**SOLDIER VS DRAGON FRAMEWORK**



Session: 2021 – 2025

**Submitted by:**

Ali Haider 2021-CS-38

**Supervised by:**

Sir. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Contents

[Description 3](#_Toc106622025)

[Game characters Description 3](#_Toc106622026)

[Rules and Interactions 3](#_Toc106622027)

[Goal of game 3](#_Toc106622028)

[Features of Framework 4](#_Toc106622029)

[Sequence diagram: 4](#_Toc106622031)

[Class Diagram: 5](#_Toc106622030)

[Example Usage of FrameWork:](#_Toc106622032) 9

[Complete Code 9](#_Toc106622034)

# 

# Description

In this game, there are mainly two main characters. One is Player (Soldier) and others are multiple enemies. Player will be controlled by keyboard while enemies will automatically move. Soldier task is to kill all enemies. He has a gun which he can use to kill enemies. There are many different platform in it. Player can climb up to platform by using ladders. Enemies will exhibit fire and if fire collides with player, its health will decrease. Player will win the game if he kills all the enemies and will lost the game if enemy kill him

# Game characters Description

* Soldier (Keyboard Movement)
* Enemies (Dragons)

# Rules and Interactions

* Soldier have to kill enemies
* Soldier has to avoid enemies fire
* If enemy fire collide with soldier, it health will decrease
* If soldier health become zero, he will lose the game
* If soldier fire collide with enemy, enemy health will decrease
* If all enemies are killed, soldier will win the game
* If enemy touches with soldier, soldier will die immediately
* Soldier can climb up to platform through ladder
* Soldier will fall down if he is in the air

# Goal of game

The goal of the game is to kill all the enemies and avoid touching with enemy and with the enemy fire.

# 

# Features of Framework

Following are the feature of framework:

* Movement Framework
* Collision Framework
* Firing Framework

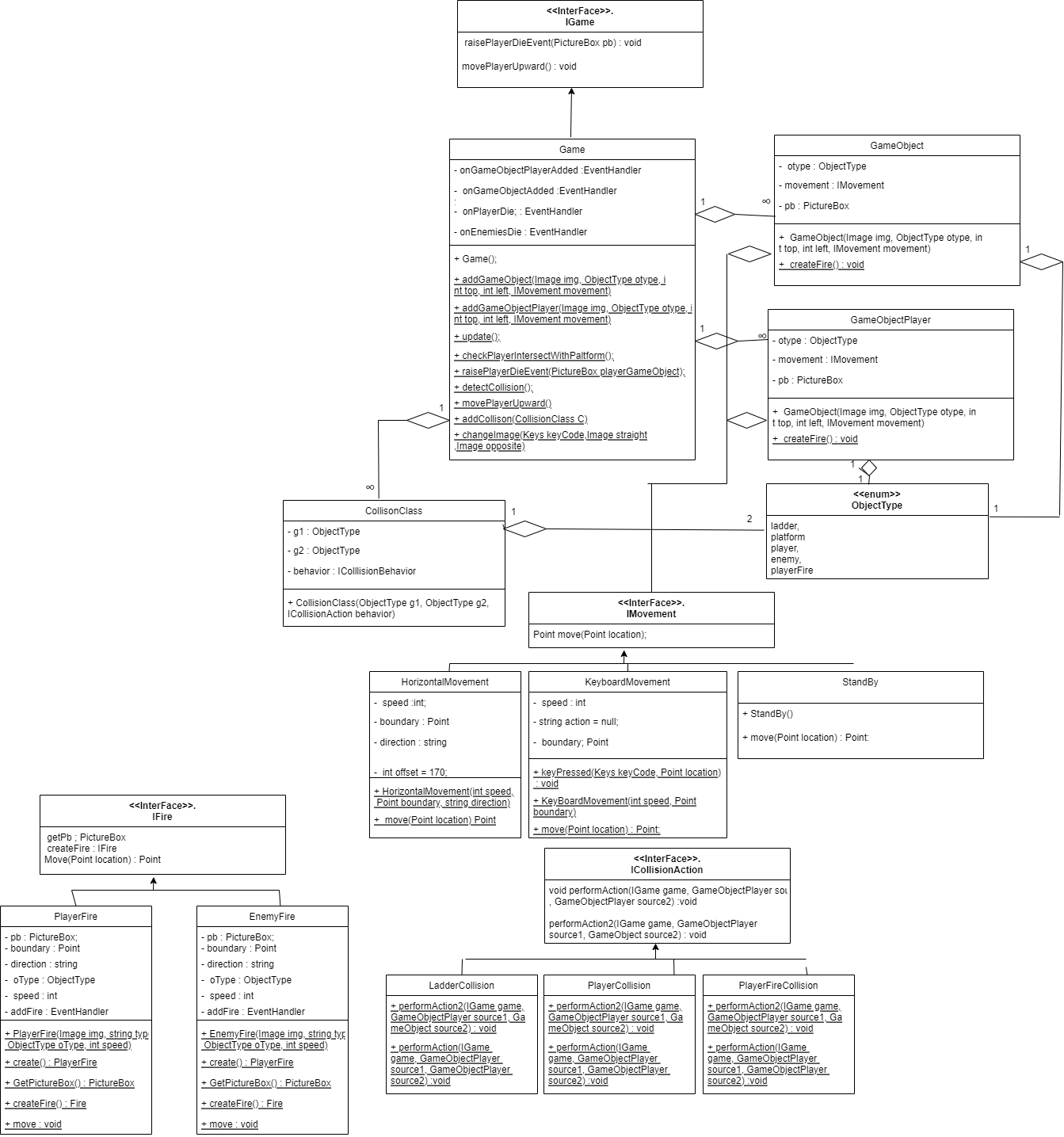
They have also the ability to get extended.

