles[i]);

}

}

}

private void rightbullet()

{

rightfire = new PictureBox();

Image img8 = Framework1.Properties.Resources.projectile\_right\_v2\_c;

rightfire.Image = img8;

rightfire.Height = img8.Height;

rightfire.Width = img8.Width;

rightfire.BackColor = Color.Transparent;

rightfire.Left = pb.Left -(-5);

rightfire.Top = pb.Top + 13;

rightprojectiles.Add(rightfire);

fire\_added?.Invoke(rightfire, EventArgs.Empty);

}

public static void moverightprojectile()

{

foreach (PictureBox rightprojectile in rightprojectiles)

{

fire\_added?.Invoke(rightprojectile, EventArgs.Empty);

rightprojectile.Left = rightprojectile.Left + 15;

}

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

{

if (rightprojectiles[i].Left <= 0)

{

rightprojectiles.Remove(rightprojectiles[i]);

}

}

}

public void keypressedbyuser(Keys keycode)

{

if(keycode==Keys.Up)

{

action = "up";

}

else if (keycode == Keys.Down)

{

action = "down";

}

else if (keycode == Keys.Left)

{

action = "left";

}

else if (keycode == Keys.Right)

{

action = "right";

}

else if (keycode == Keys.W)

{

action = "fireup";

}

else if (keycode == Keys.A)

{

action = "fireleft";

}

else if (keycode == Keys.D)

{

action = "fireright";

}

else if (keycode == Keys.ControlKey)

{

action = "idle";

}

}

public Point move(Point location)

{

location.Y += 400;

if (action != null)

{

if (action == "left")

{

if (pb.Left > 0)

{

pb.Image = Properties.Resources.right\_run\_v3;

pb.Left -= speed;

health1.Left = pb.Left;

player\_added?.Invoke(upfire, EventArgs.Empty);

}

}

else if (action == "right")

{

if (pb.Left + pb.Width + offset < Boundary.X)

{

pb.Image = Properties.Resources.left\_run\_v3;

pb.Left += speed;

health1.Left = pb.Left;

player\_added?.Invoke(upfire, EventArgs.Empty);

}

}

else if (action == "fireup")

{

pb.Image = Properties.Resources.shoot\_up\_v3;

upperBullet();

}

else if (action == "fireleft")

{

pb.Image = Properties.Resources.right\_attack\_v3\_c;

leftbullet();

}

else if (action == "fireright")

{

pb.Image = Properties.Resources.left\_attack\_v3\_c;

rightbullet();

}

else if (action == "idle")

{

pb.Image = Properties.Resources.IDLE\_F;

rightbullet();

}

action = null;

}

return location;

}

}

}

**Game Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

using Framework1.Movement;

namespace Framework1.Core

{

public class Game

{

private List<Gameobject> assets;

private int overall\_gravity;

public event EventHandler obj\_added;

public Game(int overall\_gravity)

{

assets = new List<Gameobject>();

this.overall\_gravity = overall\_gravity;

}

public void add(Image img,int top,int left,Imove movement)

{

Gameobject g = new Gameobject(img, top, left,movement);

assets.Add(g);

obj\_added?.Invoke(g.Pb,EventArgs.Empty);

}

public void update()

{

foreach(Gameobject g in assets)

{

g.update();

}

}

public void fall()

{

foreach (Gameobject g in assets)

{

g.fall();

}

}

public void wave()

{

foreach (Gameobject g in assets)

{

g.wave();

}

}

public void fireright()

{

foreach (Gameobject g in assets)

{

g.fireright();

}

}

public void firedown()

{

foreach (Gameobject g in assets)

{

g.firedown();

}

}

public void fireup()

{

foreach (Gameobject g in assets)

{

g.fireup();

}

}

public void fireleft()

{

foreach (Gameobject g in assets)

{

g.fireleft ();

}

}

public void down(Keys keycode)

{

foreach(Gameobject g in assets)

{

if(g.Movement.GetType()==typeof(Player))

{

Player keyhandler = (Player)g.Movement;

keyhandler.keypressedbyuser(keycode);

}

}

}

**Gameobject Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

using Framework1.Movement;

namespace Framework1.Core

{

public class Gameobject

{

private PictureBox pb;

private Imove movement;

public Gameobject(Image img,int left,int top, Imove movement)

{

Pb = new PictureBox();

Pb.Image = img;

Pb.Height = img.Height;

Pb.Width = img.Width;

Pb.BackColor = Color.Transparent;

Pb.Top = top;

Pb.Left = left;

this.Movement = movement;

}

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

public Imove Movement { get => movement; set => movement = value; }

public void update()

