The Prince Archer Game

Session: 2022 – 2026

Submitted by:

Abdul Rehman 2022-CS-79

Supervised by:

Ma’am Maida Shahid

Sir Laeeq Khan Niaz

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

1. Short Description and Story of Game:

In a kingdom long ago, there lived a Prince skilled in archery named Alexander. One day, the kingdom was invaded by an army of evil creatures led by Lady Night and Lord shadow.

Prince found himself facing a daunting task - he was the only one left to defend the kingdom. So, he gathered information about the enemy and comes to know that the Lady Night has known some magic and the Lord Shadow is always with its twin bodyguards named Blade and Devi.

Despite the overwhelming odds, Prince refused to back down. With his bow and arrow, he stood tall and prepared for battle. As the enemy charged towards him, Prince calmly took aim and let fly his arrows, taking down one enemy after another.

With every arrow he lost, Prince grew more confident in his abilities. His movements became fluid and graceful, and soon he was dodging incoming attacks with ease. The battle continued for hours, but Prince refused to give up. Finally, the Lord Shadow fell before his arrows. The kingdom was saved, and Prince emerged victorious, hailed as a hero by the people.

2. Game Characters Description:

2.1. Player:

There is one human player in the game.

2.1.1. Prince Alexander:

Prince is the main character in the game and is known for his archery skills. He is the hero of the game, admired for his bravery, courage, and determination in the face of danger.

2.2. Enemies:

There is only one enemy in the game till now.

2.2.1. Lord Shadow:

Lord Shadow is the leader of the evil creature's group and is the husband of the Lady Night. He is the Strongest in the group and is very skilled in shooting.

3. Game Objects Description:

Following are the objects used in the game.

3.1. Dragon Fruit:

Fruits in the game work as an energizer. It can restore 20% of the prince's energy. It appears randomly at 2 places when Prince's health is 30% or below and enemy health is 50% or less.

3.2. Walls:

Walls are the barriers in the game which The Prince and the enemies cannot cross. In game these are represented by **"Bricks"**.

3.3. Bombs:

Bombs in the game are used by the Lord Shadow. It can damage the prince and drop his health by 20 %. It appears randomly on the screen when the Lord's health is 60% or below.

3.4. Doors:

The game has 2 castle doors. The 1st door opens when the prince comes near to door. And the 2nd door opens when the Lord Shadow is dead.

3.5. Bow and Arrows:

Bow and Arrows in the game are used by the prince to fight against the crucial enemies of their time.

3.6. Gun and Bullets:

Guns and Bullets in the game are used by the Lord Shadow to fight against Prince Alexander, who's very skilled in archery and knows many tactics.

4. Rules & Interactions:

Following are the rules and Interactions of the game:

- Prince loses health whenever he is being hit by the bullet or steps on bomb.
- Prince gains energy and score points whenever he eats dragon fruits.
- The score is increased when the arrows hit the enemy and the enemy loses its health.
- Prince also loses some of its score whenever he encounters the enemy and the bomb.
- Prince must defeat enemy to proceed.

5. Goal of the Game:

The Goal of the game is to defeat the evil Lord and save the kingdom of Prince. And make yourself ready for the upcoming challenges.