# Sequence diagram:

Diagram

Description automatically generated

**Class Diagram:**

****

# Example Usage of Framework:

**First Include these NameSpaces:**

GameFrameWork.Core;

GameFrameWorkk.Collision;

GameFrameWorkk.Core.Fire;

GameFrameWorkk.Core.Movement;

**Create Object Of Game Class:**

Game go = new Game();

**On Form Load Event initialize Following Events:**

Add This Code to Open Window in full Screen for better Usage

//

this.TopMost = true;

this.FormBorderStyle = FormBorderStyle.None;

this.WindowState = FormWindowState.Maximized; //

**then these for Fires**

PlayerFire firePlayer = new PlayerFire(Properties.Resources.soldierFire2, "player", ObjectType.playerFire, 10);

PlayerFire fireEnemy = new PlayerFire(Properties.Resources.soldierFire2, "enemy", ObjectType.playerFire, 10);

**Then Event**

go.onGameObjectAdded += new EventHandler(addIntoControls);

go.onPlayerDie += new EventHandler(removePlayer)

go.onEnemyDie+=new EventHandler(removeEnemy)

**Add Player and Enemies**

go.addGameObjectPlayer(Properties.Resources.newStandStraight, ObjectType.player, 700, 20, new KeyBoardMovement(20, boundary), firePlayer);

go.addGameObjectPlayer(Properties.Resources.NewPurpleDragon,ObjectType.enemy,100, 200, new HorizontalMovement(10, boundary, "right"),fireEnemy)

**Adding Platform and Ladders**

go.addGameObject(Properties.Resources.longPlatform, ObjectType.platform, 550, 290, new StandBy());

go.addGameObject(Properties.Resources.longPlatform, ObjectType.platform, 250, 640, new StandBy());

go.addGameObject(Properties.Resources.image, ObjectType.platform, 550, 1210, new StandBy());

go.addGameObject(Properties.Resources.image, ObjectType.platform, 330, 90, new StandBy());

go.addGameObject(Properties.Resources.smalladder, ObjectType.ladder, 600, 1370, new StandBy());

go.addGameObject(Properties.Resources.ladder3, ObjectType.ladder, 250, 1450, new StandBy());

go.addGameObject(Properties.Resources.smalladder, ObjectType.ladder, 350, 350, new StandBy());

