Mission Rescue

Session: 2022 – 2026

Submitted by:

Fasi-ur-Rehman 2022-CS-71

Supervised by:

Professor Awais Hassan

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

Youtube Link:

https://www.youtube.com/watch?v=NPw0A5SHSog&ab_channel=FasiTahir

1. Short Story:

Drum kingdom was a magical land filled with snow and ice. The name of princess of that kingdom was ‘Princess Rosa’. She was famous in other kingdoms because of her beauty and the gold and diamond she had. The kingdom was very prosper. Everyone was living happily until the ‘King Grime’, king of a kingdom known as hell, kidnaped princess. Everyone was worried then the hero of our game emerged his name was ‘Nick’, He left the kingdom on the quest of rescuing the princess alone. He had to face different small villains of the game and by defeating them he will progress towards his final goal (rescuing the princess). He will fight the demon king Grime and will rescue the princess by defeating him.

During the quest main character will defeat the enemies by shooting fire. He will get points by defeating enemies and at the end of level he will get bonus points by collecting cards. Some enemies will shoot back and the health of our main character will decrease. There will be total of 3 lives of our character.

2. Game Characters Description

2.1. Player

There is one human player in the Game.

2.1.1. Nick:

Nick is the main character of our game. He is warrior on the way to rescue the princess from the King Grime. Nick is a brave person who loves to do go on different quests time to time. He is not afraid of challenges. He face them with a wide smile on his face. He can sacrifices anything for the sake of his loved ones. He is indeed the hero for everyone who know him.

2.2. Enemies

There are 3 types of enemies in the game.

2.2.1. Skeleton:

Ghosts are the one of 3 types of enemies. They are created by the black magic of demon king Grime. Skeleton will move randomly in the maze they will collide with the main character. If they get successful in colliding with the main character. He will loss a heart immediately.

2.2.2. Flying Beast:

Wizards are 2nd type of enemy which our character will face. They are very skilled in their work. They move vertically up and down in the maze. They are capable of throwing Fire Ball at Nick. Nick will lose some health once he will get hit by a bullets.

2.2.3. King Grime:

King Grime is the final boss our character will face after defeating his subordinates. After defeating him our character will become successful in his quest of rescuing the princess.

3. Game Objects Description

Following are the Objects in the Game

3.1. Spike Walls:

There are spike walls in the game they have sharp edges. Player will lose health if he collide with spike walls.

3.2. Heart:

There is heart at two places in our game. The live of Nick will increase if he pick up hearts from the maze.

3.3. Walls:

Walls are the barriers in the game which the Nick and the enemies cannot cross.

4. Difficulties for Nick:

As we discusses all of the objects earlier. Those are difficulties for our player, Nick. We will summarize those obstacles here:

- He will face 3 enemies in this game. If he collide with them he will lose one of 3 hearts and will spawn back to initial position. He can increase life by picking up hearts.

- He will face Spike Walls whenever he will collide with them he will lose health.
- He will face bullets of enemies. He will lose health if he gets hit.
- There are ordinary walls also. He cannot move through the walls.

5. Objective of the Game

The ultimate goal of the game is to rescue the Princess Rosa from the king Grime. He will achieve that after defeating the subordinates and then the king Grime himself.

6. Images of Game:

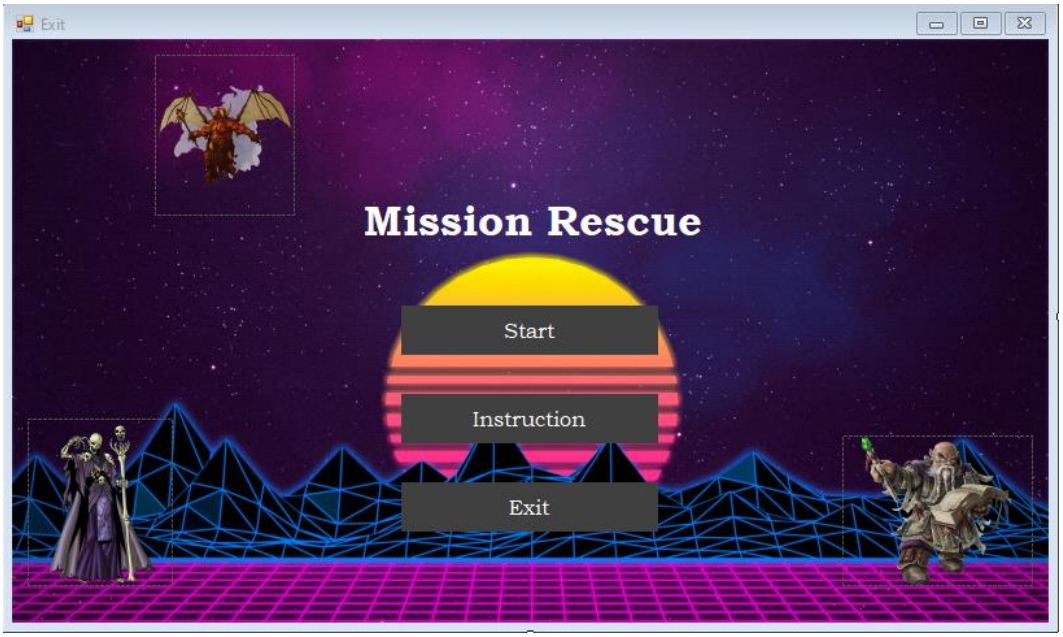
6.1. Image of Player:

6.2. Image of King Grime (Wizard):

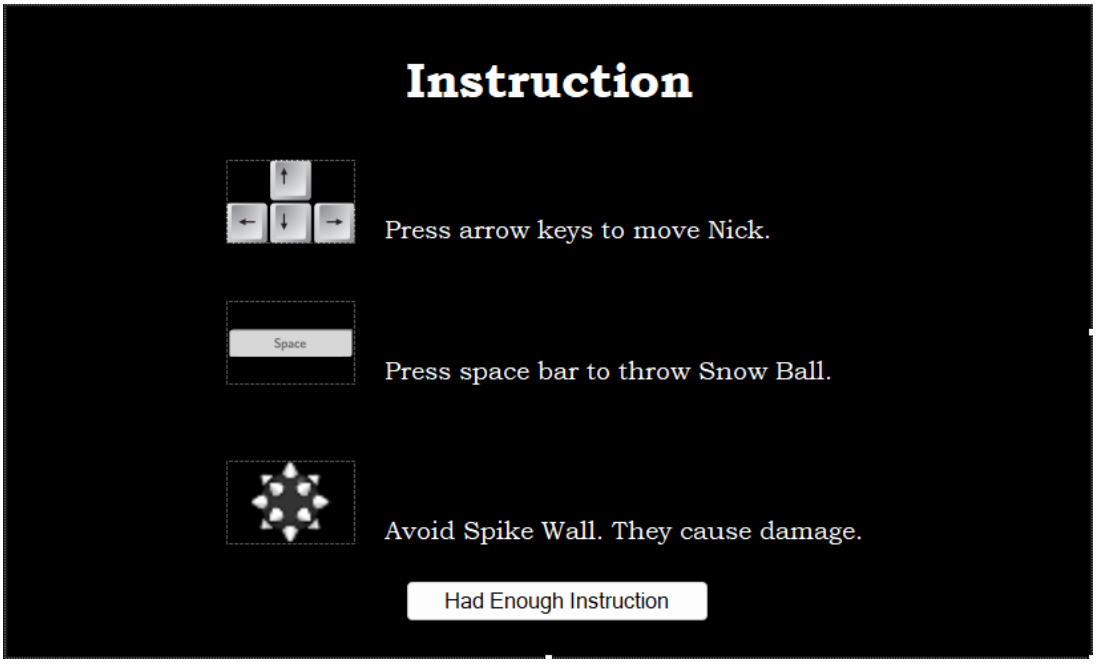
6.3. Image of Flying Beast (Vertical Enemy):

6.4. Image of Skeleton (Random Enemy):

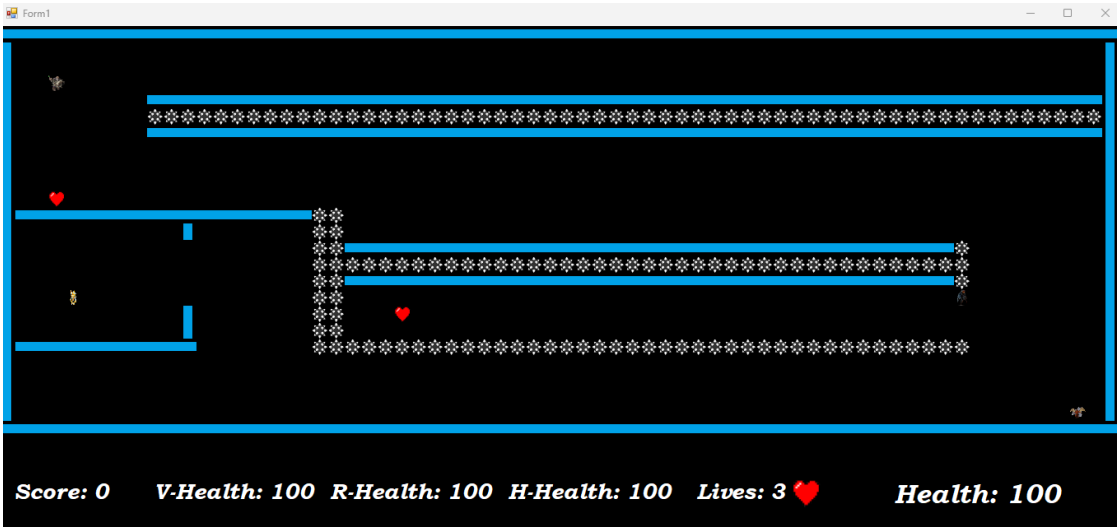
6.5. Image of Start Menu:



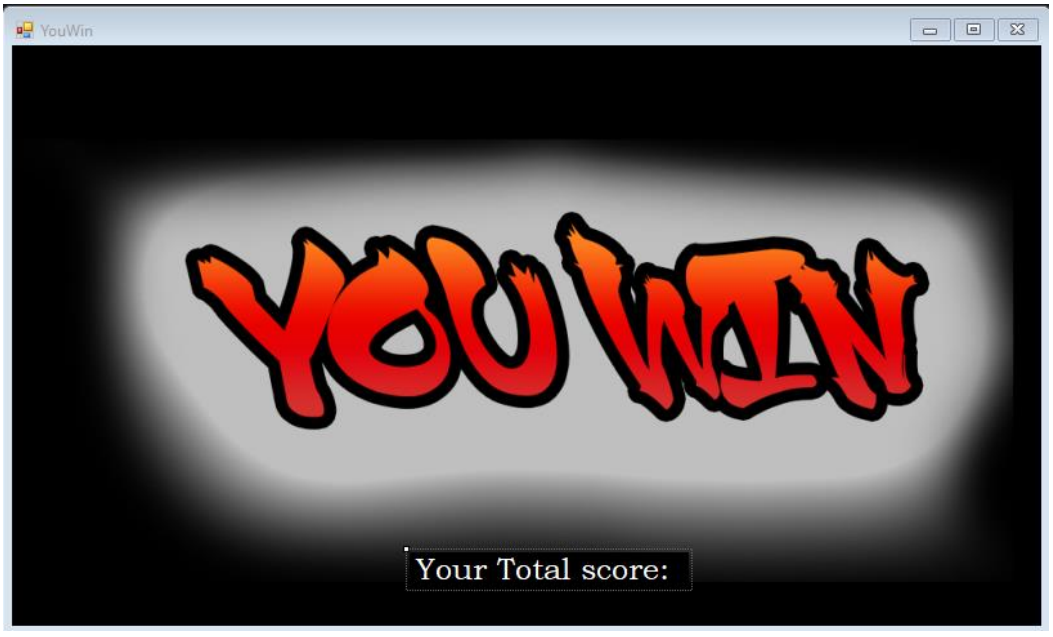
6.6. Image of Instruction Screen:



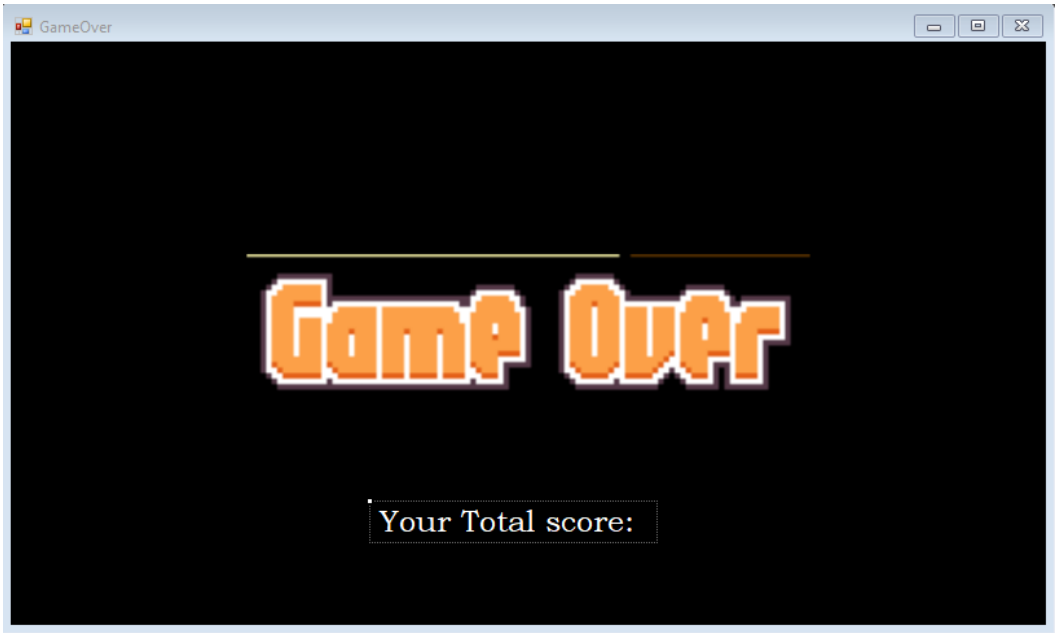
6.7. Image of Game:



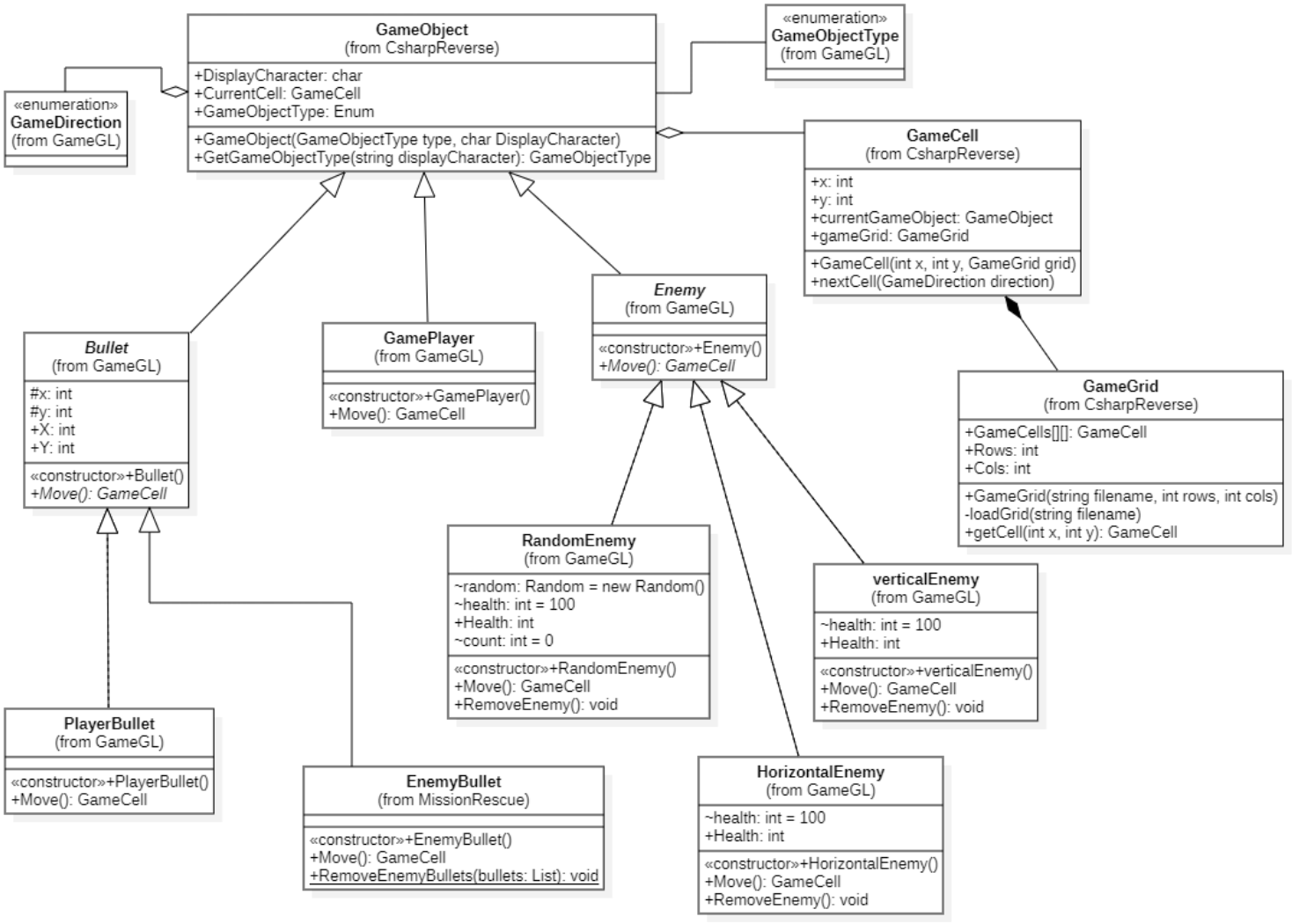
6.8. Image of You Win Screen:



6.9. Image of Game over Screen:



7. Class, Responsibility and Collaboration Diagram (CRC):



8. Code of the Game:

8.1. Game Grid Class:

```
public class GameGrid
  {
     private int rows;
     private int columns;
     private GameCell[,] gameCells;

     public GameGrid(string filePath, int rows, int columns)
     {
        this.rows = rows;
        this.columns = columns;
        gameCells = new GameCell[rows, columns];
        LoadGrid(filePath);
     }

     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.columns; col++)
            {
                GameCell cell = new GameCell(row, col, this);
                char displayCharacter = record[col];
                GameObjectType type = GameObject.GetGameObjectType(displayCharacter);
                Image displayIamge = Game.getGameObjectImage(displayCharacter);
                GameObject gameObject = new GameObject(type, displayIamge);
                cell.SetGameObject(gameObject);
                gameCells[row, col] = cell;
            }
        }

        fp.Close();
    }

    public GameCell GetCell(int x, int y)
    {
        GameCell cell;
        cell = gameCells[x, y];
        return cell;
    }

    public int Rows { get => rows; set => rows = value; }
    public int Columns { get => columns; set => columns = value; }
    public GameCell[,] GameCells { get => gameCells; }
}
```

8.2. Game Cell Class:

```
public class GameCell
  {
     private int x;
     private int y;
     private GameObject currentGameObject;
     private GameGrid gameGrid;
     private const int width = 20;
     private const int height = 20;
     PictureBox cellPicture = new PictureBox();

     public GameCell(int x, int y, GameGrid gameGrid)
     {
        this.x = x;
        this.y = y;
        this.cellPicture = new PictureBox();
        cellPicture.Left = y * width;
        cellPicture.Top = x * height;
        cellPicture.Size = new Size(width, height);
        cellPicture.SizeMode = PictureBoxSizeMode.Zoom;
        cellPicture.BackColor = Color.Transparent;
        this.gameGrid = gameGrid;
     }
     public void SetGameObject(GameObject gameObject)
     {
        this.currentGameObject = gameObject;
```

```
    cellPicture.Image = gameObject.Image;
}

public GameCell NextCell(GameDirection direction)
{
    if (direction == GameDirection.LEFT)
    {
        if (y > 0)
        {
            GameCell nCell = gameGrid.GetCell(x, y - 1);
            return nCell;
        }
    }
    else if (direction == GameDirection.RIGHT)
    {
        if (y < gameGrid.Columns - 1)
        {
            GameCell nCell = gameGrid.GetCell(x, y + 1);
            return nCell;
        }
    }
    else if (direction == GameDirection.UP)
    {
        if (x > 0)
        {
            GameCell nCell = gameGrid.GetCell(x - 1, y);
            return nCell;
        }
```

```
            }
            else if (direction == GameDirection.DOWN)
            {
                if (x < gameGrid.Rows - 1)
                {
                    GameCell nCell = gameGrid.GetCell(x + 1, y);
                    return nCell;
                }
            }

            return this;
        }

        public int X { get => x; set => x = value; }
        public int Y { get => y; set => y = value; }
        public GameObject CurrentGameObject { get => currentGameObject; set =>
currentGameObject = value; }
        public PictureBox PictureBox { get => cellPicture; set => cellPicture = value;
}
    }
```

