QATAR FIFA MANIA

Session: 2022 – 2025

Submitted by:

Saba Shahdin 2022-CS-112

Supervised by:

Sir Awais Hassan

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

Story of Qatar FIFA Mania:

Al Rihla is ambitious about playing. He loves to play any sport whether it is cricket or football. But inspired by **FIFA WORLD CUP 2022**. He wants to play football and for 2 months he has been practicing to get perfection in scoring a goal daily.
He plays in a football stadium full of adventures to find pleasure as well as to get skills. As he moves he finds **food pallets** which increase his score and encourages him to go ahead. As he goes on he finds **energy pallets** that increase his energy and gives the power to fire his enemies. He has to go to the **goalpost** protecting himself from the firings of the **defenders** and from the **goalkeeper** who is ready to push him away from the goalpost.
After facing all these hardships on the field he succeeded to score a **goal** which makes him feel better as his practice is going on and becoming better day by day. Al Rihla is happy after killing defenders with his power and making the goalkeeper fail in his efforts by scoring the goal. He will soon be the best goal-scoring player. He will make it happen with his efforts.

Objective of game:

The objective of game is to reach the goal post and get the trophy there to win the game. Player has to go to goal post saving from defenders and their fires. In its path he can get bonus scores and rewards. Player can fire to break the wall. He has 3 lives to achieve this goal.

Challenges of Player:

Player have to save himself from the moving ghost and their fires and he has only 3 lives to achieve this goal.

Game Characters Description:

Game Characters Description

Players:

There is one human player in the Game.

Al-Rihla:

Al Rihla is the human ambitious player of the game. He is a blue color throughout. He loves to play football and finds himself happy scoring goals and eating food pallets as well as energy pallets which encourages him and makes him powerful enough to defeat his enemies.

Enemies:

There are 3 enemies in the game.

- Defender 1:

  Defender 1 is the enemy of Al Rihla which moves in the horizontal direction and fires in the upward and downward direction to distract al Rihla from going to the goalpost

- Defender 2:

  Defender 2 is the enemy of al Rihla which moves randomly direction to distract Al Rihla from going to goalpost.

- Goal Keeper:

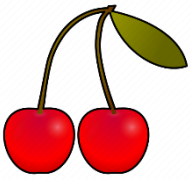
  Defender 3 is al Rihla's enemy moves vertically in the field and fire in the left distracting him from his goal.

Resources:

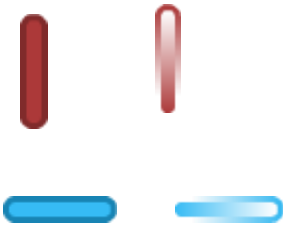
Field (None)