6. Wireframes of the Game:

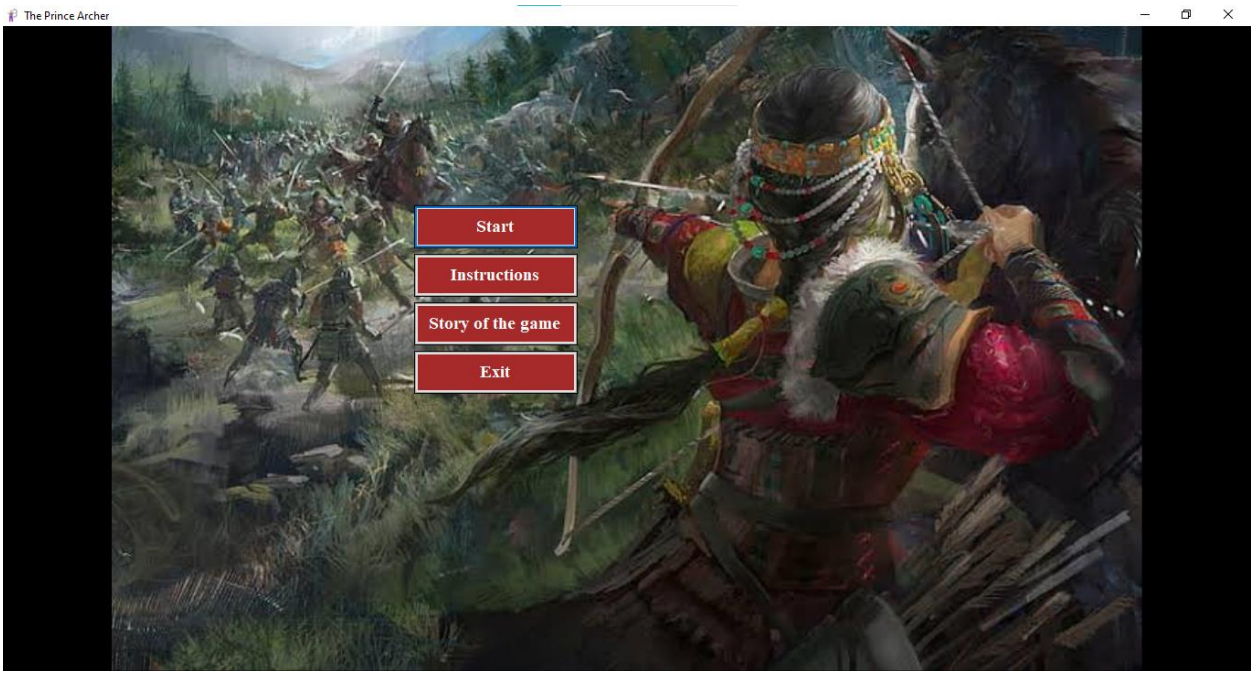


Fig.: Main Menu

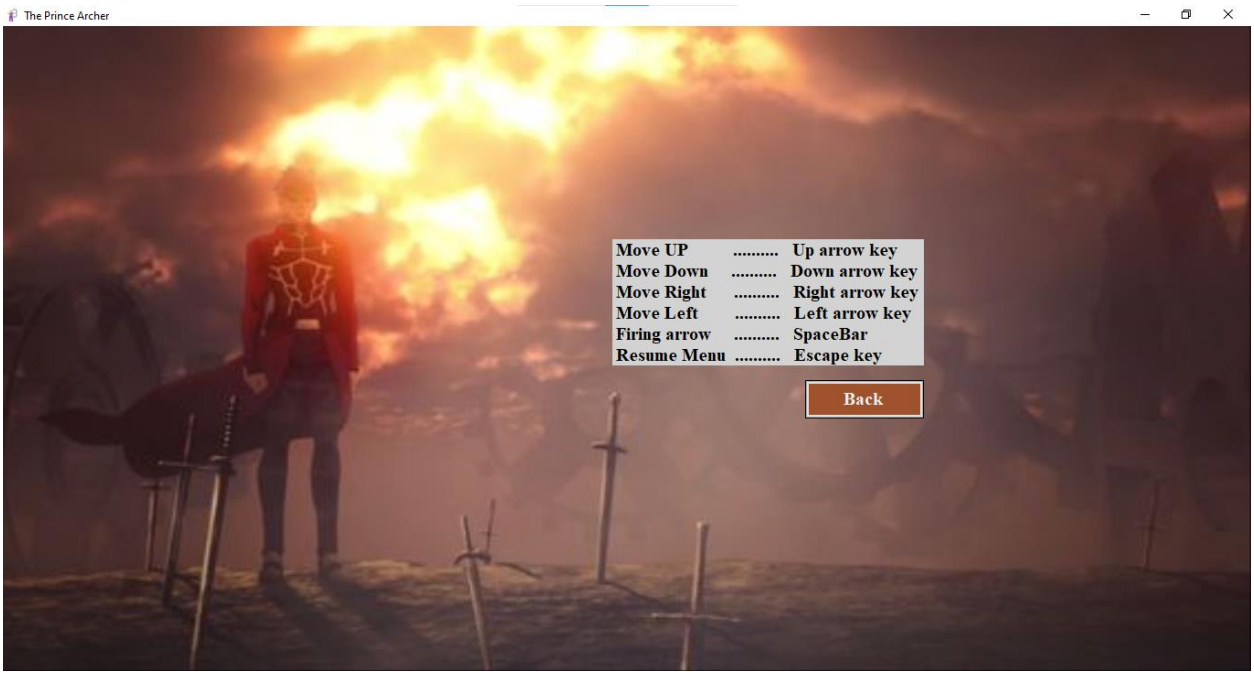


Fig.: Controller Page

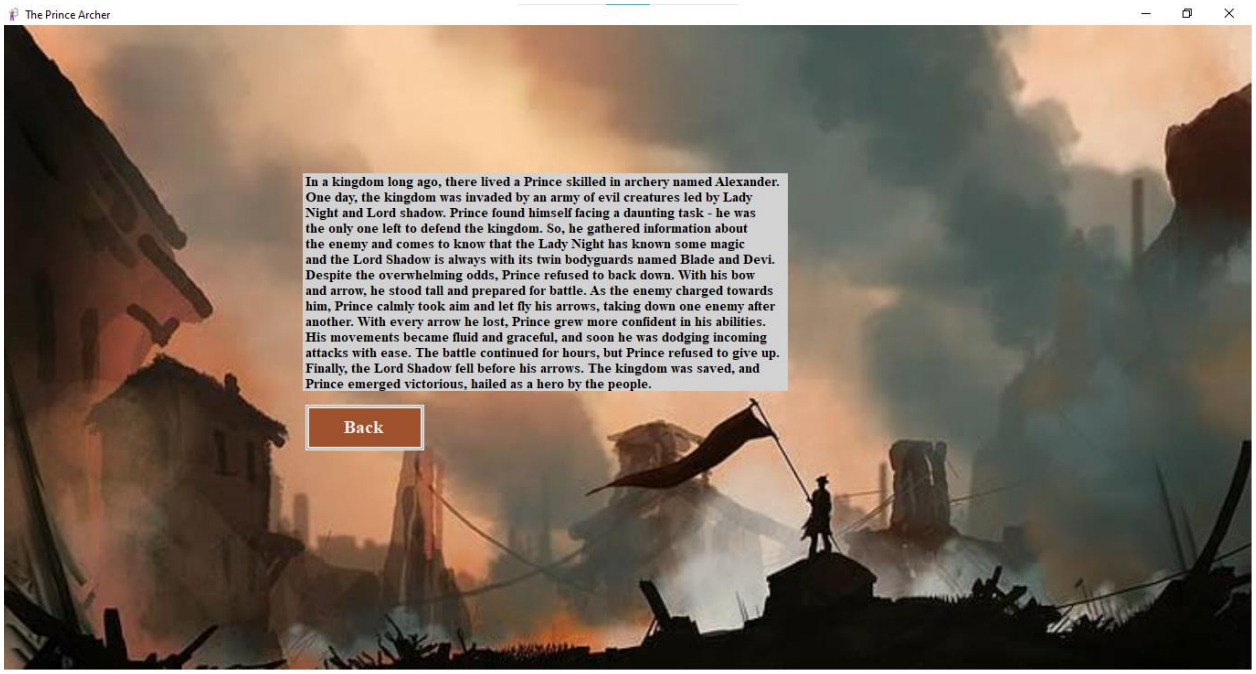


Fig.: Story of the Game

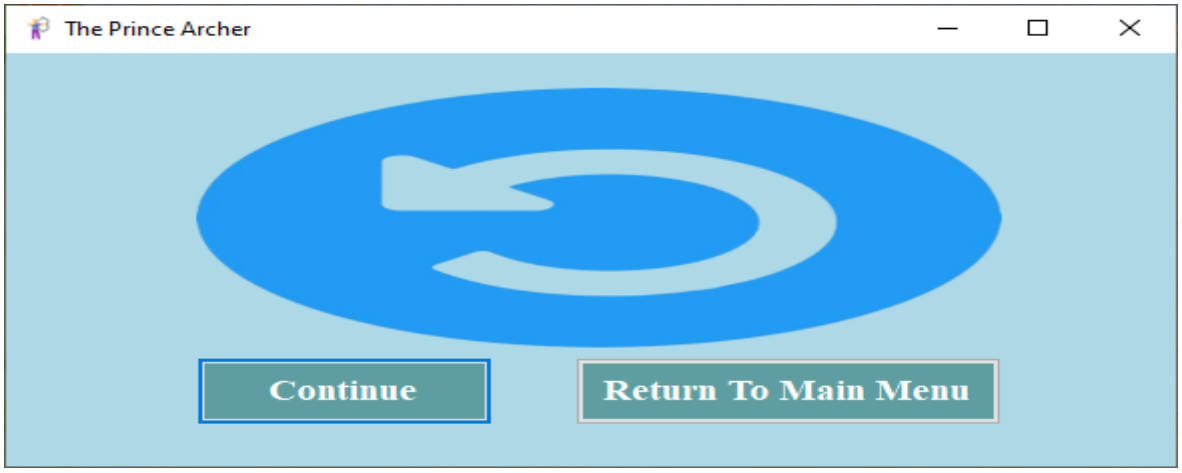


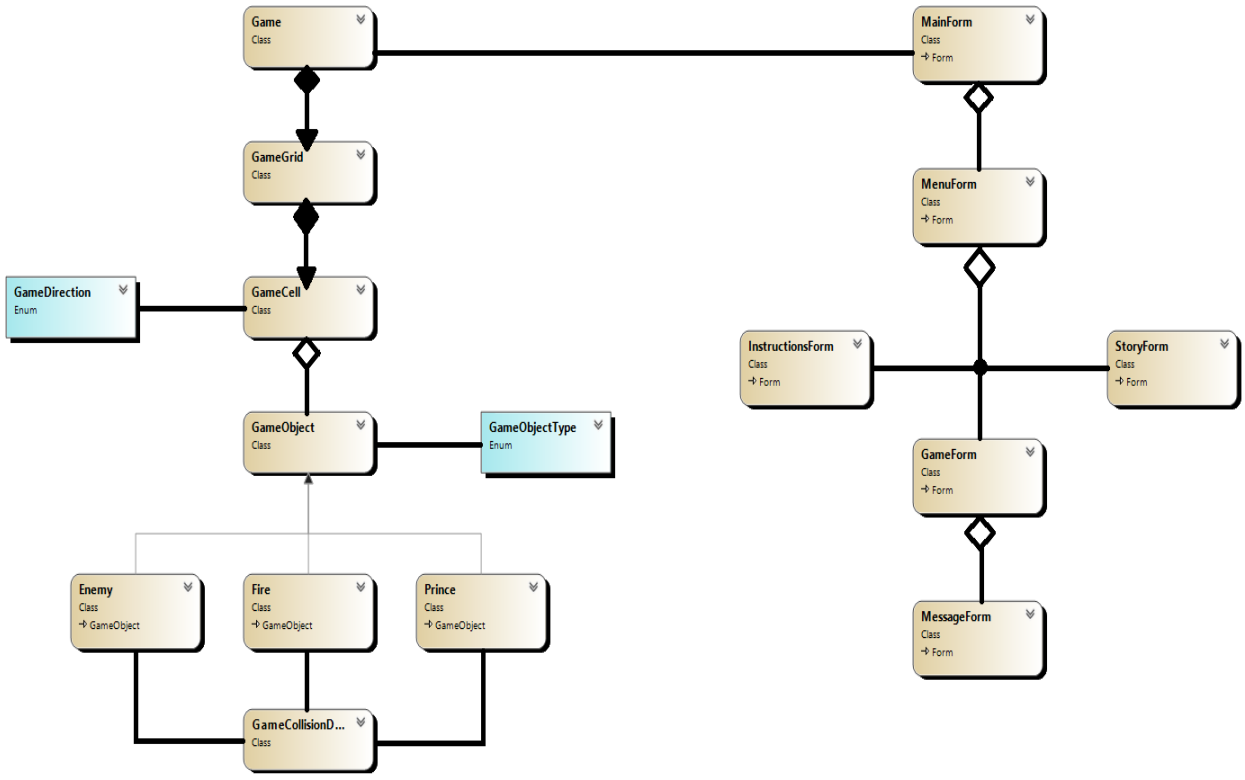
Fig.: Pause Menu

Fig.: Lose Screen

Fig.: Win Screen

Fig.: Game Screen

7. Class Diagram:



8. Complete Code of the Game:

8.1. Game Logic (GL) classes:

- **Game Class:**

```
public class Game
    {
        GameGrid grid;
        Panel game_Panel;
        Prince prince;
        Enemy enemy;
        List<Fire> prince_Fires;
        List<Fire> enemy_Fires;

        public Enemy Villian { get => enemy; set => enemy = value; }
        public GameGrid Grid { get => grid; set => grid = value; }
        public List<Fire> HeroFires { get => prince_Fires; set =>
prince_Fires = value; }
        public List<Fire> EnemyFires { get => enemy_Fires; set =>
enemy_Fires = value; }


        public Game(Panel panel)
        {
            this.game_Panel = panel;
            Grid = new GameGrid(13, 37);
            prince = new
Prince(ThePrinceArcher_Game.Properties.Resources.archer, Grid.GetCell(5,
1));
            prince_Fires = new List<Fire>();
            enemy_Fires = new List<Fire>();
            PrintMaze(Grid);
        }

        public void EnemyPresence()
        {
            enemy = new
Enemy(ThePrinceArcher_Game.Properties.Resources.enemy, Grid.GetCell(5,
25));
        }

        public void RemoveHeroFire()
        {
```

```
            for (int i = 0; i < prince_Fires.Count; i++)
            {
                if (prince_Fires[i].X)
                {
prince_Fires[i].CurrentCell.SetGameObject(GetBlankGameObject());
                    prince_Fires.RemoveAt(i);
                }
            }
        }

        public void RemoveEnemyFire()
        {
            for (int i = 0; i < enemy_Fires.Count; i++)
            {
                if (enemy_Fires[i].X)
                {
enemy_Fires[i].CurrentCell.SetGameObject(GetBlankGameObject());
                    enemy_Fires.RemoveAt(i);
                }
            }
        }

        public int GetHealth()
        {
            return prince.Health;
        }

        public GameCell GetCell(int x, int y)
        {
            return Grid.GetCell(x, y);
        }

        public Prince GetPrince()
        {
            return this.prince;
        }