**Also add Collisions as:**

CollisionClass C = new CollisionClass(ObjectType.player, ObjectType.Enemy, new PlayerCollison());

CollisionClass C2 = new CollisionClass(ObjectType.player, ObjectType.ladder, new ladderCollison());

//Adding to List

g.addCollision(C);

g.addCollision(C2);

**Implement onGameObjectAdded\_Game function as:**

private void addIntoControls(object sender, EventArgs e)

{

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

}

**Implement removePlayer function as:**

private void removePlayer(object sender , EventArgs e)

{

GameObject go = (GameObject)sender;

if (go.Healthbar.Value > 0)

{

go.Healthbar.Value--;

}

else

{

this.Controls.Remove(go.Healthbar);

this.Controls.Remove(go.Pb);

this.Close();

}

}

**Implement removeEnemy function as:**

GameObject go = (GameObject)sender;

if(go.Healthbar.Value > 0)

{

go.Healthbar.Value--;

}

else

{

go.Otype1 = ObjectType.Dead;

g.addGameObjects(Properties.Resources.snowBallgif,go.Pb.Left, go.Pb.Top, ObjectType.SnowBall,"SnowBall", new SnowBallMovement(5, new Point(this.Width, this.Height) , "left"));

this.Controls.Remove(go.Healthbar);

this.Controls.Remove(go.Pb);

}

**Add Timer and Implement TimerTick event as:**

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

{

g.update();

g.collideWithPlatform();

g.EnableEnemyFire("Enemy2", Properties.Resources.enemyFireLeft, Properties.Resources.enemyFireLeft, new FireMovement("left", 7));

g.EnableEnemyFire("Enemy1", Properties.Resources.enemyFireLeft , Properties.Resources.enemyFireLeft , new FireMovement("right", 7));

g.EnableEnemyFire("Enemy1", Properties.Resources.enemyFireLeft, Properties.Resources.enemyFireLeft, new FireMovement("left" , 7));

}

**Implement KeyDown Event as:**

g.keyPressed(e.KeyCode);

g.EnablePlayerFire(e.KeyCode, Properties.Resources.fireLeft);

# Explanation of FrameWork:

You have to create gameobject and gameObjectPlayers by passing all parameter and an object of movement class which will be derived from IMovement Interface;

Object will start moving according to given Movement.

If object is created and it is keyboard movement then player will move with keyboard arrows and space to fire

Enemies will move automatically

For Collision and their Action you have to create object of CollisionClass by passing two objects and their Action. Then add it into Collision List

# Complete Code

**Form :**

public partial class Form1 : Form

{

Game go;

public Form1()

{

InitializeComponent();

}

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

{

//Maximizing the Window

/\* this.TopMost = true;

this.FormBorderStyle = FormBorderStyle.None;

this.WindowState = FormWindowState.Maximized;\*/

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

go = new Game();

PlayerFire firePlayer = new PlayerFire(Properties.Resources.soldierFire2, "player", ObjectType.playerFire, 10);

PlayerFire fireEnemy = new PlayerFire(Properties.Resources.soldierFire2, "enemy", ObjectType.playerFire, 10);

go.onGameObjectAdded += new EventHandler(addIntoControls);

go.onPlayerDie += new EventHandler(removePlayer);

go.OnProgressBarAdded += Game\_OnProgressBarAdd;

go.onGameObjectPlayerAdded += new EventHandler(addIntoControls);

go.onSpacePressed += new EventHandler(addIntoControls);

go.addGameObject(Properties.Resources.longPlatform, ObjectType.platform, 550, 290, new StandBy());

go.addGameObject(Properties.Resources.longPlatform, ObjectType.platform, 250, 640, new StandBy());

go.addGameObject(Properties.Resources.image, ObjectType.platform, 550, 1210, new StandBy());

go.addGameObject(Properties.Resources.image, ObjectType.platform, 330, 90, new StandBy());

go.addGameObject(Properties.Resources.smalladder, ObjectType.ladder, 600, 1370, new StandBy());

go.addGameObject(Properties.Resources.ladder3, ObjectType.ladder, 250, 1450, new StandBy());

go.addGameObject(Properties.Resources.smalladder, ObjectType.ladder, 350, 350, new StandBy());

go.addGameObjectPlayer(Properties.Resources.newStandStraight, ObjectType.player, 700, 20, new KeyBoardMovement(20, boundary), firePlayer);

go.addGameObjectPlayer(Properties.Resources.NewPurpleDragon,ObjectType.enemy,100, 200, new HorizontalMovement(10, boundary, "right"),fireEnemy);