Player Ball

Defenders



Score and Bonus



Bullets

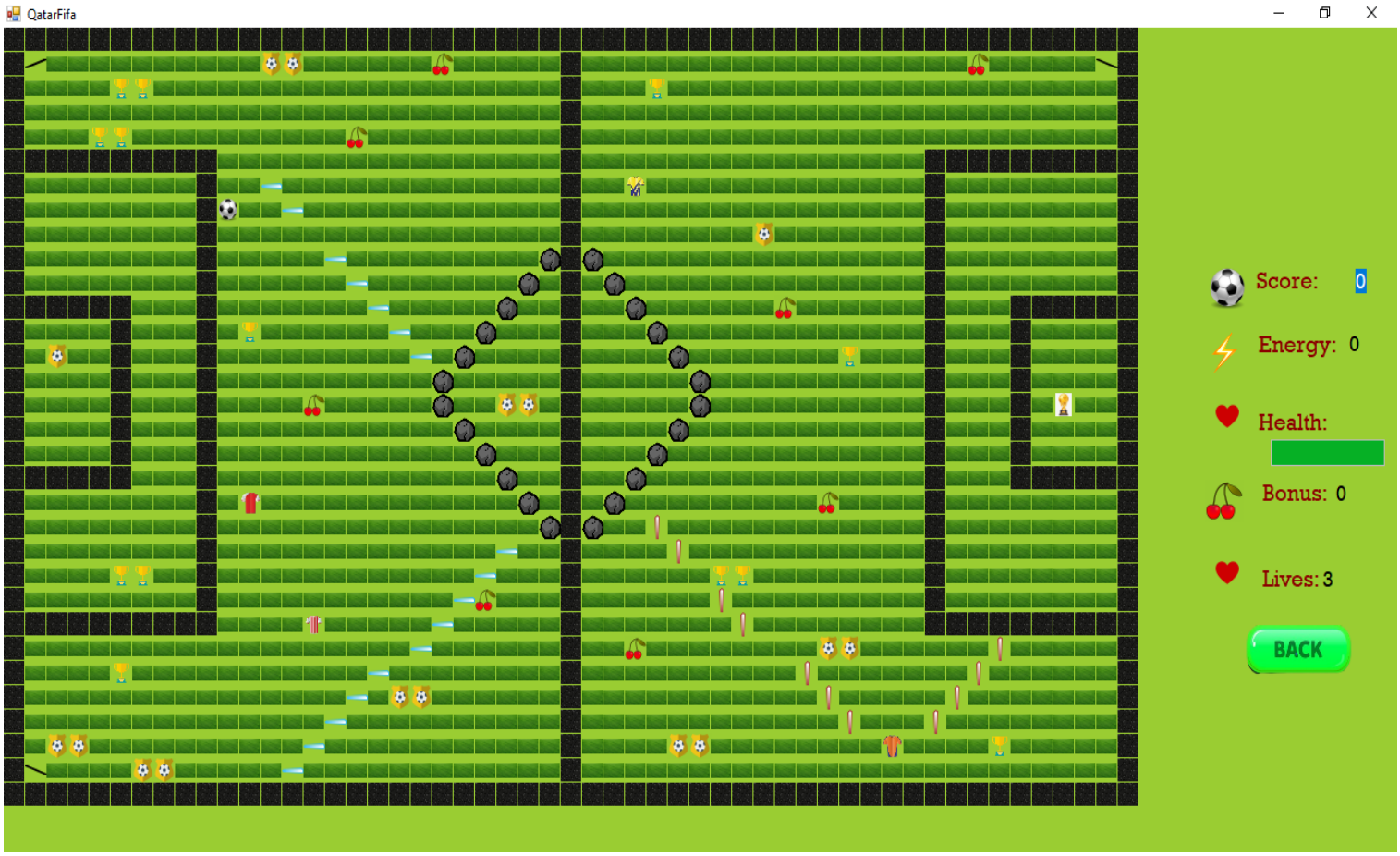
Buttons

Win and Lose

Game Graphics:

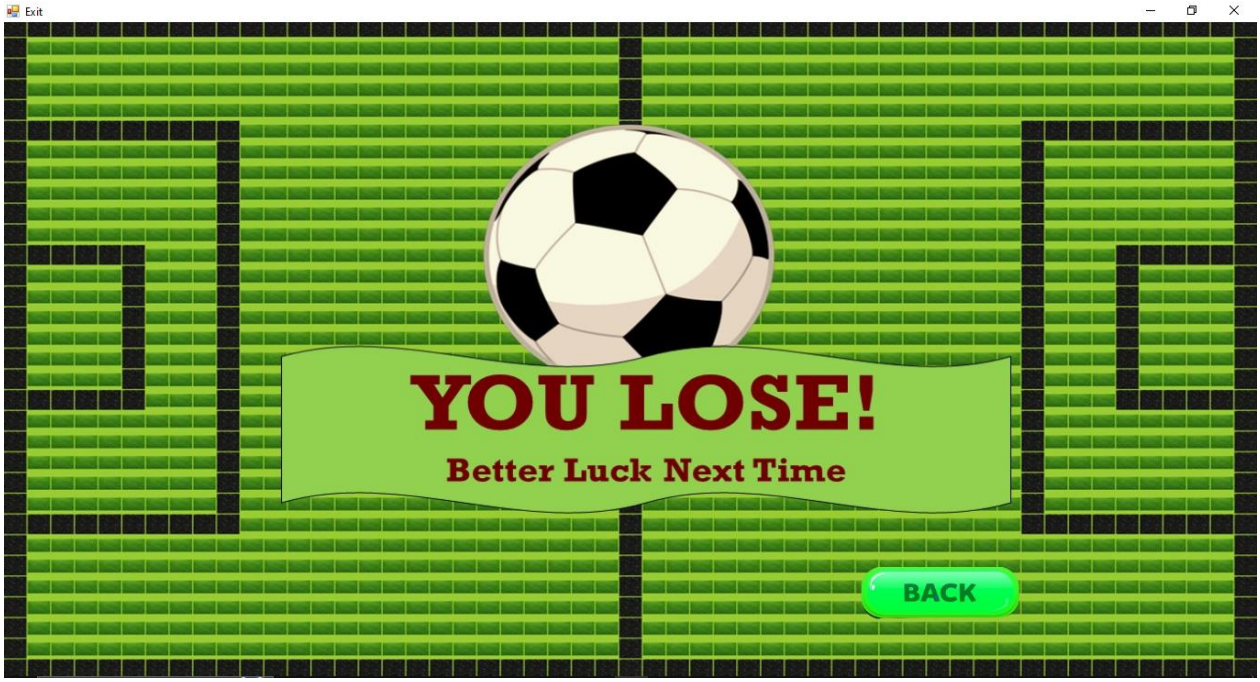
- Main Screen:

- Game Play:

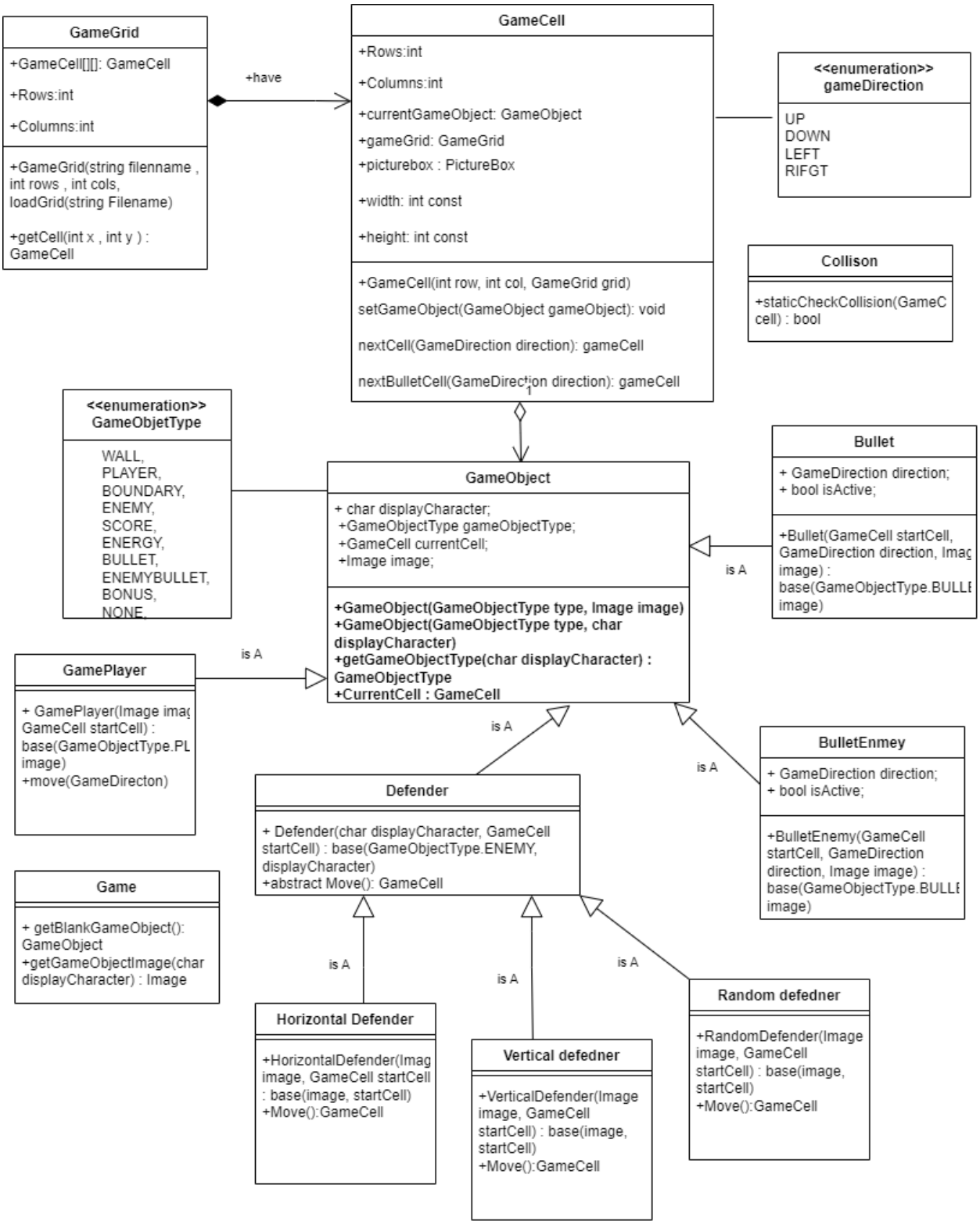


- Win Form:

- Lose Form:



Class Diagram:



Code:

- ## Game Grid:

```
public class GameGrid
{
    GameCell[,] cells;
    int rows;
    int cols;
    public GameGrid(String fileName, int rows, int cols)
    {
        this.Rows = rows;
        this.Cols = cols;
        cells = new GameCell[rows, cols];
        this.loadGrid(fileName);
    }
    public int Rows { get => rows; set => rows = value; }
    public int Cols { get => cols; set => cols = value; }
    public GameCell getCell(int x, int y)
    {
        return cells[x, y];
    }
    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 displayImge = Game.getGameObjectImage(displayCharacter);
                GameObject gameObject = new GameObject(type, displayImge);
                cell.setGameObject(gameObject);
                cells[row, col] = cell;
            }
        }

        fp.Close();
    }

}
```

