**Tank Fight Game**

****

**Session: 2022-2026**

**Submitted by:**

Rida Batool

2022-CS-118

**Supervised by:**

**Sir Awais**

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Contents**

[**Short Description and Story Writing of Game** 4](#_Toc141354658)

[**Game Characters Description** 4](#_Toc141354659)

[**Player:** 4](#_Toc141354660)

[**Enemies:** 4](#_Toc141354661)

[**Random Enemy:** 4](#_Toc141354662)

[**Vertical Enemy:** 5](#_Toc141354663)

[**Horizontal Enemy:** 5](#_Toc141354664)

[**Game object description** 5](#_Toc141354665)

[**Walls:** 5](#_Toc141354666)

[**Rules and Interaction (Game Play):** 5](#_Toc141354667)

[**Goal of the game:** 5](#_Toc141354668)

[**Wireframes:** 6](#_Toc141354669)

[**Class Details:** 7](#_Toc141354670)

[**Class Diagram:** 8](#_Toc141354671)

[**Code:** 8](#_Toc141354672)

[**Classes:** 8](#_Toc141354673)

[**Game Class:** 8](#_Toc141354674)

[**GameCell Class:** 10](#_Toc141354675)

[**GameGrid Class:** 16](#_Toc141354676)

[**GameObject Class:** 18](#_Toc141354677)

[**GameObjectType:** 20](#_Toc141354678)

[**GameDirection:** 20](#_Toc141354679)

[**GamePlayer Class:** 21](#_Toc141354680)

[**Enemy Class:** 22](#_Toc141354681)

[**EnemyVertical Class:** 22](#_Toc141354682)

[**EnemyHorizontal Class:** 24](#_Toc141354683)

[**EnemyRandom Class:** 26](#_Toc141354684)

[**Bullet Class:** 28](#_Toc141354685)

[**EnemyBullet Class:** 29](#_Toc141354686)

[**Collision Class:** 30](#_Toc141354687)

[**Forms:** 30](#_Toc141354688)

[**Start Form:** 30](#_Toc141354689)

[**Win Form:** 31](#_Toc141354690)

[**Lose Form:** 32](#_Toc141354691)

[**Main Form:** 33](#_Toc141354692)

[**Conclusion:** 43](#_Toc141354693)

**Tank Fight Game**

# **YouTube Video Link:**

* <https://www.youtube.com/watch?v=C7R6Cc2yl9U>

# **Short Description and Story Writing of Game**

Tank Fight is a console-based action-packed single-player game where players control the powerful tank. Navigate through challenging tracks and face off against three enemy tanks: vertical, horizontal and random; in fast-paced battles. Use strategy and quick reflexes to defeat your opponents and become the ultimate tank-fighting champion!

With smooth controls, and pulse-pounding action, Tank Fight is the perfect game for console gamers who love excitement and intense single-player challenges. Put your skills to the test as you face off against three formidable enemy tanks, relying only on your trusty bullets to come out on top. Can you emerge victorious against all the enemies in this high-stakes, bullet-riddled battle? Find out in Tank Fight!

# **Game Characters Description**

## **Player:**

The main player tank in the console-based game Tank Fight is equipped with only bullets, having a total of three lives and controlled by the player to defeat three enemy tanks and become the champion. Player is the key to victory, providing an intense and exciting gaming experience.

## **Enemies:**

There are three enemies in game.

### **Random Enemy:**

It is one of the three tricky enemy tanks in the console-based game Tank Fight. With unpredictable and random movements, this adversary keeps the player on their toes. Player must use his driving skills and weapons to anticipate its unpredictable actions and defeat this opponent to claim victory in this battle.

### **Vertical Enemy:**

It is one of the three enemy tanks in the game Tank Fight. Moving vertically and firing, it presents a challenge to the player, who has only a set of bullets as his weapon. Player must use its aiming skills to defeat this enemy and emerge victorious in this battle.

### **Horizontal Enemy:**

It is one of the three enemy tanks in the game Tank Fight. It is known for its firing techniques. This makes this enemy a tricky opponent for the player. The player must use strategy and quick reflexes to defeat it and become the ultimate tank fighting champion.

# **Game object description**

Following are objects in the game:

### **Walls:**

Walls are the barriers in the game which the Player and the enemy tanks cannot cross.