8.3. Game Object Class:

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

```
        public GameDirection Direction;


        public GameObject(GameObjectType type, Image image)
        {
            this.gameObjectType = type;
            this.image = image;
        }
        public GameObject(GameObjectType type, Image image, GameCell
startCell)
        {
            this.gameObjectType = type;
            this.image = image;
            this.currentCell = startCell;
            currentCell.SetGameObject(this);
        }
        public GameObject(GameObjectType gameObjectType, char
displayCharacter)
        {
            this.displayCharacter = displayCharacter;
            this.gameObjectType = gameObjectType;
        }
        public GameObject(char displayCharacter)
        {
            this.displayCharacter = displayCharacter;
            this.gameObjectType                                    =
GameObject.GetGameObjectType(displayCharacter);
        }


        public GameObject(GameObjectType gameObjectType, char
displayCharacter, GameCell currentCell)
```

```
        {
            this.displayCharacter = displayCharacter;
            this.gameObjectType = gameObjectType;
            this.currentCell = currentCell;
        }
        public static GameObjectType GetGameObjectType(char  displayCharacter)
        {
            if (displayCharacter == '|' || displayCharacter == '%' || displayCharacter == '#' || displayCharacter == '-' || displayCharacter == '_')
            {
                return GameObjectType.WALL;
            }
            else if (displayCharacter == '.')
            {
                return GameObjectType.REWARD;
            }

            else if (displayCharacter == 'P')
            {
                return GameObjectType.PLAYER;
            }

            else if(displayCharacter == '*' || displayCharacter == '<' || displayCharacter == '>')
            {
                return GameObjectType.SPIKE;
            }

            else if(displayCharacter == '3' || displayCharacter == 3)
```

```
            {
                return GameObjectType.HEART;
            }

            return GameObjectType.NONE;
        }

        public Image Image { get { return image; } set { this.image = value; } }
        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);

            }
        }
    }
```

8.4. Game Class:

```
public class Game
  {
    public static GameObject getBlankGameObject()
    {
```

```
            GameObject blankGameObject = new
GameObject(GameObjectType.NONE,
MissionRescue.Properties.Resources.simplebox);
            return blankGameObject;
        }

        public static Image getGameObjectImage(char displayCharacter)
        {
            Image img = MissionRescue.Properties.Resources.simplebox;
            if (displayCharacter == '|' || displayCharacter == '%' )
            {
                img = MissionRescue.Properties.Resources.vertical;
            }

            if (displayCharacter == '-' || displayCharacter == '#')
            {
                img = MissionRescue.Properties.Resources.horizontal;
            }

            if (displayCharacter == '3')
            {
                img = MissionRescue.Properties.Resources.heart;
            }
            if (displayCharacter == 'P' || displayCharacter == 'p')
            {
                img = MissionRescue.Properties.Resources.hero;
            }
            if(displayCharacter == '<' || displayCharacter == '>' || displayCharacter ==
'*')
```

```
            {
                img = MissionRescue.Properties.Resources.spikeWall;
            }
            if(displayCharacter == 'f')
            {
                img = MissionRescue.Properties.Resources.fire;
            }
            return img;
        }

        public static Image getGameObjectImage(GameObjectType type)
        {
            Image img = MissionRescue.Properties.Resources.simplebox;
            if (type == GameObjectType.WALL)
            {
                img = MissionRescue.Properties.Resources.horizontal;
            }

            return img;
        }
    }
```

8.5. Game Player Class:

```
public class GamePlayer : GameObject
  {

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

```
        }

        public GameCell Move()
        {
            GameCell nextCell = this.CurrentCell.NextCell(Direction);
            if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.ENEMY)
            {
                return null;
            }

            else if (nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.WALL && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.SPIKE)
            {
                if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
                {
                    ((Form1)Program.currentForm).ScoreUpdation();
                }
                if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.HEART)
                {
                    ((Form1)Program.currentForm).nickLives++;
                    ((Form1)Program.currentForm).NickLivesDisplay();

                }
                CurrentCell.SetGameObject(Game.getBlankGameObject());
                CurrentCell = nextCell;
                return nextCell;
```

```
            }
            if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.SPIKE)
            {
               ((Form1)Program.currentForm).NickHealth(1);
            }
            return this.CurrentCell;
        }
    }
```

8.6. Enemy Class:

```
public abstract class Enemy : GameObject
    {

        public GameObject previousObject;
        public Enemy(Image img, GameCell currentCell) :
base(GameObjectType.ENEMY, img, currentCell)
        {
            previousObject = Game.getBlankGameObject();
        }

        public abstract GameCell Move();
    }
```

8.7. Horizontal Enemy Class:

```
public class HorizontalEnemy : Enemy
    {
        int health = 100;
        public int Health { get { return health; } set { health = value; } }
```

```
        public HorizontalEnemy(GameCell currentCell) :
base(MissionRescue.Properties.Resources.hEnemy, currentCell)
        {
            Direction = GameDirection.LEFT;
        }

        public override GameCell Move()
        {
            GameCell nextCell = this.CurrentCell.NextCell(Direction);
            if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.PLAYER)
            {
                return null;
            }
            else if (nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.WALL && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.SPIKE && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.HEART)
            {
                CurrentCell.SetGameObject(previousObject);
                GameObjectType gameObjectType =
nextCell.CurrentGameObject.GameObjectType;
                previousObject = new GameObject(gameObjectType ,
Game.getGameObjectImage(gameObjectType));
                CurrentCell = nextCell;
                return nextCell;
            }
            else
            {
                if (Direction == GameDirection.LEFT)
                {
```