- ## Game Cell:

```
public class GameCell
  {
    int row;
```

```
    int col;
    GameObject currentGameObject;
    GameGrid grid;
    PictureBox pictureBox;
    int score = 0;
    const int width = 21;
    const int height = 21;
    public int Row { get => row; set => row = value; }
    public int Col { get => col; set => col = value;
    public GameObject CurrentGameObject { get => currentGameObject; set => currentGameObject = value; }
    public GameGrid Grid { get => grid; set => grid = value; }
    public PictureBox PictureBox { get => pictureBox; set => pictureBox = value; }
    public int Score { get => score; set => score = value; }
    public GameCell()
    {
    }
    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.Image;

    }
    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 &&
ncell.CurrentGameObject.GameObjectType != GameObjectType.BOUNDARY)
          {
            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 &&
ncell.CurrentGameObject.GameObjectType != GameObjectType.BOUNDARY )
                {
                    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 &&
ncell.CurrentGameObject.GameObjectType != GameObjectType.BOUNDARY)
                {
                    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 &&
ncell.CurrentGameObject.GameObjectType != GameObjectType.BOUNDARY )
                {
                    return ncell;
                }
            }
        }
        return this;
    }
    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.BOUNDARY)
                {
```

```
                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.BOUNDARY)
            {
                return ncell;
            }
        }
    }

    if (direction == GameDirection.Up)
    {
        if (this.Row > 0)
        {
            GameCell ncell = Grid.getCell(this.Row - 1, this.Col);
            if (ncell.CurrentGameObject.GameObjectType != GameObjectType.BOUNDARY)
            {
                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.BOUNDARY)
            {
                return ncell;
            }
        }
    }
    return this;
  }
}
```

- Game Object:

```
public class GameObject
{
  char displayCharacter;
  GameObjectType gameObjectType;
  GameCell currentCell;
  Image image;
```

```
public GameObject(GameObjectType type, Image image)
{
   this.GameObjectType = type;
   this.Image = image;
}
public GameObject(GameObjectType type, char displayCharacter)
{
   this.GameObjectType = type;
   this.DisplayCharacter = displayCharacter;
}

public static GameObjectType getGameObjectType(char displayCharacter)
{
   if (displayCharacter == '_')
   {
      return GameObjectType.WALL;
   }
   if (displayCharacter == '|')
   {
      return GameObjectType.WALL;
   }
   if (displayCharacter == '#')
   {
      return GameObjectType.BOUNDARY;
   }
   if (displayCharacter == '>')
   {
      return GameObjectType.SCORE;
   }
   if (displayCharacter == '$')
   {
      return GameObjectType.ENERGY;
   }
   if (displayCharacter == 'P')
   {
      return GameObjectType.PLAYER;
   }
   if (displayCharacter == '*')
   {
      return GameObjectType.WIN;
   }
   if (displayCharacter == '^')
   {
      return GameObjectType.BONUS;
   }
   if (displayCharacter == 'H')
   {
      return GameObjectType.ENEMY;
   }
   if (displayCharacter == 'V')
```

```
        {
            return GameObjectType.ENEMY;
        }
        if (displayCharacter == 'v')
        {
            return GameObjectType.ENEMY;
        }
        if (displayCharacter == 'R')
        {
            return GameObjectType.ENEMY;
        }
        if (displayCharacter == '-')
        {
            return GameObjectType.BULLET;
        }
        if (displayCharacter == '&')
        {
            return GameObjectType.BULLET;
        }
        if (displayCharacter == 'A')
        {
            return GameObjectType.ENEMYBULLET;
        }
        if (displayCharacter == 'B')
        {
            return GameObjectType.ENEMYBULLET;
        }

        return GameObjectType.NONE;
    }

    public GameCell CurrentCell
    {
        get => CurrentCell1;
        set
        {
            CurrentCell1 = value;
            CurrentCell1.setGameObject(this);
        }
    }
public char DisplayCharacter { get => displayCharacter; set =>displayCharacter = value; }
public GameObjectType GameObjectType { get => gameObjectType; set =>gameObjectType = value; }
public GameCell CurrentCell1 { get => currentCell; set => currentCell = value; }
public Image Image { get => image; set => image = value; }
  }
```

- Game Object Type:

```
public enum GameObjectType
  {
    WALL,
```

```
    PLAYER,
    BOUNDARY,
    ENEMY,
    SCORE,
    ENERGY,
    BULLET,
    ENEMYBULLET,
    BONUS,
    NONE,
    WIN
}
```