CollisionClass C = new CollisionClass(ObjectType.player, ObjectType.enemy, new PlayerCollision());

CollisionClass C2 = new CollisionClass(ObjectType.player, ObjectType.ladder, new LadderCollision());

go.addCollison(C);

go.addCollison(C2);

}

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

{

go.update();

/\*go.checkPlayerIntersectWithPaltform();\*/

}

private void addIntoControls(object sender, EventArgs e)

{

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

}

private void Game\_OnProgressBarAdd(object sender, EventArgs e)

{

ProgressBar bar = (ProgressBar)sender;

this.Controls.Add(bar);

}

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

{

if (e.KeyCode == Keys.Escape)

{

this.Close();

}

go.keyPressed(e.KeyCode);

go.makeFire(e.KeyCode,Properties.Resources.soldierFire2);

go.changeImage(e.KeyCode, Properties.Resources.newStandStraight, Properties.Resources.NewOppositStraight);

}

private void removePlayer(object sender, EventArgs e)

{

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

}

private void Player\_OnLeftMove(object sender, EventArgs e)

{

GameObjectPlayer obj = (GameObjectPlayer)sender;

obj.changeImage(Properties.Resources.NewOppositStraight);

}

private void Player\_OnRightMove(object sender, EventArgs e)

{

GameObjectPlayer obj = (GameObjectPlayer)sender;

obj.changeImage(Properties.Resources.newStandStraight);

}

**Core Classes:**

public interface IGame

{

void RaisePlayerDieEvent(GameObject go);

void RaiseEnemyDieEvent(GameObject go);

}

public class Game : IGame

{

private List<GameObjectPlayer> gameObjectPlayerList;

private List<GameObject> gameObjectList;

public event EventHandler onGameObjectAdded;

public event EventHandler onGameObjectPlayerAdded;

private List<CollisionClass> collisions;

public event EventHandler onPlayerDie;

public event EventHandler onSpacePressed;

private EventHandler onProgressBarAdded;

private List<PictureBox> PlayerFireList = new List<PictureBox>();

EventHandler onLeftMove;

EventHandler onRightMove;

public EventHandler OnLeftMove { get => onLeftMove; set => onLeftMove = value; }

public EventHandler OnRightMove { get => onRightMove; set => onRightMove = value; }

public EventHandler OnProgressBarAdded { get => onProgressBarAdded; set => onProgressBarAdded = value; }

public Game()

{

gameObjectList = new List<GameObject>();

gameObjectPlayerList = new List<GameObjectPlayer>();

collisions = new List<CollisionClass>();

}

public void addGameObject(Image img, ObjectType otype, int top, int left, IMovement movement)

{

GameObject go = new GameObject(img, otype, top, left, movement);

gameObjectList.Add(go);

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

}

public void addGameObjectPlayer(Image img, ObjectType otype, int top, int left, IMovement movement,IFire fire)

{

GameObjectPlayer go = new GameObjectPlayer(img, otype, top, left, movement,fire);

gameObjectPlayerList.Add(go);

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

}

public void update()

{

detectCollision();

foreach (GameObjectPlayer go in gameObjectPlayerList)

{

go.gameObjectMove();

go.update2();

}

}

public void keyPressed(Keys keyCode)

{

foreach (var go in gameObjectPlayerList)

{

if (go.Movement.GetType() == typeof(KeyBoardMovement))

{

KeyBoardMovement keyBoardHandler = (KeyBoardMovement)go.Movement;

keyBoardHandler.keyPressed(keyCode, go.Pb.Location);

}

}

}

public void checkPlayerIntersectWithPaltform()

{

foreach (var go in gameObjectPlayerList)