        public Enemy GetEnemy()
        {
            return this.enemy;
        }

        public void AddHeroFire(Fire f)
        {
            prince_Fires.Add(f);
        }

        public void AddEnemyFire(Fire s)
```

```
{
    enemy_Fires.Add(s);
}

public void DecreaseHeroHealth()
{
    prince.Health -= 10;
}

public void RestoreHeroHealth()
{
    prince.Health += 20;
}

public int GetPrinceHealth()
{
    return prince.Health;
}

public void DecreaseEnemyHealth()
{
    enemy.Health -= 10;
}

public int GetEnemyHealth()
{
    return enemy.Health;
}



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);
            game_Panel.Controls.Add(cell.PictureBox);
        }
    }
}



public static GameObject GetBlankGameObject()
{
    return new GameObject(GameObjectType.NONE, null);
}
```

```
        public GameObject GetWallGameObject()
        {
            return new GameObject(GameObjectType.WALL,
ThePrinceArcher_Game.Properties.Resources.gate);
        }

        public GameObject GetBombGameObject()
        {
            return new GameObject(GameObjectType.BOMB,
ThePrinceArcher_Game.Properties.Resources.bomb);
        }

        public GameObject GetFruitGameObject()
        {
            return new GameObject(GameObjectType.REWARD,
ThePrinceArcher_Game.Properties.Resources.fruit);
        }

        public static Image GetGameObjectImage(char displayCharacter)
        {
            Image img = null;
            if (displayCharacter == '#')
            {
                img = ThePrinceArcher_Game.Properties.Resources.bricks;
            }
            else if (displayCharacter == '|')
            {
                img = ThePrinceArcher_Game.Properties.Resources.gate;
            }
            else if (displayCharacter == '_')
            {
                img =
ThePrinceArcher_Game.Properties.Resources.steel_gate;
            }
            return img;
        }



        int bombs = 0;
        public void CreateBombs()
        {
            if (enemy.Health <= 60 && bombs < 5)
            {
                Random rnd = new Random();
                int x, y;
                x = 5 + rnd.Next(1, 17);
                y = 3 + rnd.Next(1, 7);
                GameCell cell = grid.GetCell(y, x);
                cell.SetGameObject(GetBombGameObject());
```

```
                bombs++;
            }
        }

        int fruits = 0;
        public void GenerateFruit(int points)
        {