```
                Direction = GameDirection.RIGHT;
            }
            else
            {
                Direction = GameDirection.LEFT;
            }
        }
        return CurrentCell;
    }

    public void RemoveEnemy()
    {
        CurrentCell.SetGameObject(Game.getBlankGameObject());
    }
}
```

8.8. Vertical Enemy Class:

```
public class verticalEnemy : Enemy
  {
    int health = 100;
    public int Health { get { return health; } set { health = value; } }
    public verticalEnemy(GameCell current) :
base(MissionRescue.Properties.Resources.vEnemy, current)
    {
      this.Direction = GameDirection.UP;
    }
    public override GameCell Move()
    {
```

```
        GameCell nextCell = this.CurrentCell.NextCell(Direction);
        if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.PLAYER)
        {
            return null;
        }
        else if (nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.WALL && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.SPIKE && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.HEART)
        {
            CurrentCell.SetGameObject(previousObject);
            GameObjectType gameObjectType =
nextCell.CurrentGameObject.GameObjectType;
            previousObject = new GameObject(gameObjectType,
Game.getGameObjectImage(gameObjectType));
            CurrentCell = nextCell;
            return nextCell;
        }
        else
        {
            if (this.Direction == GameDirection.UP)
            {
                this.Direction = GameDirection.DOWN;
            }
            else
            {
                this.Direction = GameDirection.UP;
            }
            return CurrentCell;
```

```
            }



        }

        public void RemoveEnemy()
        {
            CurrentCell.SetGameObject(Game.getBlankGameObject());
        }

    }
```

8.9. Random Enemy Class:

```
public class RandomEnemy : Enemy
  {
        Random random = new Random();
        int health = 100;
        public int Health { get { return health; } set { health = value; } }
        int count = 0;

        public RandomEnemy(GameCell currentCell) :
base(MissionRescue.Properties.Resources.rEnemy, currentCell)
        {
            this.Direction = GameDirection.RIGHT;
        }

        public override GameCell Move()
        {
```

```
GameCell nextCell = this.CurrentCell.NextCell(Direction);
int number = random.Next(4);
if (count == 2)
{
   count = 0;
   switch (number)
   {
      case 0:
         this.Direction = GameDirection.RIGHT;
         break;
      case 1:
         this.Direction = GameDirection.LEFT;
         break;
      case 2:
         this.Direction = GameDirection.UP;
         break;
      case 3:
         this.Direction = GameDirection.DOWN;
         break;
   }
}
else
{
   count++;
}
if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.PLAYER)
{
   return null;
```

```
            }
            else if (nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.WALL && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.SPIKE && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.HEART)
            {
                CurrentCell.SetGameObject(previousObject);
                GameObjectType gameObjectType =
nextCell.CurrentGameObject.GameObjectType;
                previousObject = new GameObject(gameObjectType,
Game.getGameObjectImage(gameObjectType));
                CurrentCell = nextCell;
                return nextCell;
            }
            else
            {
                return this.CurrentCell;
            }
        }

        public void RemoveEnemy()
        {
            CurrentCell.SetGameObject(Game.getBlankGameObject());
        }
    }
```

8.10. Bullet Class:

```
public abstract class Bullet : GameObject
    {
        protected int x;
```

```
        protected int y;


        public int X { get { return this.x; } set { this.x = value; } }
        public int Y { get { return this.y; } set { this.y = value; } }


        public Bullet(GameCell startCell,Image img ) :
base(GameObjectType.BULLET, img, startCell)
        {
            this.x = startCell.X;
            this.Y = startCell.Y;
        }



        public abstract GameCell Move();


    }
```

8.11. Player Bullet Class:

```
public class PlayerBullet : Bullet
    {
        public PlayerBullet(GameCell startCell) : base(startCell ,
MissionRescue.Properties.Resources.snowBall)
        {
            this.x = startCell.X;
            this.Y = startCell.Y;
        }
```

```
public override GameCell Move()
{
    GameCell nextCell = this.CurrentCell.NextCell(this.Direction);
    if (nextCell.CurrentGameObject.GameObjectType != GameObjectType.WALL && nextCell.CurrentGameObject.GameObjectType != GameObjectType.SPIKE && nextCell.CurrentGameObject.GameObjectType != GameObjectType.HEART)
    {
        // detect enemies
        if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY)
        {
            if (nextCell.CurrentGameObject is RandomEnemy)
            {

                ((Form1)Program.currentForm).REnemyHealth();
            }
            if (nextCell.CurrentGameObject is HorizontalEnemy)
            {

                ((Form1)Program.currentForm).HEnemyHealth();
            }

            if (nextCell.CurrentGameObject is verticalEnemy)
            {

                ((Form1)Program.currentForm).VEnemyHealth();
            }

            CurrentCell.SetGameObject(Game.getBlankGameObject());
```

```
                return null;
            }

            //Move Bullets
            CurrentCell.SetGameObject(Game.getBlankGameObject());
            this.CurrentCell = nextCell;
            return nextCell;

        }
        //Detect Wall
        else if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.WALL || nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.SPIKE)
        {
            CurrentCell.SetGameObject(Game.getBlankGameObject());
            return null;
        }
        return this.CurrentCell;
    }
  }
```

8.12. Enemy Bullet Class:

```
public class EnemyBullet : Bullet
  {
    public EnemyBullet(GameCell startCell) : base( startCell,
MissionRescue.Properties.Resources.fireBall)
    {
      this.x = startCell.X;
```