- Game:

```
public class Game
{
    public static GameObject getBlankGameObject()
    {
        GameObject blankGameObject = new GameObject(GameObjectType.NONE,
game.Properties.Resources.Field);
        return blankGameObject;
    }
    public static Image getGameObjectImage(char displayCharacter)
    {
        Image img = game.Properties.Resources.Field;
        if (displayCharacter == '|' || displayCharacter == '_')
        {
            img = game.Properties.Resources.Rocks;
        }
        if (displayCharacter == '#')
        {
            img = game.Properties.Resources.Rocks;
        }
        if (displayCharacter == '/')
        {
            img = game.Properties.Resources.lINESS__2_;
        }
        if (displayCharacter == '\\')
        {
            img = game.Properties.Resources.Slant;
        }
        if (displayCharacter == '>')
        {
            img = game.Properties.Resources.Medal;
        }
        if (displayCharacter == '@')
        {
            img = game.Properties.Resources.rock_icon_14;
        }
        if (displayCharacter == '$')
        {
```

```
            img = game.Properties.Resources.Score;
        }
        if (displayCharacter == 'P' )
        {
            img = game.Properties.Resources.PlayerBall;
        }
        if (displayCharacter == '^')
        {
            img = game.Properties.Resources.Bonus;
        }
        if (displayCharacter == '*')
        {
            img = game.Properties.Resources.wine;
        }
        if (displayCharacter == 'H')
        {
            img = game.Properties.Resources.HorizontalGhost;
        }
        if (displayCharacter == 'V')
        {
            img = game.Properties.Resources.VerticalGhost;
        }
        if (displayCharacter == 'v')
        {
            img = game.Properties.Resources.GoalKeeper;
        }
        if (displayCharacter == 'R')
        {
            img = game.Properties.Resources.Random;
        }
        if (displayCharacter == '-')
        {
            img = game.Properties.Resources.laserRed12;
        }
        if (displayCharacter == '&')
        {
            img = game.Properties.Resources.laserBlue12;
        }
        if (displayCharacter == 'A')
        {
            img = game.Properties.Resources.laserRed16;
        }
        if (displayCharacter == 'B')
        {
            img = game.Properties.Resources.laserBlue16;
        }

        return img;
    }
}
```

- Game Player:

```
public class GamePlayer : GameObject
  {
 public GamePlayer(Image image, GameCell startCell) : base(GameObjectType.PLAYER, image)
    {
      this.CurrentCell = startCell;
    }
    public void move(GameDirection direction)
    {
      GameCell currentCell = this.CurrentCell;
      GameCell nextCell = currentCell.nextCell(direction);
      if (nextCell != currentCell && nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL &&
nextCell.CurrentGameObject.GameObjectType != GameObjectType.BOUNDARY)
      {
        if (Collision.CheckCollision(nextCell))
        {
          if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.SCORE)
          {
            QatarFifa.Scoring();
            nextCell.setGameObject(Game.getBlankGameObject());
          }
          if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.BONUS)
          {
            QatarFifa.bonus();
            nextCell.setGameObject(Game.getBlankGameObject());
          }
         if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.ENERGY)
          {
            QatarFifa.energize();
            nextCell.setGameObject(Game.getBlankGameObject());
          }
          if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY)
          {
            QatarFifa.DecreaseHealth();
          }
       if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMYBULLET)
          {
            QatarFifa.DecreaseHealth();
          }
          if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.WIN)
          {
            Form form = new WinForm();
            form.Show();
          }
        }
        this.CurrentCell = nextCell;
        currentCell.setGameObject(Game.getBlankGameObject());
      }
    }
  }
```

- Defender:

```
public abstract class Defender : GameObject
  {
     GameDirection currentDirection;
 public GameDirection CurrentDirection { get => currentDirection; set => currentDirection = value; }
     public Defender(char displayCharacter, GameCell startCell) : base(GameObjectType.ENEMY, displayCharacter)
     {
        this.CurrentCell = startCell;
     }
    public Defender(Image image, GameCell startCell) : base(GameObjectType.ENEMY, image)
     {
        this.CurrentCell = startCell;
     }
     public abstract GameCell Move();
  }
```

- Horizontal Defender:

```
public class HorizontalDefender : Defender
  {
     private GameCell initialCell;
    public HorizontalDefender(Image image, GameCell startCell) : base(image, startCell)
     {
        this.CurrentCell = startCell;
        this.initialCell = startCell;
     }
     public override GameCell Move()
     {
        GameCell currentCell = this.CurrentCell;
        GameCell nextCell = currentCell.nextCell(GameDirection.Right);
        this.CurrentCell = nextCell;
 if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL || nextCell == currentCell)
        {
           currentCell.setGameObject(Game.getBlankGameObject());
           this.CurrentCell = this.initialCell;
           nextCell = this.initialCell.nextCell(GameDirection.Left);
        }

        else if (currentCell != nextCell)
        {
           if(Collision.CheckCollision(currentCell))
           {
            if( currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)
              {
                 QatarFifa.Scoring();
              }
           }
           currentCell.setGameObject(Game.getBlankGameObject());
        }
        return nextCell;
     }
```

```
  }
```