            if(prince.Health <= 30 && points >= 25 && fruits < 2)
            {
                Random rnd = new Random();
                int x, y;
                x = 5 + rnd.Next(1, 17);
                y = 3 + rnd.Next(1, 7);
                GameCell cell = grid.GetCell(y, x);
                cell.SetGameObject(GetFruitGameObject());
                fruits++;
            }
        }


        public void NextCellFunctions(GameCell ncell,
GameCollisionDetector collider, ref int points)
        {
            if (collider.PrinceWithBomb(ncell))
            {
                DecreaseHeroHealth();
                points -= 5;
                bombs--;
            }

            if (collider.PrinceWithEnemy(ncell))
            {
                DecreaseHeroHealth();
                points -= 5;
            }

            if(collider.PrinceWithFruit(ncell))
            {
                RestoreHeroHealth();
                points += 5;
            }
        }


        public void DecreaseBombNFruit(Fire f)
        {
            if(f.CurrentCell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
            {
                fruits--;
```

```
            }
            if(f.CurrentCell.CurrentGameObject.GameObjectType ==
GameObjectType.BOMB)
            {
                bombs--;
            }
        }

        public bool WinGame()
        {
            bool flag = false;

            GameCell cell1 = grid.GetCell(5, 32);
            GameCell cell2 = grid.GetCell(6, 32);

            if (prince.CurrentCell == cell1 || prince.CurrentCell ==
cell2)
            {
                flag = true;
            }

            return flag;
        }

        public void WallRemove()
        {
            GameCell cell1 = GetCell(5, 4);
            GameCell cell2 = GetCell(6, 4);

            cell1.SetGameObject(GetBlankGameObject());
            cell2.SetGameObject(GetBlankGameObject());
        }

        public void PrintWall()
        {
            GameCell cell1 = GetCell(5, 4);
            GameCell cell2 = GetCell(6, 4);

            cell1.SetGameObject(GetWallGameObject());
            cell2.SetGameObject(GetWallGameObject());
        }

        public void OpenGate()
        {
            GameCell cell1 = GetCell(5, 32);
            GameCell cell2 = GetCell(6, 32);

            cell1.SetGameObject(GetBlankGameObject());
            cell2.SetGameObject(GetBlankGameObject());
        }
```

```
}
```