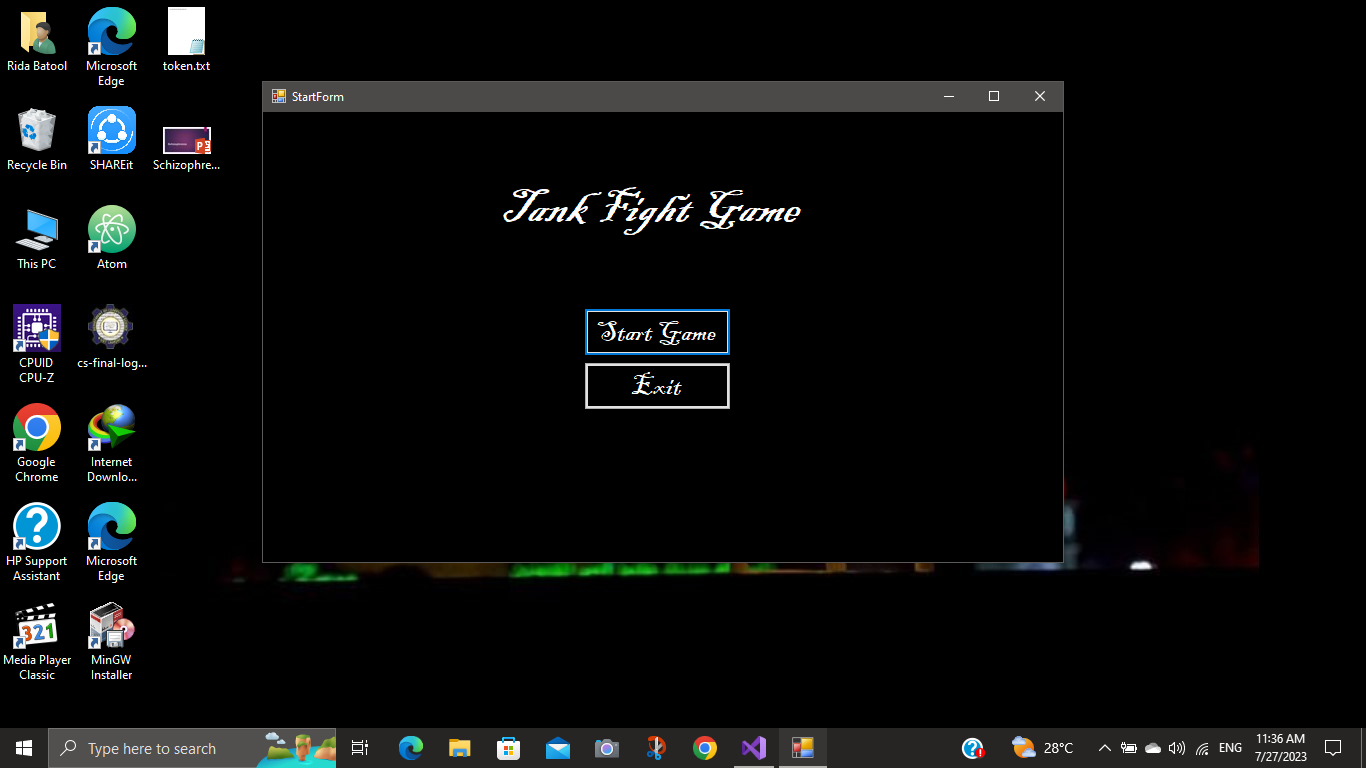
# **Rules and Interaction (Game Play):**

The gameplay involves using player’s bullets to defeat enemies, and firing by aiming at the enemy tanks and avoiding enemy attacks. Player loses a life if he collides with any of the three enemies. If player collides with the bullets coming from the enemy tanks, he loses a score.