```
        this.Y = startCell.Y;
    }

    public override GameCell Move()
    {
        GameCell nextCell = this.CurrentCell.NextCell(this.Direction);
        if (nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.WALL && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.SPIKE && nextCell.CurrentGameObject.GameObjectType !=
GameObjectType.HEART)
        {

            if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.PLAYER)
            {
                CurrentCell.SetGameObject(Game.getBlankGameObject());
                ((Form1)Program.currentForm).NickHealth(5);
                return null;
            }
            //Move Bullets
            CurrentCell.SetGameObject(Game.getBlankGameObject());
            this.CurrentCell = nextCell;
            return nextCell;

        }
        //Detect Wall
        else if (nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.WALL || nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.SPIKE && nextCell.CurrentGameObject.GameObjectType ==
GameObjectType.HEART)
        {
```

```
            CurrentCell.SetGameObject(Game.getBlankGameObject());
            return null;
        }
        return this.CurrentCell;
    }

    public static void RemoveEnemyBullets(List<EnemyBullet> bullets)
    {
        foreach(EnemyBullet e in bullets)
        {
            e.CurrentCell.SetGameObject(Game.getBlankGameObject());
        }
    }
}
```

8.13. Form1 Class:

```
public partial class Form1 : Form
  {
    GamePlayer nick;
    HorizontalEnemy hEnemy;
    verticalEnemy vEnemy;
    RandomEnemy rEnemy;

    List<Enemy> enemies = new List<Enemy>();
    List<PlayerBullet> nickBullets = new List<PlayerBullet>();
    List<EnemyBullet> hEnemyBullets = new List<EnemyBullet>();
    List<EnemyBullet> vEnemyBullets = new List<EnemyBullet>();
```

```
int enemyTick = 0;
int bulletTick = 0;
int enemyBulletTick = 0;
public int score = 0;
bool isKeyRight, isKeyLeft, isKeyDown, isKeyUp, isKeySpace;

public int nickHealth = 100;
public int nickLives = 3;

public int rEnemyHealth = 100;

GameGrid maze;

public Form1()
{
   InitializeComponent();
   this.KeyDown += Form1_KeyDown;
   this.KeyUp += Form1_KeyUp;
}


private void Form1_Load(object sender, EventArgs e)
{
   maze = new GameGrid("maze.txt", 25, 68);
   Image pacmanImage = Game.getGameObjectImage('P');
   GameCell startCell = maze.GetCell(16, 4);
```

```
    GameCell hEnemyStart = maze.GetCell(3, 51);
    GameCell vEnemyStart = maze.GetCell(9, 65);
    GameCell rEnemyStart = maze.GetCell(18, 44);

    hEnemy = new HorizontalEnemy(hEnemyStart);
    vEnemy = new verticalEnemy(vEnemyStart);
    rEnemy = new RandomEnemy(rEnemyStart);
    nick = new GamePlayer(pacmanImage, startCell);
    enemies.Add(hEnemy);
    enemies.Add(vEnemy);
    enemies.Add(rEnemy);
    printMaze(maze);
}

private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Up)
    {
        isKeyUp = true;
    }
    if (e.KeyCode == Keys.Down)
    {
        isKeyDown = true;
    }
    if (e.KeyCode == Keys.Right)
    {
        isKeyRight = true;
    }
```

```
    if (e.KeyCode == Keys.Left)
    {
        isKeyLeft = true;
    }
    if (e.KeyCode == Keys.Space)
    {
        isKeySpace = true;
    }
}

private void Form1_KeyUp(object sender, KeyEventArgs e)
{
    if (e.KeyCode == Keys.Up)
    {
        isKeyUp = false;
    }
    if (e.KeyCode == Keys.Down)
    {
        isKeyDown = false;
    }
    if (e.KeyCode == Keys.Right)
    {
        isKeyRight = false;
    }
    if (e.KeyCode == Keys.Left)
    {
        isKeyLeft = false;
    }
```

```
    if (e.KeyCode == Keys.Space)
    {
        isKeySpace = false;
    }
}
void printMaze(GameGrid grid)
{
    for (int x = 0; x < grid.Rows; x++)
    {

        for (int y = 0; y < grid.Columns; y++)
        {
            GameCell cell = grid.GetCell(x, y);
            this.Controls.Add(cell.PictureBox);
        }
    }
}

private void Form1_KeyDown_1(object sender, KeyEventArgs e)
{

}

private void gameLoop_Tick_Tick(object sender, EventArgs e)
{

    GameCell nextCell = null;
    if (isKeyLeft)
```

```
{
    nick.Direction = GameDirection.LEFT;

    nextCell = nick.Move();
    DetectCollision(nextCell);
}
if (isKeyRight)
{

    nick.Direction = GameDirection.RIGHT;
    nextCell = nick.Move();
    DetectCollision(nextCell);
}
if (isKeyUp)
{
    nick.Direction = GameDirection.UP;
    nextCell = nick.Move();
    DetectCollision(nextCell);
}
if (isKeyDown)
{
    nick.Direction = GameDirection.DOWN;
    nextCell = nick.Move();
    DetectCollision(nextCell);
}
if (isKeySpace && bulletTick >= 3)
{
    int x = nick.CurrentCell.X;
```