- **GameCell:**

```
public class GameCell
    {
        int row;
        int col;
        GameObject currentGameObject;
        GameGrid grid;
        PictureBox pictureBox;
        const int width = 35;
        const int height = 40;


        public int X { get => row; set => row = value; }
        public int Y { get => col; set => col = value; }
        public GameObject CurrentGameObject { get => currentGameObject;
}
        public PictureBox PictureBox { get => pictureBox; set =>
pictureBox = value; }
        public GameGrid Grid { get => grid; set => grid = value; }


        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.StretchImage;
            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 (CheckNextCell(ncell))
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.RIGHT)
            {
                if (this.col < Grid.Cols - 1)
                {
                    GameCell ncell = Grid.GetCell(this.row, this.col +
1);
                    if (CheckNextCell(ncell))
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.UP)
            {
                if (this.row > 0)
                {
                    GameCell ncell = Grid.GetCell(this.row - 1,
this.col);
                    if (CheckNextCell(ncell))
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.DOWN)
            {
                if (this.row < Grid.Rows - 1)
                {
                    GameCell ncell = Grid.GetCell(this.row + 1,
this.col);
                    if (CheckNextCell(ncell))
                    {
                        return ncell;
                    }

                }
```

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

        private bool CheckNextCell(GameCell ncell)
        {
            bool flag = false;
            if(ncell.CurrentGameObject.GameObjectType !=
GameObjectType.WALL && ncell.CurrentGameObject.GameObjectType !=
GameObjectType.GATE && ncell.currentGameObject.GameObjectType !=
GameObjectType.GRID)
            {
                flag = true;
            }
            return flag;
        }

    }
```

- **GameGrid:**

```
public class GameGrid
    {
        GameCell[,] cells;
        int rows;
        int cols;

        public GameGrid(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();
        }

        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()
        {
            StreamReader fp = new StreamReader("maze.txt");
            string record;
```

```
            for (int row = 0; row < this.rows; row++)
            {
                record = fp.ReadLine();
                if (record != null)
                {
                    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 displayIamge =
Game.GetGameObjectImage(displayCharacter);
                        GameObject gameObject = new GameObject(type,
displayIamge);
                        cell.SetGameObject(gameObject);
                        cells[row, col] = cell;
                    }
                }
            }
            fp.Close();
        }

    }
```

- **GameObject;**

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

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

        public Image Image { get => image; set => image = value; }

        public GameCell CurrentCell
        {
            get => currentCell;
            set
            {
                currentCell = value;
                currentCell.SetGameObject(this);
            }
        }
```

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

        public static GameObjectType GetGameObjectType(char
displayCharacter)
        {
            GameObjectType type = GameObjectType.NONE;

            if (displayCharacter == '#' )
            {
                type = GameObjectType.WALL;
            }

            else if (displayCharacter == '|')
            {
                type = GameObjectType.GATE;
            }

            else if (displayCharacter == '_')
            {
                type = GameObjectType.GRID;
            }

            return type;
        }
 }
```

- **GameCollisionDetector:**

```
public class GameCollisionDetector
    {
        public bool FireWithEnemy(Fire f)
        {
            bool flag = false;

            if(f.NextCell().CurrentGameObject.GameObjectType ==
GameObjectType.ENEMY)
            {
                flag = true;
            }

            return flag;
        }
```

```
        public bool FireWithPrince(Fire f)
        {
            bool flag = false;