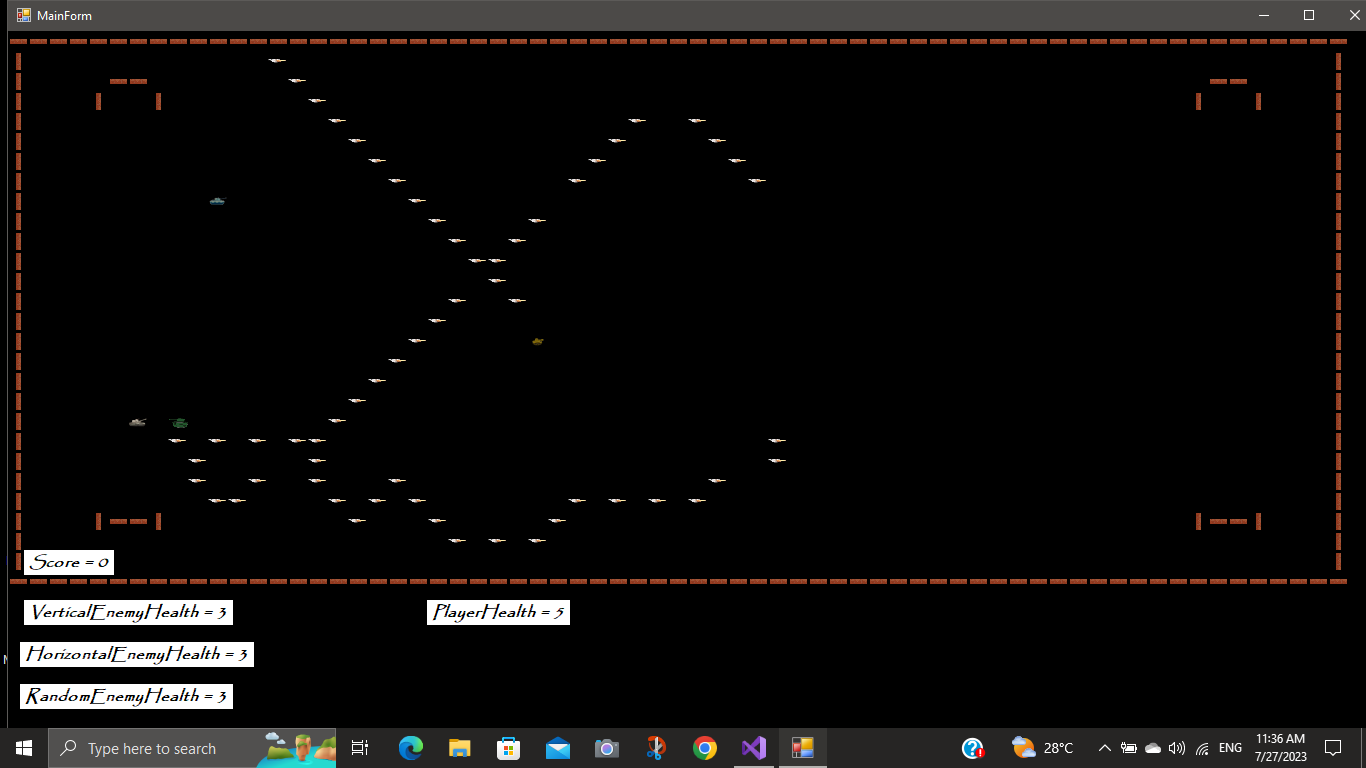
# **Goal of the game:**

The goal of the game Tank Fight is for the player to defeat all three enemy tanks, Vertical, Horizontal and Random tank, and become the champion tank fighter. The player must use his bullets to defeat the enemies and avoid enemy attacks. The game's difficulty increases as the player progress and the player's score are tracked throughout the game. The ultimate objective is to destroy all the three enemy tanks and become the ultimate tank fighting champion.