- Vertical Defender:

```
public class VerticalDefender : Defender
  {
    private GameCell initialCell;
    public VerticalDefender(Image image, GameCell startCell) : base(image, startCell)
    {
      this.CurrentCell = startCell;
      this.initialCell = startCell;
    }
    public override GameCell Move()
    {
      GameCell currentCell = this.CurrentCell;
      GameCell nextCell = currentCell.nextCell(GameDirection.Down);
      this.CurrentCell = nextCell;
      if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL || currentCell == nextCell)
      {
        currentCell.setGameObject(Game.getBlankGameObject());
        this.CurrentCell = this.initialCell;
        nextCell = this.initialCell.nextCell(GameDirection.Up);
      }
      else if (currentCell != nextCell)
      {
        if (Collision.CheckCollision(currentCell))
        {
        if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)
          {
            QatarFifa.Scoring();
          }
        }
        currentCell.setGameObject(Game.getBlankGameObject());
      }
      return nextCell;

    }
  }
```

- Random Defender:

```
public class RandomDefender : Defender
  {
    public RandomDefender(Image image, GameCell startCell) : base(image, startCell)
    {
      this.CurrentCell = startCell;
    }
    public int ghostDirection()
    {
      Random r = new Random();
      int value = r.Next(4);
      return value;
```

```
    }
    public override GameCell Move()
    {
        GameCell currentCell = this.CurrentCell;
        GameDirection currentDirection = CurrentDirection;
        GameCell nextCell = currentCell;
        int value = ghostDirection();
        if (value == 0)
        {
            if ((nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL &&
nextCell.CurrentGameObject.GameObjectType == GameObjectType.PLAYER) || nextCell == currentCell)
            {
                currentDirection = GameDirection.Left;
                nextCell = currentCell.nextCell(currentDirection);
                if (Collision.CheckCollision(currentCell))
                {
                    if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)
                    {
                        QatarFifa.Scoring();
                    }
                }
                currentCell.setGameObject(Game.getBlankGameObject());
                this.CurrentCell = nextCell;
            }
        }
        if (value == 1)
        {
            if ((nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL &&
nextCell.CurrentGameObject.GameObjectType == GameObjectType.PLAYER) || nextCell == currentCell)
            {
                currentDirection = GameDirection.Right;
                nextCell = currentCell.nextCell(currentDirection);
                if (Collision.CheckCollision(currentCell))
                {
                    if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)
                    {
                        QatarFifa.Scoring();
                    }

                }
                currentCell.setGameObject(Game.getBlankGameObject());
                this.CurrentCell = nextCell;
            }
        }
        if (value == 2)
        {
            if ((nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL &&
nextCell.CurrentGameObject.GameObjectType == GameObjectType.PLAYER) || nextCell == currentCell)
            {
                currentDirection = GameDirection.Up;
```

```
            nextCell = currentCell.nextCell(currentDirection);
            if (Collision.CheckCollision(currentCell))
            {
              if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)
              {
                QatarFifa.Scoring();
              }

            }
            currentCell.setGameObject(Game.getBlankGameObject());
            this.CurrentCell = nextCell;
          }
        }
        if (value == 3)
        {
          if ((nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL &&
nextCell.CurrentGameObject.GameObjectType == GameObjectType.PLAYER) || nextCell == currentCell)
          {
            currentDirection = GameDirection.Down;
            nextCell = currentCell.nextCell(currentDirection);
            if (Collision.CheckCollision(currentCell))
            {
              if (currentCell.CurrentGameObject.GameObjectType == GameObjectType.BULLET)
              {
                QatarFifa.Scoring();
              }

            }
            currentCell.setGameObject(Game.getBlankGameObject());
            this.CurrentCell = nextCell;
          }
        }
        return nextCell;
      }
    }
```

- Collision:

```
public static class Collision
  {

    public static bool CheckCollision(GameCell cell)
    {
      if (cell.CurrentGameObject.GameObjectType == GameObjectType.SCORE ||
cell.CurrentGameObject.GameObjectType == GameObjectType.BONUS ||
cell.CurrentGameObject.GameObjectType == GameObjectType.ENERGY ||
cell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY ||
cell.CurrentGameObject.GameObjectType == GameObjectType.BULLET ||
cell.CurrentGameObject.GameObjectType == GameObjectType.ENEMYBULLET ||
cell.CurrentGameObject.GameObjectType == GameObjectType.WIN)
        {
```

```
            return true;
        }
        return false;
    }

}
```

- Bullet:

```
public class Bullet : GameObject
{
    private GameDirection direction;
    private bool isActive;

    public Bullet(GameCell startCell, GameDirection direction, Image image) : base(GameObjectType.BULLET,
image)
    {
        this.CurrentCell = startCell;
        this.direction = direction;
        this.isActive = true;
    }

    public GameDirection Direction { get => direction; set => direction = value; }
    public bool IsActive { get => isActive; set => isActive = value;
}
```