            if (f.NextCell().CurrentGameObject.GameObjectType ==
GameObjectType.PRINCE)
            {
                flag = true;
            }

            return flag;
        }

        public bool PrinceWithEnemy(GameCell cell)
        {
            bool flag = false;

            if (cell.CurrentGameObject.GameObjectType ==
GameObjectType.ENEMY)
            {
                flag = true;
            }

            return flag;
        }

        public bool PrinceWithBomb(GameCell cell)
        {
            bool flag = false;

            if(cell.CurrentGameObject.GameObjectType ==
GameObjectType.BOMB)
            {
                flag = true;
            }

            return flag;
        }

        public bool PrinceWithFruit(GameCell cell)
        {
            bool flag = false;

            if(cell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
            {
                flag = true;
            }

            return flag;
```

```
        }

    }
```

- **Fire:**

```
public class Fire : GameObject
    {
        GameDirection dir;
        bool x = false;

        public Fire(Image image, GameCell startCell, GameDirection dir)
: base(GameObjectType.FIRE, image)
        {
            this.CurrentCell = startCell;
            this.dir = dir;
        }

        public bool X { get => x; set => x = value; }

        public void Move(GameCell gameCell)
        {
            if (this.CurrentCell != null)
            {

this.CurrentCell.SetGameObject(Game.GetBlankGameObject());

            }
            this.CurrentCell = gameCell;
        }

        public GameCell NextCell()
        {

            GameCell currentCell = this.CurrentCell;

            GameCell nextCell = this.CurrentCell.NextCell(dir);

            if (nextCell == currentCell ||
nextCell.CurrentGameObject.GameObjectType == GameObjectType.PRINCE ||
nextCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY ||
nextCell.CurrentGameObject.GameObjectType == GameObjectType.BOMB ||
nextCell.CurrentGameObject.GameObjectType == GameObjectType.REWARD)
            {
                x = true;
                currentCell = nextCell;
            }
            else
            {
```

```
                currentCell = nextCell;
            }
            return currentCell;

        }

    }
```

- **Prince:**

```
public class Prince : GameObject
    {
        int health = 100;
        bool fire = true;

        public Prince(Image img, GameCell startCell) :
base(GameObjectType.PRINCE, img)
        {
            this.CurrentCell = startCell;
        }

        public int Health { get => health; set => health = value; }

        public bool isFire { get => fire; }

        public void Move(GameCell gameCell)
        {
            CurrentCell = gameCell;
        }

        public void Reload(ref int loadedArrow)
        {
            if (loadedArrow <= 0)
            {
                fire = false;
            }
            if (isFire == false)
            {
                loadedArrow++;
            }
            if (loadedArrow == 9)
            {
                fire = true;
            }
        }
    }
```

- **Enemy:**

```
public class Enemy : GameObject
    {
        GameDirection dir = GameDirection.DOWN;
        bool x = false;
        int health = 100;
        int stepCounter = 0;

        public int Health { get => health; set => health = value; }


        public Enemy(Image img, GameCell startCell) :
base(GameObjectType.ENEMY, img)
        {
            this.CurrentCell = startCell;
        }


        public bool X { get => x; set => x = value; }


        public void Move(GameCell gameCell)
        {
            if (this.CurrentCell != null)
            {

this.CurrentCell.SetGameObject(Game.GetBlankGameObject());
            }
            CurrentCell = gameCell;
            stepCounter++;
        }

        public bool isEnemyFire()
        {
            bool fire = false;

            if (stepCounter == 4)
            {
                fire = true;
                stepCounter = 0;
            }

            return fire;
        }


        public GameCell NextCell()
        {
```

```
            GameCell currentCell = this.CurrentCell;

            GameCell nextCell = this.CurrentCell.NextCell(dir);

            if (nextCell == currentCell)
            {
                x = true;
            }
            else
            {
                currentCell = nextCell;
            }
            return currentCell;

        }
    }
```

8.2. Enums Classes:

- **GameDirection:**

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

- **GameObjectType:**

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

```
        REWARD,
        FIRE,
        GATE,
        GRID,
        ENEMY,
        BOMB,
        NONE
    }
```

8.3. Graphical User Interface (GUI) classes:

- **Main Form:**

```
public partial class MainForm : Form
    {
        public MainForm()
        {
            this.WindowState = FormWindowState.Maximized;
            InitializeComponent();
            RunForm();
        }

        private void RunForm()
        {
            MenuForm form = new MenuForm();
            form.StartBtnClick += MenuForm_StartBtnClick;
            form.InstructionsBtnClick += MenuForm_InstructionsBtnClick;
            form.StoryBtnClick += MenuForm_StoryBtnClick;
            form.ExitBtnClick += MenuForm_ExitBtnClick;
            Set_Form(form);
        }



        private void MenuForm_StartBtnClick(object sender, EventArgs e)
        {
            GameForm form = new GameForm();
            form.IsExitClick += GameForm_IsExitClick;
            Set_Form(form);
        }


        private void GameForm_IsExitClick(object sender, EventArgs e)
        {
```

```
            RunForm();
        }

        private void MenuForm_InstructionsBtnClick(object sender,
EventArgs e)
        {
            InstructionsForm form = new InstructionsForm();
            form.IsBackBtnClick += GameForm_IsExitClick;
            Set_Form(form);
        }

        private void MenuForm_StoryBtnClick(object sender, EventArgs e)
        {
            StoryForm form = new StoryForm();
            form.IsBackBtnClick += GameForm_IsExitClick;
            Set_Form(form);
        }

        private void MenuForm_ExitBtnClick(object sender, EventArgs e)
        {
            Application.Exit();
        }


        private void Set_Form(Form name)
        {
            name.TopLevel = false;
            name.FormBorderStyle = FormBorderStyle.None;
            name.Dock = DockStyle.Fill;
            main_Panel.Controls.Add(name);
            main_Panel.Tag = name;
            name.BringToFront();
            name.Show();
        }