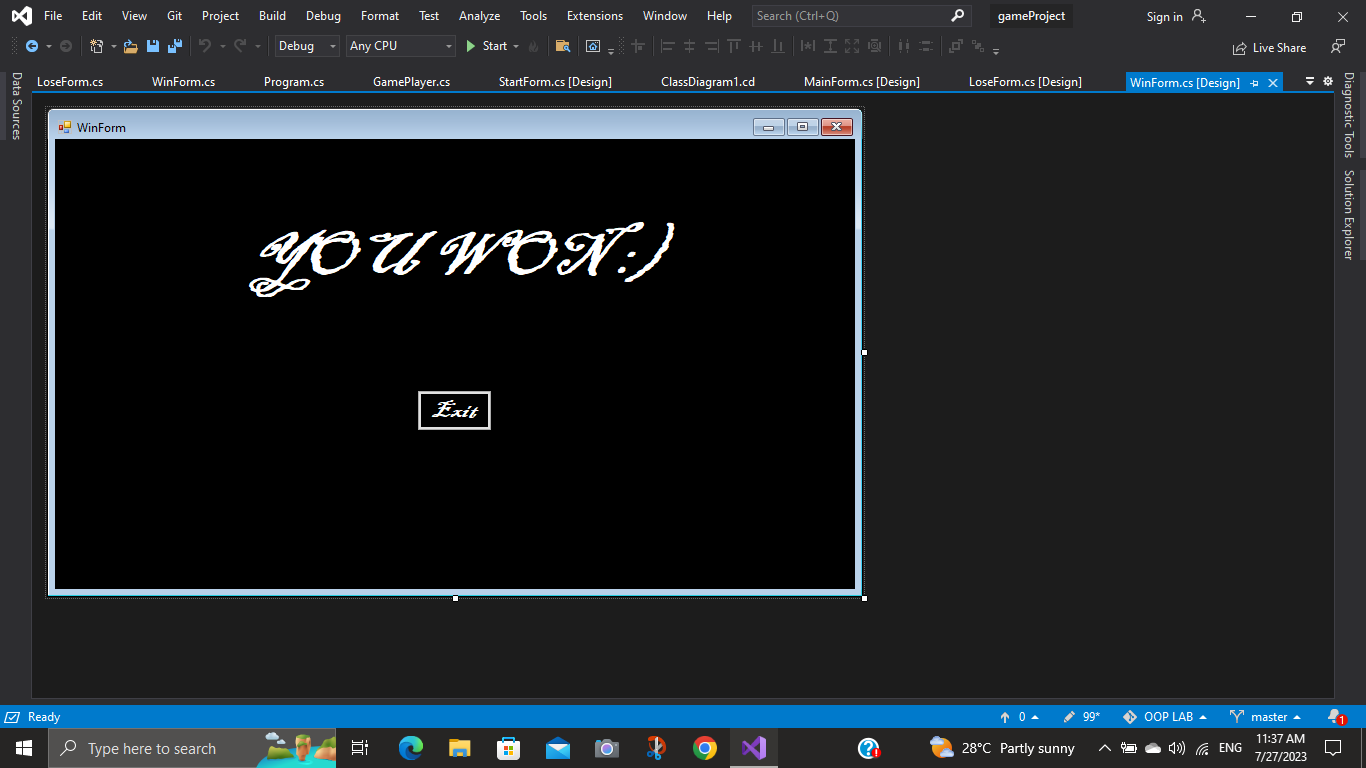
# **Wireframes:**

****

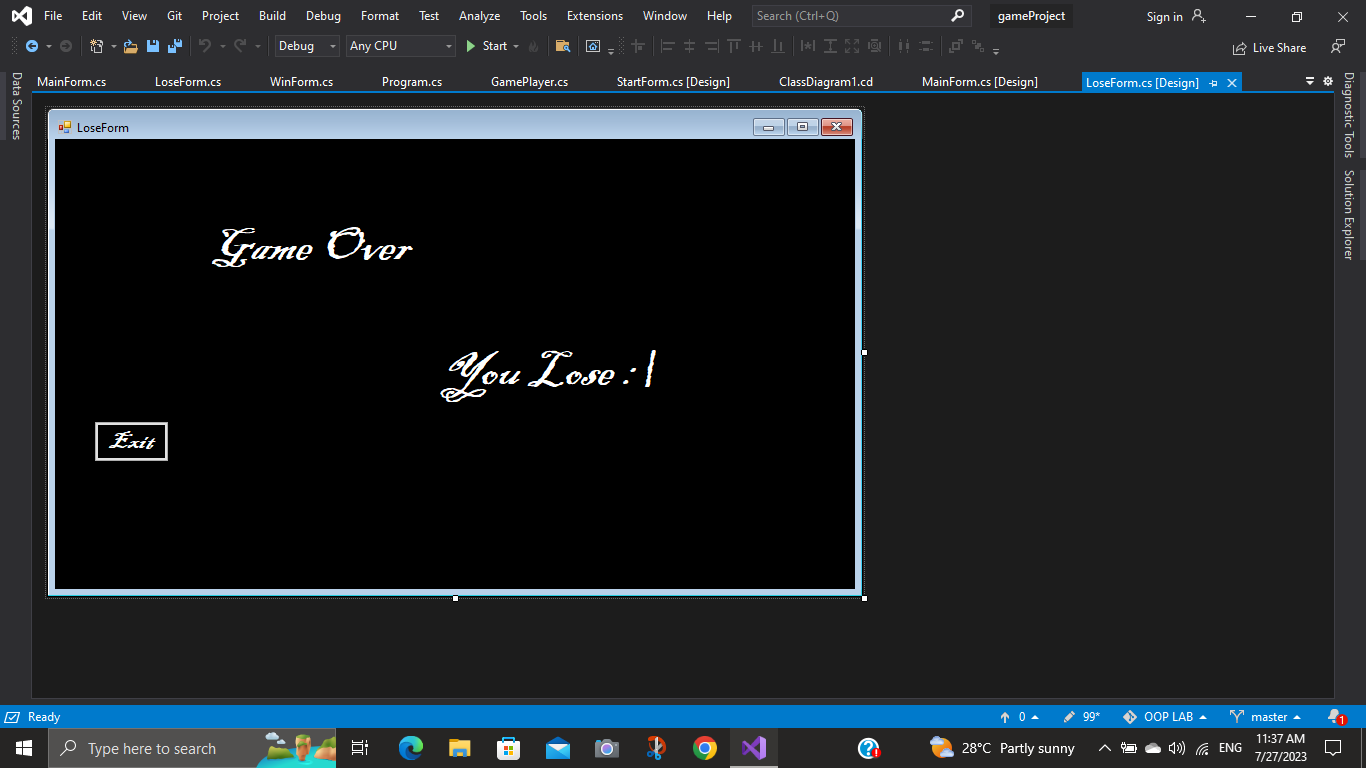
**Figure 1: Start Game**



**Figure 2: Main Form**

****

**Figure 3: Winning Form**

****

**Figure 4: Losing Form**

# **Class Details:**

The GameObject Class is a parent class that inherits GamePlayer Class, Enemy Class, Bullet Class and EnemyBullet Class. Enemy Class is abstracted that is parent class for VerticalEnemy class, HorizontalEnemy class and Random Enemy Class.

Game class is associated with GameGrid class which is associated with GameCell class and GameCell class is further associated with GameObject class.

Collision class is responsible for checking the collision between player, enemies, bullets and walls.

GameObjectType and GameDirection are enum classes.