```
    int y = nick.CurrentCell.Y;
    if(y+2 < 67)
    {
        GameCell startCell = maze.GetCell(x, y + 2);
        PlayerBullet b = new PlayerBullet(startCell);
        nickBullets.Add(b);
    }

    bulletTick = 0;
}
//Move player bullets
for (int i = nickBullets.Count - 1; i >= 0; i--)
{
    PlayerBullet b = nickBullets[i];
    b.Direction = GameDirection.RIGHT;
    GameCell c = b.Move();
    if (c == null)
    {
        nickBullets.RemoveAt(i);
    }
}

if (enemyTick == 5)
{
    foreach (Enemy enemy in enemies)
    {
        DetectCollision(enemy.Move());
    }
```

```
            enemyTick = 0;
        }

        if(enemies.Count == 0)
        {
            gameLoop.Stop();
            YouWin win = new YouWin();
            win.Show();
            this.Hide();

        }

        //Generate Enemy Bullets
        if(enemyBulletTick > 5)
        {
            //Horizontal Enemy Bullets Generation
            if (CalculateDistance(hEnemy.CurrentCell.X, hEnemy.CurrentCell.Y) <
50 && hEnemy.Health > 0)
            {
                int x = hEnemy.CurrentCell.X;
                int y = hEnemy.CurrentCell.Y;
                if(y-5 > 2)
                {
                    GameCell startCell = maze.GetCell(x, y - 1);
                    EnemyBullet bullet = new EnemyBullet(startCell);

                    hEnemyBullets.Add(bullet);
                }
            }
```

```
            else if (hEnemy.Health < 0)
            {
                EnemyBullet.RemoveEnemyBullets(hEnemyBullets);
            }

            //Vertical Enemy Bullets Generation
            if (CalculateDistance(vEnemy.CurrentCell.X, vEnemy.CurrentCell.Y) <
50 && vEnemy.Health > 0)
            {
                int x = vEnemy.CurrentCell.X;
                int y = vEnemy.CurrentCell.Y;
                GameCell startCell = maze.GetCell(x, y - 2);
                EnemyBullet bullet = new EnemyBullet(startCell);

                vEnemyBullets.Add(bullet);



            }
            else if(vEnemy.Health < 0)
            {
                EnemyBullet.RemoveEnemyBullets(vEnemyBullets);
            }
            enemyBulletTick = 0;
        }
        //move enemy bullets
        for (int i = vEnemyBullets.Count - 1; i >= 0; i--)
        {
            EnemyBullet b = vEnemyBullets[i];
            b.Direction = GameDirection.LEFT;
```

```
            GameCell c = b.Move();
            if (c == null)
            {
                vEnemyBullets.RemoveAt(i);
            }
        }
        for (int i = hEnemyBullets.Count - 1; i >= 0; i--)
        {
            EnemyBullet b = hEnemyBullets[i];
            b.Direction = GameDirection.LEFT;
            GameCell c = b.Move();
            if (c == null)
            {
                hEnemyBullets.RemoveAt(i);
            }
        }

        enemyTick++;
        bulletTick++;
        enemyBulletTick++;
    }

    public double CalculateDistance(int enemyX, int enemyY)
    {
        int nickX = nick.CurrentCell.X;
        int nickY = nick.CurrentCell.Y;

        int dx = enemyX - nickX;
```

```
    int dy = enemyY - nickY;

    return Math.Sqrt(dx * dx + dy * dy);
}

private void DetectCollision(GameCell nextCell)
{
    if (nextCell == null)
    {

        NickLiveDecrement();
    }
    if (nickLives <= 0)
    {
        gameLoop.Stop();
        GameOver over = new GameOver();
        over.Show();
        this.Hide();
    }
}

public void NickLivesDisplay()
{
    livesLabel.Text = "Lives: " + nickLives;
}
public void NickLiveDecrement()
{
    nickHealth = 100;
```

```
    nickLives--;
    NickLivesDisplay();
    nick.CurrentCell.X = 16;
    nick.CurrentCell.Y = 4;
}

public void NickHealth(int decrement)
{
    nickHealth -= decrement;
    if (nickHealth <= 0)
    {
        NickLiveDecrement();
    }
    healthLabel.Text = "Health: " + nickHealth;
}

public void REnemyHealth()
{
    rEnemy.Health-=5;
    ScoreUpdation();
    if(rEnemy.Health <= 0)
    {
        rEnemy.RemoveEnemy();
        enemies.Remove(rEnemy);
    }
    rEnemyHealthL.Text = "R-Health: " + rEnemy.Health;
}
public void VEnemyHealth()
```

```
        {
            vEnemy.Health -= 5;
            ScoreUpdation();
            if (vEnemy.Health <= 0)
            {
                vEnemy.RemoveEnemy();
                enemies.Remove(vEnemy);
            }
            vEnemyHealthL.Text = "V-Health: " + vEnemy.Health;
        }
        public void HEnemyHealth()
        {
            hEnemy.Health -= 5;
            ScoreUpdation();
            if (hEnemy.Health <= 0)
            {
                hEnemy.RemoveEnemy();
                enemies.Remove(hEnemy);
            }
            hEnemyHealthL.Text = "H-Health: " + hEnemy.Health;
        }
        public void ScoreUpdation()
        {

            score++;
            ScoreLabel.Text = "Score:" + score;
        }
    }
```

8.14. Game Object Type (ENUM):

```
public enum GameObjectType
  {
    WALL,
    SPIKE,
    PLAYER,
    ENEMY,
    REWARD,
    HEART,
    BULLET,
    NONE
  }
```

8.15. Game Direction (ENUM):

```
public enum GameDirection
  {
    LEFT,
    RIGHT,
    UP,
    DOWN,
    NONE
  }
```