    }
```

- **Menu Form:**

```
public partial class MenuForm : Form
    {
        public event EventHandler StartBtnClick;
        public event EventHandler InstructionsBtnClick;
        public event EventHandler StoryBtnClick;
        public event EventHandler ExitBtnClick;

        public MenuForm()
        {
```

```
            InitializeComponent();
        }

        private void start_Btn_Click(object sender, EventArgs e)
        {
            StartBtnClick?.Invoke(this, e);
            this.Close();
        }

        private void instruction_Btn_Click(object sender, EventArgs e)
        {
            InstructionsBtnClick?.Invoke(this, e);
            this.Close();
        }

        private void exit_Btn_Click(object sender, EventArgs e)
        {
            ExitBtnClick?.Invoke(this, e);
            this.Close();
        }

        private void story_Btn_Click(object sender, EventArgs e)
        {
            StoryBtnClick?.Invoke(this, e);
            this.Close();
        }
    }
```

- **Instructions Form:**

```
public partial class InstructionsForm : Form
    {
        public event EventHandler IsBackBtnClick;

        public InstructionsForm()
        {
            InitializeComponent();
        }

        private void back_Btn_Click(object sender, EventArgs e)
        {
            IsBackBtnClick?.Invoke(this, e);
            this.Close();
        }


    }
```

- **Story Form:**

```
public partial class StoryForm : Form
    {

        public event EventHandler IsBackBtnClick;

        public StoryForm()
        {
            InitializeComponent();
        }

        private void back_Btn_Click(object sender, EventArgs e)
        {
            IsBackBtnClick?.Invoke(this, e);
            this.Close();
        }


    }
```

- **Game Form:**

```
public partial class GameForm : Form
    {
        public event EventHandler IsExitClick;


        Game game;
        GameCollisionDetector collider;

        int points = 0;
        int loadedArrow = 9;
        int rightSteps = 0;
        bool enemyPresence = false;
        bool enemyDead = false;

        public GameForm()
        {
            InitializeComponent();
            game = new Game(gameForm_Panel);
            collider = new GameCollisionDetector();
        }

        private void gameLoop_Tick(object sender, EventArgs e)
        {
            ShowScore();
            ShowArrows();
```

```
            ShowHerHealth();
            MoveHeroFire();
            MoveEnemyFire();
            game.RemoveHeroFire();
            game.RemoveEnemyFire();
            game.GenerateFruit(points);
            TakeKeyInput();
            GeneralFunction();
        }

        protected override bool ProcessCmdKey(ref Message msg, Keys
keyData)
        {
            return true;
        }

        private void ResumeFrm_IsBtnClick(object sender, EventArgs e)
        {
            IsExitClick?.Invoke(this, e);
        }


        private void GeneralFunction()
        {
            if (enemyPresence)
            {
                ShowEnemyHealth();
                MoveEnemy();
                game.CreateBombs();
            }

            if (game.WinGame())
            {
                gameLoop.Stop();
                MessageForm frm = new
MessageForm(ThePrinceArcher_Game.Properties.Resources.game_win);
                frm.HideContinueBtn();
                frm.IsReturnBtnClick += ResumeFrm_IsBtnClick;
                frm.ShowDialog();
            }

            if (rightSteps == 2)
            {
                AddRightStep();
                game.WallRemove();
            }

            if (rightSteps == 10)
            {
                AddRightStep();
```

```
                game.PrintWall();
                enemyPresence = true;
                game.EnemyPresence();
            }

            if (enemyDead)
            {
                enemyPresence = false;
                game.OpenGate();
            }

        }



        private void TakeKeyInput()
        {
            Prince prince = game.GetPrince();
            GameCell potentialNewCell = prince.CurrentCell;

            if (Keyboard.IsKeyPressed(Key.LeftArrow))
            {
                potentialNewCell =
prince.CurrentCell.NextCell(GameDirection.LEFT);
            }

            else if (Keyboard.IsKeyPressed(Key.RightArrow))
            {
                AddRightStep();
                potentialNewCell =
prince.CurrentCell.NextCell(GameDirection.RIGHT);
            }

            else if (Keyboard.IsKeyPressed(Key.UpArrow))
            {
                potentialNewCell =
prince.CurrentCell.NextCell(GameDirection.UP);
            }

            else if (Keyboard.IsKeyPressed(Key.DownArrow))
            {
                potentialNewCell =
prince.CurrentCell.NextCell(GameDirection.DOWN);
            }

            else if (Keyboard.IsKeyPressed(Key.Space))
            {
                if (prince.isFire)
                {
```