# **Class Diagram:**

Below is given the Class Diagram that helps understanding all the relations;

# **Code:**

**Figure 5: Class Diagram**

## **Classes:**

### **Game Class:**

public class Game

{

public GameGrid GameGrid

{

get => default;

set

{

}

}

public static GameObject getBlankObject()

{

GameObject blankObject = new GameObject(GameObjectType.NONE, Properties.Resources.simplebox);

return blankObject;

}

public static Image getCharacterImage(char displayCharacter)

{

Image img = Properties.Resources.simplebox;

if (displayCharacter == '|' || displayCharacter == '?')

{

img = Properties.Resources.brickVertical;

}

else if (displayCharacter == '!' || displayCharacter == '\_')

{

img = Properties.Resources.brickHorizontal\_\_\_Copy;

}

else if (displayCharacter == 'P')

{

img = Properties.Resources.tank1;

}

else if (displayCharacter == '.')

{

img = Properties.Resources.bullet1;

}

else if (displayCharacter == 'B')

{

img = Properties.Resources.bullet2;

}

else if (displayCharacter == 'V')

{

img = Properties.Resources.tank2;

}

else if (displayCharacter == 'H')

{

img = Properties.Resources.tank3;

}

else if (displayCharacter == 'R')

{

img = Properties.Resources.tank4;

}

else if (displayCharacter == 'S')

{

img = Properties.Resources.tank5;

}

return img;

}

}

### **GameCell Class:**

public class GameCell

{

int row;

int col;

GameObject currentGameObject;

GameGrid grid;

PictureBox pictureBox;

const int height = 20;

const int width = 20;

public GameCell(int row, int col, GameGrid grid)

{

this.row = row;

this.col = col;

PictureBox = new PictureBox();

PictureBox.Left = col \* width;

PictureBox.Top = row \* height;

PictureBox.Size = new Size(width, height);

PictureBox.SizeMode = PictureBoxSizeMode.Zoom;

PictureBox.BackColor = Color.Transparent;

this.grid = grid;

}

public void setGameObject(GameObject gameObject)

{

currentGameObject = gameObject;

PictureBox.Image = gameObject.Img;

}

public GameCell nextCell(GameDirection direction)

{

if (direction == GameDirection.Left)

{

if (this.col > 0)

{

GameCell ncell = grid.getCell(row, col - 1);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Right)

{

if (this.col < grid.Cols - 1)

{

GameCell ncell = grid.getCell(this.row, this.col + 1);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Up)

{

if (this.row > 0)

{

GameCell ncell = grid.getCell(this.row - 1, this.col);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Down)

{

if (this.row < grid.Rows - 1)

{

GameCell ncell = grid.getCell(this.row + 1, this.col);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

return this; // if can not return next cell return its own reference

}

public GameCell nextBulletCell(GameDirection direction)

{

if (direction == GameDirection.Left)

{

if (this.col > 0)

{

GameCell ncell = grid.getCell(row, col - 1);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Right)

{

if (this.col < grid.Cols - 1)

{

GameCell ncell = grid.getCell(this.row, this.col + 1);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Up)

{

if (this.row > 0)

{

GameCell ncell = grid.getCell(this.row - 1, this.col);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

if (direction == GameDirection.Down)

{

if (this.row < grid.Rows - 1)

{

GameCell ncell = grid.getCell(this.row + 1, this.col);

if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)

{

return ncell;

}

}

}

return this;

}

public int X { get => row; set => row = value; }

public int Y { get => col; set => col = value; }

public GameObject CurrentGameObject { get => currentGameObject; set => currentGameObject = value; }

public PictureBox PictureBox { get => pictureBox; set => pictureBox = value; }

public GameObject GameObject

{

get => default;

set

{

}

}

}

### **GameGrid Class:**

public class GameGrid

{

GameCell[,] cells;

int rows;

int cols;

public GameGrid(String fileName, int rows, int cols)

{

//Numbers of rows and cols should load from the text file

this.rows = rows;

this.cols = cols;

cells = new GameCell[rows, cols];

this.loadGrid(fileName);

}

public GameCell getCell(int x, int y)

{

return cells[x, y];

}

public int Rows { get => rows; set => rows = value; }

public int Cols { get => cols; set => cols = value; }

void loadGrid(string fileName)

{

StreamReader fp = new StreamReader(fileName);

string record;

for (int row = 0; row < this.rows; row++)

{

record = fp.ReadLine();

for (int col = 0; col < this.cols; col++)

{

GameCell cell = new GameCell(row, col, this);

Char displayCharacter = record[col];

GameObjectType type = GameObject.getGameObjectType(displayCharacter);

Image img = Game.getCharacterImage(displayCharacter);

GameObject gameObject = new GameObject(type, img);

cell.setGameObject(gameObject);

cells[row, col] = cell;

}

}

fp.Close();

}

public GameCell GameCell

{

get => default;

set

{

}

}

}

### **GameObject Class:**

public class GameObject

{

char displayCharacter;

GameObjectType gameObjectType;

GameCell currentCell;

Image img;

public GameObject(GameObjectType type, char displayCharacter)

{

this.displayCharacter = displayCharacter;

this.gameObjectType = type;

}

public GameObject(GameObjectType type, Image img)

{

this.gameObjectType = type;

this.Img = img;

}

public static GameObjectType getGameObjectType(char displayCharacter)

{

if (displayCharacter == '|' || displayCharacter == '\_' || displayCharacter == '!' || displayCharacter == '?')