{

Pb.Location = Movement.move(Pb.Location);

}

public Point givelocation()

{

return Pb.Location;

}

public void fall()

{

Pb.Location = Movement.move(Pb.Location);

}

public void wave()

{

Pb.Location = Movement.move(Pb.Location);

}

public void fireright()

{

Pb.Location = Movement.move(Pb.Location);

}

public void firedown()

{

Pb.Location = Movement.move(Pb.Location);

}

public void fireup()

{

Pb.Location = Movement.move(Pb.Location);

}

public void fireleft()

{

Pb.Location = Movement.move(Pb.Location);

}

}

}

**Firedown Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace Framework1.Movement

{

public class Firedown : Imove

{

private int speed;

private Point Boundary;

private int offset;

private static PictureBox pplayer, pdownprojectile, enemydownfire;

private static List<PictureBox> enemydownprojetiles = new List<PictureBox>();

private static ProgressBar health1;

public static event EventHandler fire\_added;

public Firedown(int speed, Point Boundary, PictureBox player, PictureBox downprojectile, ProgressBar health)

{

this.Boundary = Boundary;

this.speed = speed;

offset = 50;

pplayer = player;

pdownprojectile = downprojectile;

health1 = health;

generate\_enemybullet2();

}

public Point move(Point location)

{

enemydownfire.Location = location;

location.Y += 7;

return location;

}

public static void collisionofenemyprojectiles()

{

foreach (PictureBox enemydownfire in enemydownprojetiles)

{

if (enemydownfire.Bounds.IntersectsWith(pplayer.Bounds))

{

enemydownfire.Visible = false;

pdownprojectile.Visible = false;

if (health1.Value > 0)

{

pplayer.Image = Properties.Resources.taking\_damage\_v3;

health1.Value -= 1;

}

}

}

}

private void generate\_enemybullet2()

{

enemydownfire = new PictureBox();

Image img7 = Framework1.Properties.Resources.gollemprojectile;

enemydownfire.Image = img7;

enemydownfire.Height = img7.Height;

enemydownfire.Width = img7.Width;

enemydownfire.BackColor = Color.Transparent;

enemydownfire.Left = pdownprojectile.Left;

enemydownfire.Top = pdownprojectile.Top;

enemydownprojetiles.Add(enemydownfire);

fire\_added?.Invoke(enemydownfire, EventArgs.Empty);

}

public static void movegollemenemyprojectile()

{

// if (turretfire)

// {

foreach (PictureBox enemydownfire in enemydownprojetiles)

{

fire\_added?.Invoke(enemydownfire, EventArgs.Empty);

enemydownfire.Top += 7;

}

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

{

if (enemydownprojetiles[i].Bottom <= 0)

{

enemydownprojetiles.Remove(enemydownprojetiles[i]);

}

}

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

{

if (enemydownprojetiles[i].Visible == false)

{

enemydownprojetiles.Remove(enemydownprojetiles[i]);

}

}

//}

}

}

}

**Diagonal Class**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace Framework1.Movement

{

public class Diagonal:Imove

{

private int speed;

private Point Boundary;

private string direction;

private int offset;

private PictureBox enemydownfire, boss;

private static PictureBox pplayer;

private static List<PictureBox> enemydownprojetiles = new List<PictureBox>();

public static event EventHandler fire\_added;

private static ProgressBar health1;

public Diagonal(int speed, Point Boundary, string direction,PictureBox Boss,ProgressBar health)

{

this.Boundary = Boundary;

this.speed = speed;

this.direction = direction;

boss = Boss;

health1 = health;

offset = 130;

}

public Point move(Point location)

{

boss.Image = Framework1.Properties.Resources.drone;

boss.BackColor = Color.Transparent;

location.Y -= 100;

fire\_added?.Invoke(boss, EventArgs.Empty);

if (boss.Left <= 0)

{

direction = "right";

}

else if ((boss.Left + boss.Width) + 10 >= Boundary.X)

{

direction = "left";

}

if (direction == "left")

{

boss.Left -= speed;

boss.Top -= speed;

boss.Left -= speed;

boss.Left -= speed;

health1.Left-=speed;

health1.Top -= speed;

health1.Left -= speed;

health1.Left -= speed;

/\* location.X -= speed;

location.Y -= speed;

location.X -= speed;

location.X -= speed;\*/

}

else if (direction == "right")

{

boss.Left += speed;

boss.Top += speed;

boss.Left += speed;

boss.Left += speed;

health1.Left += speed;

health1.Top += speed;

health1.Left += speed;

health1.Left += speed;

}

return location;

}

}

}