- Bullet Enemy:

```
public class BulletEnemy : GameObject
{
    private GameDirection direction;
    private bool isActive;

    public BulletEnemy(GameCell startCell, GameDirection direction, Image image) :
base(GameObjectType.ENEMYBULLET, image)
    {
        this.CurrentCell = startCell;
        this.direction = direction;
        this.isActive = true
 }

    public GameDirection Direction { get => direction; set => direction = value; }
    public bool IsActive { get => isActive; set => isActive = value; }
}
```

- Qatar Fifa Main Form:

```
public partial class QatarFifa : Form

{

    GamePlayer pacman;

    public static int Score = 0;
```

```
public static int Health = 100;
public static int Bonus = 0;
public static int Energy = 0;
public static int Lives = 3;
HorizontalDefender horizontal;
VerticalDefender goalkeeper;
GameGrid grid;
public static List<Defender> ghost = new List<Defender>();
public static List<Bullet> Bullets = new List<Bullet>();
public static List<BulletEnemy> enemybullets = new List<BulletEnemy>();
public QatarFifa()
{
   InitializeComponent();
}
private void QatarFifa_Load(object sender, EventArgs e)
{
   GameGrid grid = new GameGrid("mazes.txt", 32, 53);
   Image pacManImage = Game.getGameObjectImage('P');
   Image bulletImage = Game.getGameObjectImage('-');
   Image HorizontalGhost = Game.getGameObjectImage('H');
   Image VerticalGhost = Game.getGameObjectImage('V');
   Image GoalKeeperGhost = Game.getGameObjectImage('v');
   Image randomGhost = Game.getGameObjectImage('R');
   GameCell startCell = grid.getCell(7, 10);
   GameCell startGhost = grid.getCell(19, 11);
   GameCell Defender = grid.getCell(5, 31);
   GameCell GoalKeeper = grid.getCell(3, 41);
   GameCell startRandom = grid.getCell(29, 15);
   GameLoop.Start();
```

```
    bulletTimer.Start();
    pacman = new GamePlayer(pacManImage, startCell);
    horizontal = new HorizontalDefender(HorizontalGhost, startGhost);
    Defender vertical = new RandomDefender(VerticalGhost, Defender);
    Defender random = new RandomDefender(randomGhost, startRandom);
    goalkeeper = new VerticalDefender(GoalKeeperGhost, GoalKeeper);
    printMaze(grid);
    ghost.Add(horizontal);
    ghost.Add(vertical);
    ghost.Add(random);
    ghost.Add(goalkeeper);
}
void printMaze(GameGrid grid)
{
    for (int x = 0; x < grid.Rows; x++)
    {
        for (int y = 0; y < grid.Cols; y++)
        {
            GameCell cell = grid.getCell(x, y);
            this.Controls.Add(cell.PictureBox);
        }
    } }
public static void moveGhost()
{
    foreach (Defender g in ghost)
    {
        g.Move();
    }
}
```

```
private void GameLoop_Tick(object sender, EventArgs e)
{
   if (Keyboard.IsKeyPressed(Key.LeftArrow))
   {
      pacman.move(GameDirection.Left);
   }
   if (Keyboard.IsKeyPressed(Key.RightArrow))
   {
      pacman.move(GameDirection.Right);
   }
   if (Keyboard.IsKeyPressed(Key.UpArrow))
   {
      pacman.move(GameDirection.Up);
   }
   if (Keyboard.IsKeyPressed(Key.DownArrow))
   {
      pacman.move(GameDirection.Down);
   }
   generateEnemyBullet(horizontal, GameDirection.Up);
   generateEnemyBullet(horizontal, GameDirection.Down);
   generateEnemyBullet(goalkeeper, GameDirection.Left);
   generateEnemyBullet(goalkeeper, GameDirection.Right);
   MoveEnemyBullets();
   moveGhost();
   txtScore.Text = Score.ToString();
   txtBonus.Text = Bonus.ToString();
   txtEnergy.Text = Energy.ToString();
   txtLivess.Text = Lives.ToString();
   healthBar.Value = Health;
```

```
}
private void bulletTimer_Tick_1(object sender, EventArgs e)
{
   if (Keyboard.IsKeyPressed(Key.A))
   {
      generateBullet(pacman, GameDirection.Left);
   }
   if (Keyboard.IsKeyPressed(Key.Space))
   {
      generateBullet(pacman, GameDirection.Right);
   }
   if (Keyboard.IsKeyPressed(Key.S))
   {
      generateBullet(pacman, GameDirection.Up);
   }
   if (Keyboard.IsKeyPressed(Key.D))
   {
      generateBullet(pacman, GameDirection.Down);
   }
   MoveBullets();
}
public static void generateBullet(GamePlayer pacman, GameDirection direction)
{
   GameCell startCell = pacman.CurrentCell.nextBulletCell(direction);
   if (startCell != null)
   {
      if (direction == GameDirection.Up || direction == GameDirection.Down)
      {
         Image bulletImage = Game.getGameObjectImage('-');
```

```
            Bullet bullet = new Bullet(startCell, direction, bulletImage);
            Bullets.Add(bullet);
            startCell.setGameObject(bullet);
            bullet.IsActive = true;