{

return GameObjectType.WALL;

}

if (displayCharacter == '.')

{

return GameObjectType.BULLET;

}

if (displayCharacter == 'B')

{

return GameObjectType.ENEMYBULLET;

}

return GameObjectType.NONE;

}

public char DisplayCharacter { get => displayCharacter; set => displayCharacter = value; }

public GameObjectType GameObjectType { get => gameObjectType; set => gameObjectType = value; }

public GameCell CurrentCell

{

get => currentCell;

set

{

currentCell = value;

currentCell.setGameObject(this);

}

}

public Image Img { get => img; set => img = value; }

}

### **GameObjectType:**

public enum GameObjectType

{

WALL,

PLAYER,

ENEMY,

BULLET,

ENEMYBULLET,

NONE

}

### **GameDirection:**

public enum GameDirection

{

Left,

Right,

Up,

Down

}

### **GamePlayer Class:**

public class GamePlayer : GameObject

{

public GamePlayer(Image img, GameCell startCell) : base(GameObjectType.PLAYER, img)

{

this.CurrentCell = startCell;

}

public GameCell move(GameDirection direction)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = CurrentCell.nextCell(direction);

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

if (Collision.CheckCollision(currentCell))

{

if ( currentCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY)

{

MainForm.decreasePlayerHealth();

}

if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMYBULLET)

{

MainForm.decreaseScore();

}

}

currentCell.setGameObject(Game.getBlankObject());

}

return nextCell;

}

}

### **Enemy Class:**

abstract public class Enemy : GameObject

{

public Enemy(Image img, GameCell startCell) : base(GameObjectType.ENEMY, img)

{

this.CurrentCell = startCell;

}

public abstract void direction();

public abstract GameCell move();

}

### **EnemyVertical Class:**

class EnemyVertical : Enemy

{

public GameDirection gameDirection = GameDirection.Up;

public EnemyVertical(Image img, GameCell startCell) : base(img, startCell)

{

this.CurrentCell = startCell;

}

public override void direction()

{

if (gameDirection == GameDirection.Up)

{

gameDirection = GameDirection.Down;

}

else if (gameDirection == GameDirection.Down)

{

gameDirection = GameDirection.Up;

}

}

public override GameCell move()

{

GameCell CurrentCell = this.CurrentCell;

GameCell nextCell = CurrentCell.nextCell(gameDirection);

this.CurrentCell = nextCell;

if (CurrentCell == nextCell)

{

direction();

}

if (CurrentCell != nextCell && (nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL))

{

if (Collision.CheckCollision(CurrentCell))

{

if (CurrentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)

{

MainForm.addScore();

MainForm.decreaseVerticalEnemyHealth();

}

}

CurrentCell.setGameObject(Game.getBlankObject());

}

return nextCell;

}

}

### **EnemyHorizontal Class:**

class EnemyHorizontal : Enemy

{

public GameDirection gameDirection = GameDirection.Left;

public EnemyHorizontal(Image img, GameCell startCell) : base(img, startCell)

{

this.CurrentCell = startCell;

}

public override void direction()

{

if (gameDirection == GameDirection.Left)

{

gameDirection = GameDirection.Right;

}

else if (gameDirection == GameDirection.Right)

{

gameDirection = GameDirection.Left;

}

}

public override GameCell move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(gameDirection);

this.CurrentCell = nextCell;

if (currentCell == nextCell)

{

direction();

}

if (currentCell != nextCell && (nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL))

{

if (Collision.CheckCollision(currentCell))

{

if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)

{

MainForm.addScore();

MainForm.decreaseHorizontalEnemyHealth();

}

}

currentCell.setGameObject(Game.getBlankObject());

}

return nextCell;

}

}

### **EnemyRandom Class:**

class EnemyRandom : Enemy

{

GameDirection gameDirection;

public EnemyRandom(Image img, GameCell startCell) : base(img, startCell)

{

this.CurrentCell = startCell;

}

public override void direction()

{

if (gameDirection == GameDirection.Up)

{

gameDirection = GameDirection.Down;

}

else if (gameDirection == GameDirection.Down)

{

gameDirection = GameDirection.Up;

}

else if (gameDirection == GameDirection.Left)

{

gameDirection = GameDirection.Right;

}

else if (gameDirection == GameDirection.Right)

{

gameDirection = GameDirection.Left;

}

}

public void setRandomDirection()

{

Random r = new Random();

int value = r.Next(4);

if (value == 0)

{

gameDirection = GameDirection.Left;

}

else if (value == 1)