{

if (go.Movement.GetType() == typeof(KeyBoardMovement))

{

foreach (var coll in gameObjectList)

{

if (go.Pb.Bounds.IntersectsWith(coll.Pb.Bounds))

{

if (coll.Movement.GetType() == typeof(StandBy))

{

go.Pb.Top = coll.Pb.Top - (go.Pb.Height+5);

}

}

}

}

}

}

public void raisePlayerDieEvent(PictureBox playerGameObject)

{

onPlayerDie?.Invoke(playerGameObject, EventArgs.Empty);

}

public void detectCollision()

{

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

{

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

{

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

{

foreach (var c in collisions)

{

if(c.G2==ObjectType.ladder)

{

checkPlayerIntersectWithPaltform();

}

if (c.G1 == gameObjectPlayerList[i].Otype && gameObjectPlayerList[j].Otype == c.G2)

{

c.Behavior.performAction(this, gameObjectPlayerList[i], gameObjectPlayerList[j]);

}

}

}

}

}

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

{

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

{

if (gameObjectPlayerList[i].Pb.Bounds.IntersectsWith(gameObjectList[j].Pb.Bounds))

{

foreach (var c in collisions)

{

if (c.G1 == gameObjectPlayerList[i].Otype && gameObjectList[j].Otype == c.G2)

{

c.Behavior.performAction2(this, gameObjectPlayerList[i], gameObjectList[j]);

}

}

}

}

}

}

public void movePlayerUpward()

{

checkPlayerIntersectWithPaltform();

}

public void addCollison(CollisionClass C)

{

collisions.Add(C);

}

public void changeImage(Keys keyCode,Image straight,Image opposite)

{

GameObjectPlayer soldier = gameObjectPlayerList.Find(o => o.Otype == ObjectType.player);

if(keyCode==Keys.Left)

{

soldier.Pb.Image = opposite;

soldier.Pb.Tag = "oppo";

}

else if(keyCode==Keys.Right)

{

soldier.Pb.Image = straight;

soldier.Pb.Tag = "straight";

}

}

public void makeFire(Keys keyCode,Image image)

{

GameObjectPlayer soldier = gameObjectPlayerList.Find(o => o.Otype == ObjectType.player);

if(keyCode==Keys.Space)

{

PictureBox fireImage = new PictureBox();

fireImage.Image = image;

fireImage.Width = image.Width;

fireImage.Height = image.Height;

fireImage.BackColor = Color.Transparent;

if((string)soldier.Pb.Tag=="straight")

{

fireImage.Left += 10;

fireImage.Tag = "Left";

fireImage.Left = (soldier.Pb.Left +soldier.Pb.Width)+ 5;

}

if ((string)soldier.Pb.Tag == "oppo")

{

fireImage.Left -= 10;

fireImage.Tag = "Right";

fireImage.Left = soldier.Pb.Left - 40;

}

fireImage.Top = soldier.Pb.Top + 37;

PlayerFireList.Add(fireImage);

onSpacePressed?.Invoke(fireImage, EventArgs.Empty);

}

}

private void player\_OnProgressBarAdd(object sender, EventArgs e)

{

OnProgressBarAdded?.Invoke(sender, EventArgs.Empty);

}

public class GameObject

{

private PictureBox pb;

private IMovement movement;

private ObjectType otype;

public GameObject(Image img, ObjectType otype, int top, int left, IMovement movement)

{

Pb = new PictureBox();

Pb.Image = img;

Pb.Width = img.Width;

Pb.Height = img.Height;

Pb.Left = left;

Pb.Tag = "platform";

Pb.Top = top;

Pb.BackColor = Color.Transparent;

this.Movement = movement;

this.Otype = otype;

}

public virtual void update()

{

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

}

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

public ObjectType Otype { get => otype; set => otype = value; }

internal IMovement Movement { get => movement; set => movement = value; }

}

public class GameObjectPlayer

{

private PictureBox pb;

private IMovement movement;

private ObjectType otype;

private IFire fire;

private ProgressBar healthBar;

private List<IFire> fireList;

private EventHandler fireAdded;

private EventHandler onhealthProgressBarAdded;

public GameObjectPlayer(Image img, ObjectType otype, int top, int left, IMovement movement, IFire fire)