        }
        if (direction == GameDirection.Left || direction == GameDirection.Right)
        {
            Image bulletImage = Game.getGameObjectImage('&');
            Bullet bullet = new Bullet(startCell, direction, bulletImage);
            Bullets.Add(bullet);
            startCell.setGameObject(bullet);
            bullet.IsActive = 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.getBlankGameObject());
              bullet.CurrentCell = nextCell;
              nextCell.setGameObject(bullet);
```

```
                bullet.IsActive = true;
            }
            else
            {
                bullet.IsActive = false;
            }
        }
        else
        {
            bullet.IsActive = false;
        }
        if (!bullet.IsActive)
        {
            bulletsToRemove.Add(bullet);
        }
    }
    foreach (Bullet bullet in bulletsToRemove)
    {
        bullet.CurrentCell.setGameObject(Game.getBlankGameObject());
        Bullets.Remove(bullet);
    }
}
public static void DecreaseHealth()
{
    Health = Health - 1;
    if (Health == 0)
    {
        Lives = Lives - 1;
        Health = 100;
```

```
    }
}
public static void DecreaseLives()
{
    if (Lives == 0)
    {
        Form form = new Lose();
        form.Show();
    }
}
public static void Scoring()
{
    Score = Score + 1;
}
public static void bonus()
{
    Bonus = Bonus + 5;
}
public static void energize()
{
    Energy = Energy + 1;
}
public void generateEnemyBullet(Defender defender, GameDirection direction)
{
    GameCell startCell = defender.CurrentCell.nextCell(direction);
    if (startCell != null)
    {
        if (direction == GameDirection.Up || direction == GameDirection.Down)
        {
```

```
            Image bulletImage = Game.getGameObjectImage('B');
            BulletEnemy enemyBullet = new BulletEnemy(startCell, direction, bulletImage);
            enemybullets.Add(enemyBullet);
            startCell.setGameObject(enemyBullet);
            enemyBullet.IsActive = true;
        }
        if (direction == GameDirection.Left || direction == GameDirection.Right)
        {
            Image bulletImage = Game.getGameObjectImage('A');
            BulletEnemy enemyBullet = new BulletEnemy(startCell, direction, bulletImage);
            enemybullets.Add(enemyBullet);
            startCell.setGameObject(enemyBullet);
            enemyBullet.IsActive = true;
        }
    }
}
public void MoveEnemyBullets()
{
    List<BulletEnemy> bulletsToRemove = new List<BulletEnemy>();
    foreach (BulletEnemy bullet in enemybullets)
    {
        GameCell currentCell = bullet.CurrentCell;
        GameCell nextCell = currentCell.nextCell(bullet.Direction);
        if (nextCell != currentCell)
        {
            if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.NONE)
            {
                currentCell.setGameObject(Game.getBlankGameObject());
                bullet.CurrentCell = nextCell;
```

```
                nextCell.setGameObject(bullet);
                bullet.IsActive = true;
            }
            else
            {
                bullet.IsActive = false;
            }
        }
        else
        {
            bullet.IsActive = false;
        }
        if (!bullet.IsActive)
        {
            bulletsToRemove.Add(bullet);
        }
    }
    foreach (BulletEnemy bullet in bulletsToRemove)
    {
        bullet.CurrentCell.setGameObject(Game.getBlankGameObject());
        enemybullets.Remove(bullet);
    }
}
private void btnBack_Click(object sender, EventArgs e)
{
    Form form = new Main();
    form.Show();
    this.Hide();
}}
```

- Main Form:

```
public partial class Main : Form
  {
    public Main()
    {
      InitializeComponent();
    }
    private void button1_Click(object sender, EventArgs e)
    {
      Form form = new QatarFifa();
      form.Show();
      this.Hide();
    }
    private void btnExit_Click(object sender, EventArgs e)
    {
      Application.Exit();
    }
  }
```

- Win Form:

```
  public partial class WinForm : Form
  {
    public WinForm()
    {
      InitializeComponent();
    }

    private void btnBack_Click(object sender, EventArgs e)
    {
      Form form = new Main();
      form.Show();
      this.Hide();
    }
  }
```

- Lose Form:

```
public partial class Lose : Form
  {
    public Lose()
    {
      InitializeComponent();
    }

    private void button1_Click(object sender, EventArgs e)
    {
      Form form = new Main();
      form.Show();
      this.Hide();
    }
  }
```