{

gameDirection = GameDirection.Right;

}

else if (value == 2)

{

gameDirection = GameDirection.Up;

}

else if (value == 3)

{

gameDirection = GameDirection.Down;

}

}

public override GameCell move()

{

setRandomDirection();

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(gameDirection);

this.CurrentCell = nextCell;

if (currentCell != nextCell && (nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL))

{

if (Collision.CheckCollision(currentCell))

{

if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)

{

MainForm.addScore();

MainForm.decreaseRandomEnemyHealth();

}

}

currentCell.setGameObject(Game.getBlankObject());

}

if (currentCell == nextCell)

{

direction();

}

return nextCell;

}

}

### **Bullet Class:**

public class Bullet : GameObject

{

GameDirection direction;

bool active;

public Bullet(GameCell startCell, GameDirection direction, Image img) : base(GameObjectType.BULLET, img)

{

this.CurrentCell = startCell;

this.Direction = direction;

this.Active = true;

}

public GameDirection Direction { get => direction; set => direction = value; }

public bool Active { get => active; set => active = value; }

}

### **EnemyBullet Class:**

public class EnemyBullet : GameObject

{

GameDirection direction;

bool active;

public EnemyBullet(GameCell startCell, GameDirection direction, Image img) : base(GameObjectType.ENEMYBULLET, img)

{

this.CurrentCell = startCell;

this.Direction = direction;

this.Active = true;

}

public GameDirection Direction { get => direction; set => direction = value; }

public bool Active { get => active; set => active = value; }

}

### **Collision Class:**

public class Collision

{

public GamePlayer GamePlayer

{

get => default;

set

{

}

}

public static bool CheckCollision(GameCell cell)

{

if (cell.CurrentGameObject.GameObjectType == GameObjectType.BULLET || cell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY)

{

return true;

}

return false;

}

}

## **Forms:**

### **Start Form:**

public partial class StartForm : Form

{

public StartForm()

{

InitializeComponent();

}

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

{

MainForm mainForm = new MainForm();

this.Hide();

mainForm.Show();

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

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

{

}

}

### **Win Form:**

public partial class WinForm : Form

{

public WinForm()

{

InitializeComponent();

}

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

{

this.Close();

}

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

{

}

}

### **Lose Form:**

public partial class LoseForm : Form

{

public LoseForm()

{

InitializeComponent();

}

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

{

}

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

{

}

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

{

this.Close();

}

}

### **Main Form:**

public partial class MainForm : Form

{

GamePlayer player;

public static int Score = 0;

public static int PlayerHealth = 5;

public static int VerticalEnemyHealth = 3;

public static int HorizontalEnemyHealth = 3;

public static int RandomEnemyHealth = 3;

EnemyHorizontal horEnemy;

EnemyVertical verEnemy;

EnemyRandom randEnemy;

static List<Enemy> enemies = new List<Enemy>();

public static List<EnemyBullet> enemyBullets = new List<EnemyBullet>();

public static List<Bullet> bullets = new List<Bullet>();

public MainForm()

{

InitializeComponent();

}

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

{

GameGrid gameGrid = new GameGrid("maze1.txt", 28, 67);

Image playerImage = Game.getCharacterImage('P');

Image verticalEnemyImage = Game.getCharacterImage('V');

Image horizontalEnemyImage = Game.getCharacterImage('H');

Image randomEnemyImage = Game.getCharacterImage('R');

Image bulletImage = Game.getCharacterImage('.');

GameCell start = gameGrid.getCell(8, 10);

GameCell ghostVCell = gameGrid.getCell(8, 6);

GameCell ghostHCell = gameGrid.getCell(15, 6);

GameCell ghostRCell = gameGrid.getCell(21, 7);

gameLoop.Start();

player = new GamePlayer(playerImage, start);

verEnemy = new EnemyVertical(verticalEnemyImage, ghostVCell);

horEnemy = new EnemyHorizontal(horizontalEnemyImage, ghostHCell);

randEnemy = new EnemyRandom(randomEnemyImage, ghostRCell);

enemies.Add(verEnemy);

enemies.Add(horEnemy);

enemies.Add(randEnemy);

printMaze(gameGrid);

}

private void printMaze(GameGrid gameGrid)

{

for (int x = 0; x < gameGrid.Rows; x++)

{

for (int y = 0; y < gameGrid.Cols; y++)

{

GameCell cell = gameGrid.getCell(x, y);

this.Controls.Add(cell.PictureBox);

}

}

}

private void moveEnemy()

{

foreach(Enemy e in enemies)

{

e.move();

}

}

public static void generateBullet(GamePlayer pacman, GameDirection direction)

{

GameCell startCell = pacman.CurrentCell.nextBulletCell(direction);

{

if (startCell != null)

{

Image bulletImage = Game.getCharacterImage('.');

Bullet bullet = new Bullet(startCell, direction, bulletImage);

bullets.Add(bullet);

startCell.setGameObject(bullet);

bullet.Active = true;