{

Pb = new PictureBox();

Pb.Image = img;

Pb.Width = img.Width;

Pb.Height = img.Height;

Pb.Left = left;

pb.BackColor = Color.Transparent;

this.Fire = fire;

this.movement = movement;

FireList = new List<IFire>();

Pb.Top = top;

Pb.BackColor = Color.Transparent;

this.Movement = movement;

this.Otype = otype;

makeHealthBar();

}

public void gameObjectMove()

{

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

}

public void makeHealthBar()

{

healthBar = new ProgressBar();

this.healthBar.Location = new Point(Pb.Left, Pb.Top - 20);

this.healthBar.Name = "progressBar1";

this.healthBar.Size = new System.Drawing.Size(89, 16);

this.healthBar.Value = 100;

this.healthBar.BringToFront();

onHealthProgressBarAdded?.Invoke(healthBar, new EventArgs());

}

public void changeImage(Image img)

{

Pb.Image = img;

Pb.Width = img.Width;

Pb.Height = img.Height;

}

public void createFire()

{

IFire f = fire.createFire(new Point(Pb.Left, Pb.Top));

if (f != null)

{

fireList.Add(f);

FireAdded?.Invoke(f.GetPictureBox(), new EventArgs());

}

}

public void moveFires()

{

foreach (IFire f in fireList)

{

f.GetPictureBox().Location = f.move();

}

}

public void moveHealthBar()

{

healthBar.Left = Pb.Left;

healthBar.Top = Pb.Top - 20;

}

public void update2()

{

moveFires();

moveHealthBar();

}

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

public ObjectType Otype { get => otype; set => otype = value; }

internal IMovement Movement { get => movement; set => movement = value; }

public EventHandler FireAdded { get => fireAdded; set => fireAdded = value; }

public EventHandler onHealthProgressBarAdded { get => onhealthProgressBarAdded; set => onhealthProgressBarAdded = value; }

public ProgressBar HealthBar { get => healthBar; set => healthBar = value; }

public List<IFire> FireList { get => fireList; set => fireList = value; }

public IFire Fire { get => fire; set => fire = value; }

}

public interface IGame

{

void raisePlayerDieEvent(PictureBox pb);

void movePlayerUpward();

}

public enum ObjectType

{

player,

enemy,

platform,

ladder,

playerFire

}

**Movement Classes:**

public interface IMovement

{

Point move(Point Location);

}

public class HorizontalMovement : IMovement

{

private int speed;

private Point boundary;

private string direction;

private int offset = 170;

public HorizontalMovement(int speed, Point boundary, string direction)

{

this.speed = speed;

this.boundary = boundary;

this.direction = direction;

}

public Point move(Point location)

{

if (location.X < 0)

{

direction = "right";

}

else if (location.X + offset >= boundary.X)

{

direction = "left";

}

if (direction == "left")

{

location.X -= speed;

}

else if (direction == "right")

{

location.X += speed;

}

return location;

}

}

public class KeyBoardMovement : IMovement

{

private int speed;

private Point boundary;

private string action = null;

public string Action { get => action; set => action = value; }

public KeyBoardMovement(int speed, Point boundary)

{

this.speed = speed;

this.boundary = boundary;

}

public KeyBoardMovement()

{

}

public void keyPressed(Keys keyCode, Point location)

{

if (keyCode == Keys.Up)

{

Action = "up";

}

else if (keyCode == Keys.Down && location.Y < 700)

{

Action = "down";

}

else if (keyCode == Keys.Left && location.X > 0)

{

Action = "left";

}

else if (keyCode == Keys.Right && location.X + 80 < boundary.X)

{

Action = "right";

}

}

public Point move(Point location)

{

if (Action != null)

{

if (Action == "up")

{

/\*location.Y -= speed;\*/

}

else if (Action == "down")

{

location.Y += speed;

}

else if (Action == "left")

{

location.X -= speed;