```
                    GameCell NewCell =
prince.CurrentCell.NextCell(GameDirection.RIGHT);
                    Fire f = new
Fire(ThePrinceArcher_Game.Properties.Resources.arrow, NewCell,
GameDirection.RIGHT);
                    game.AddHeroFire(f);
                    loadedArrow--;
                }
            }

            else if (Keyboard.IsKeyPressed(Key.Escape))
            {
                gameLoop.Stop();
                MessageForm frm = new
MessageForm(ThePrinceArcher_Game.Properties.Resources._continue);
                frm.IsContinueBtnClick += ResumeFrm_IsContinueBtnClick;
                frm.IsReturnBtnClick += ResumeFrm_IsBtnClick;
                frm.ShowDialog();
            }

            prince.Reload(ref loadedArrow);
            game.NextCellFunctions(potentialNewCell, collider, ref
points);

            GameCell currentCell = prince.CurrentCell;
            currentCell.SetGameObject(Game.GetBlankGameObject());
            prince.Move(potentialNewCell);
        }

        private void ResumeFrm_IsContinueBtnClick(object sender,
EventArgs e)
        {
            gameLoop.Start();
        }

        public void AddRightStep()
        {
            rightSteps++;
            if (rightSteps >= 12)
            {
                rightSteps = 12;
            }
        }



        private void ShowScore()
        {
            score.Text = points.ToString();
        }
```

```
        private void ShowArrows()
        {
            arrows.Text = "X " + loadedArrow.ToString();
        }



        private void ShowHerHealth()
        {
            if (0 < game.GetPrinceHealth())
            {
                prince_Health_Bar.Value = game.GetPrinceHealth();
            }
            else if (0 == game.GetPrinceHealth())
            {
                prince_Health_Bar.Value = game.GetPrinceHealth();

                gameLoop.Stop();
                MessageForm frm = new
MessageForm(ThePrinceArcher_Game.Properties.Resources.game_lose);
                frm.HideContinueBtn();
                frm.IsReturnBtnClick += ResumeFrm_IsBtnClick;
                frm.ShowDialog();
            }
        }

        private void ShowEnemyHealth()
        {
            if (0 < game.GetEnemyHealth())
            {
                enemy_Health_Bar.Value = game.GetEnemyHealth();
            }
            else if (0 == game.GetEnemyHealth())
            {
                enemy_Health_Bar.Value = game.GetEnemyHealth();

                enemyDead = true;
            }
        }



        private void MoveHeroFire()
        {
            foreach (Fire f in game.HeroFires)
            {
                if (collider.FireWithEnemy(f))
                {
                    points = points + 5;
```

```
                    game.DecreaseEnemyHealth();
                    game.DecreaseBombNFruit(f);
                }
                f.Move(f.NextCell());
            }
        }

        private void MoveEnemyFire()
        {
            foreach (Fire f in game.EnemyFires)
            {
                if (collider.FireWithPrince(f))
                {
                    game.DecreaseHeroHealth();
                    game.DecreaseBombNFruit(f);
                }

                f.Move(f.NextCell());
            }
        }




        private void MoveEnemy()
        {
            Enemy enemy = game.GetEnemy();
            GameCell potentialNewCell = enemy.CurrentCell;
            GameCell nextCell;

            Random rnd = new Random();
            int num = rnd.Next(2, 4);

            if (num == 2)
            {
                nextCell = enemy.CurrentCell.NextCell(GameDirection.UP);
                if (nextCell != potentialNewCell)
                {
                    GameCell currentCell = enemy.CurrentCell;

currentCell.SetGameObject(Game.GetBlankGameObject());
                    enemy.Move(nextCell);
                }
            }
            else if (num == 3)
            {
                nextCell =
enemy.CurrentCell.NextCell(GameDirection.DOWN);
                if (nextCell != potentialNewCell)
                {
```

```
                    GameCell currentCell = enemy.CurrentCell;

currentCell.SetGameObject(Game.GetBlankGameObject());
                    enemy.Move(nextCell);
                }
            }

            EnemyFunctions(enemy);
        }

        private void EnemyFunctions(Enemy enemy)
        {
            if (enemy.isEnemyFire())
            {
                GameCell NewCell =
enemy.CurrentCell.NextCell(GameDirection.LEFT);
                Fire f = new
Fire(ThePrinceArcher_Game.Properties.Resources.bullet, NewCell,
GameDirection.LEFT);
                game.AddEnemyFire(f);
            }
        }
    }
```

- **Message Form:**

```
public partial class MessageForm : Form
    {
        public event EventHandler IsContinueBtnClick;
        public event EventHandler IsReturnBtnClick;

        public MessageForm(Image img)
        {
            InitializeComponent();
            showImage_PB.Image = img;
        }

        private void continue_Btn_Click(object sender, EventArgs e)
        {
            IsContinueBtnClick?.Invoke(this, e);
            this.Close();
        }

        private void return_Btn_Click(object sender, EventArgs e)
        {
            IsReturnBtnClick?.Invoke(this, e);
            this.Close();
        }

        public void HideContinueBtn()
```

```
        {
            continue_Btn.Visible = false;
        }
    }
```

9. CONCLUSION:

In conclusion, I proudly present "The Prince Archer," an enchanting WinForms experience that seamlessly fuses captivating gameplay with the innovative implementation of GL (Game Logic) and UI (User Interface) classes. This extraordinary gaming adventure invites players to embody the role of the ultimate fire-wielding hero, embarking on a thrilling journey filled with challenges, excitement, and the joy of mastering the art of archery. With meticulous attention to detail and a passion for immersive storytelling, "The Prince Archer" aims to leave an indelible mark on players, offering an unparalleled gaming odyssey that will be cherished and remembered for years to come..