}

}

}

public void moveBullets()

{

List<Bullet> bulletsToRemove = new List<Bullet>();

foreach(Bullet bullet in bullets)

{

GameCell currentCell = bullet.CurrentCell;

GameCell nextCell = currentCell.nextBulletCell(bullet.Direction);

if(nextCell != currentCell)

{

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.NONE || nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

currentCell.setGameObject(Game.getBlankObject());

bullet.CurrentCell = nextCell;

nextCell.setGameObject(bullet);

bullet.Active = true;

}

else

{

bullet.Active = false;

}

}

else

{

bullet.Active = false;

}

if (!bullet.Active)

{

bulletsToRemove.Add(bullet);

}

}

foreach(Bullet bullet in bulletsToRemove)

{

bullet.CurrentCell.setGameObject(Game.getBlankObject());

bullets.Remove(bullet);

}

}

public static void decreaseVerticalEnemyHealth()

{

if (VerticalEnemyHealth > 0)

{

VerticalEnemyHealth = VerticalEnemyHealth - 1;

}

}

public static void decreaseHorizontalEnemyHealth()

{

if (HorizontalEnemyHealth > 0)

{

HorizontalEnemyHealth = HorizontalEnemyHealth - 1;

}

}

public static void decreaseRandomEnemyHealth()

{

if (RandomEnemyHealth > 0)

{

RandomEnemyHealth = RandomEnemyHealth - 1;

}

}

public static void addScore()

{

Score = Score + 1;

if (VerticalEnemyHealth == 0 && HorizontalEnemyHealth == 0 && RandomEnemyHealth == 0)

{

WinForm f = new WinForm();

f.Show();

}

}

public static void decreaseScore()

{

if (Score > 0)

{

Score = Score - 1;

}

}

public static void decreasePlayerHealth()

{

if (PlayerHealth > 0)

{

PlayerHealth = PlayerHealth - 1;

}

if (PlayerHealth == 0)

{

LoseForm f = new LoseForm();

f.Show();

}

}

public static void generateEnemyBullet(Enemy pacman, GameDirection direction)

{

GameCell startCell = pacman.CurrentCell.nextBulletCell(direction);

{

if (startCell != null)

{

Image bulletImage = Game.getCharacterImage('B');

EnemyBullet bullet = new EnemyBullet(startCell, direction, bulletImage);

enemyBullets.Add(bullet);

startCell.setGameObject(bullet);

bullet.Active = true;

}

}

}

public void moveEnemyBullets()

{

List<EnemyBullet> bulletsToRemove = new List<EnemyBullet>();

foreach (EnemyBullet bullet in enemyBullets)

{

GameCell currentCell = bullet.CurrentCell;

GameCell nextCell = currentCell.nextBulletCell(bullet.Direction);

if (nextCell != currentCell)

{

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.NONE || nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

currentCell.setGameObject(Game.getBlankObject());

bullet.CurrentCell = nextCell;

nextCell.setGameObject(bullet);

bullet.Active = true;

}

else

{

bullet.Active = false;

}

}

else

{

bullet.Active = false;

}

if (!bullet.Active)

{

bulletsToRemove.Add(bullet);

}

}

foreach (EnemyBullet bullet in bulletsToRemove)

{

bullet.CurrentCell.setGameObject(Game.getBlankObject());

enemyBullets.Remove(bullet);

}

}

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

{

if (Keyboard.IsKeyPressed(Key.LeftArrow))

{

player.move(GameDirection.Left);

}

if (Keyboard.IsKeyPressed(Key.RightArrow))

{

player.move(GameDirection.Right);

}

if (Keyboard.IsKeyPressed(Key.UpArrow))

{

player.move(GameDirection.Up);

}

if (Keyboard.IsKeyPressed(Key.DownArrow))

{

player.move(GameDirection.Down);

}

if (Keyboard.IsKeyPressed(Key.Space))

{

generateBullet(player, GameDirection.Left);

}

moveBullets();

generateEnemyBullet(verEnemy, GameDirection.Right);

generateEnemyBullet(horEnemy, GameDirection.Up);

generateEnemyBullet(randEnemy, GameDirection.Right);

moveEnemy();

moveEnemyBullets();

lblScore.Text = "Score = " + Score.ToString();

lblVerHealth.Text = "VerticalEnemyHealth = " + VerticalEnemyHealth.ToString();

lblHorHealth.Text = "HorizontalEnemyHealth = " + HorizontalEnemyHealth.ToString();

lblRandHealth.Text = "RandomEnemyHealth = " + RandomEnemyHealth.ToString();

lblHealth.Text = "PlayerHealth = " + PlayerHealth.ToString();

}

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

{

}

}

# **Conclusion:**

I have developed this game by using all OOP (Object Oriented Programming) concepts. This game has proved to be entertaining as well as interesting at the same time. I hope to improve this game further and make it even better in coming future.