}

else if (Action == "right")

{

location.X += speed;

}

}

if (location.Y < 700)

{

location.Y = location.Y + (speed - 9);

}

Action = null;

return location;

}

}

}

}

public class StandBy : IMovement

{

public StandBy()

{

}

public Point move(Point location)

{

return location;

}

}

}

**Collision Classes:**

public interface ICollisionAction

{

void performAction(IGame game, GameObject source1, GameObject source2);

void performAction2(IGame game, GameObjectPlayer source1, GameObject source2);

}

public class CollisionClass

{

private ObjectType g1;

private ObjectType g2;

private ICollisionAction behavior;

public CollisionClass(ObjectType g1, ObjectType g2, ICollisionAction behavior)

{

this.G1 = g1;

this.G2 = g2;

this.Behavior = behavior;

}

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

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

public ICollisionAction Behavior { get => behavior; set => behavior = value; }

public class LadderCollision : ICollisionAction

{

public void performAction2(IGame game, GameObjectPlayer source1, GameObject source2)

{

GameObjectPlayer player = source1;

GameObject hurdle = source2;

game.movePlayerUpward();

}

public void performAction(IGame game, GameObjectPlayer source1, GameObjectPlayer source2)

{

}

}

public class PlayerCollision : ICollisionAction

{

public void performAction(IGame game, GameObjectPlayer source1, GameObjectPlayer source2)

{

GameObjectPlayer player;

if (source1.Otype == ObjectType.player)

{

player = source1;

}

else

{

player = source2;

}

game.raisePlayerDieEvent(player.Pb);

}

public void performAction2(IGame game, GameObjectPlayer source1, GameObject source2)

{

}

**Fire Classes:**

public interface IFire

{

PictureBox GetPictureBox();

IFire createFire(Point location);

Point move();

}

public class PlayerFire : IFire

{

private PictureBox pb;

private ObjectType oType;

private string direction = "left";

private int speed;

public event EventHandler addFire;

public PlayerFire(Image img, string type, ObjectType oType, int speed)

{

Pb = new PictureBox();

Pb.Image = img;

this.Speed = speed;

Pb.BackColor = Color.Transparent;

Pb.Tag = type;

Pb.Width = img.Width;

Pb.Height = img.Height;

this.OType = oType;

}

public PlayerFire(PlayerFire fire)

{

this.pb = fire.Pb;

this.speed = fire.Speed;

this.OType = fire.OType;

}

public PlayerFire create()

{

PlayerFire F = new PlayerFire(Pb.Image, "player", ObjectType.playerFire, 10);

return F;

}

public IFire createFire()

{

if (Keyboard.IsKeyPressed(Key.Space))

{

PlayerFire fire = create();

addFire?.Invoke(Pb, EventArgs.Empty);

return fire;

}

return null;

}

public PictureBox GetPictureBox()

{

return pb;

}

public IFire createFire(System.Drawing.Point location)

{

if (EZInput.Keyboard.IsKeyPressed(Key.Space))

{

if (direction == "left")

{

makeFire(moveDirection.left, location);

return new PlayerFire(this);

}

if (direction == "right")

{

makeFire(moveDirection.right, location);

return new PlayerFire(this);

}

}

return null;

}

private void makeFire(moveDirection direction, System.Drawing.Point location)

{

pb = new PictureBox();

pb.Left = location.X;

pb.Top = location.Y + 30;

pb.Image = pb.Image;

pb.Height = pb.Image.Height;

pb.Width = pb.Image.Width;

pb.BackColor = Color.Transparent;

pb.BringToFront();

}

public System.Drawing.Point move()

{

if (direction == "left")

{

pb.Left -= speed;

}

else if (direction == "right")

{

pb.Left += speed;

}

return new System.Drawing.Point(pb.Left, pb.Top);

}

public System.Drawing.Point move(System.Drawing.Point location)

{

location.X += speed;

return location;

}

public void update()

{

createFire();

}

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

public int Speed { get => speed; set => speed = value; }

public ObjectType OType { get => oType; set => oType = value; }

}

enum moveDirection

{

right,

left

}