Super Mario Game

Session: 2022-2026

Submitted by:

Noor Fatima 2022-CS-12

Supervised by:

Maida Shahid

Department of Computer science

University of Engineering and Technology Lahore Pakistan

Table of Content

Topics **Page no.**

Description 02
Game Character Description....................................02
Player..02
Enemies..02
Wireframes..04
Code..08
Game Objects Description....................................19
Rules & Interactions 19
Goal of the game....................................19

My YouTube video link

https://youtu.be/xRAPnzJKybU

Description

A plumber name Mario and his brother Luigi travels through Mushroom kingdom to save princess. Mario is always bright, cheerful and recognizable with his blue overalls, red caps and trademark. He is the trusted friend of princess toadstool. Plumber Mario and Luigi from Brooklyn just arrived in an outlandish realm called mushroom kingdom. It was ruled by toadstool and her faithful mushroom people.

But one day, princess toadstool kidnapped by famous king koopa and toads of the mushroom kingdom have been held captive in many castles of king koopa. So, its Mario up to Mario to save princess toadstool from king koopa.

For this Mario have to defeat king koopa and his minions. He has to walk, run, jump and climb through obstacles. The world is full of enemies, platform and enemies. So, Mario has access to a variety of power-ups that give him different abilities.

Game Character Description

Player

There is one human player in the game.

Mario

Mario is the main hero Mushroom kingdom. He is always bright, cheerful and recognizable with his blue overalls, red caps and trademark. He is the trusted friend of princess toadstool. Mario is brave, determined and has a never-say-die spirit. He is the hero of the game, admired for his bravery and determination in the face of danger.

Enemies

There are three enemies in this game

Goomba:

Goomba is the biggest enemy of Mario. He is move randomly in the maze.

Pokey:

Pokey is similar to sun. He moves horizontally in the maze.

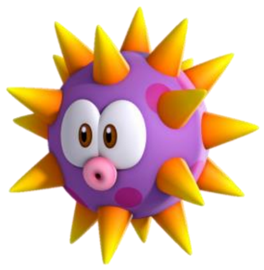
Blooper:

Blooper is a white color enemy. He moves vertically in the maze.

Wireframes:

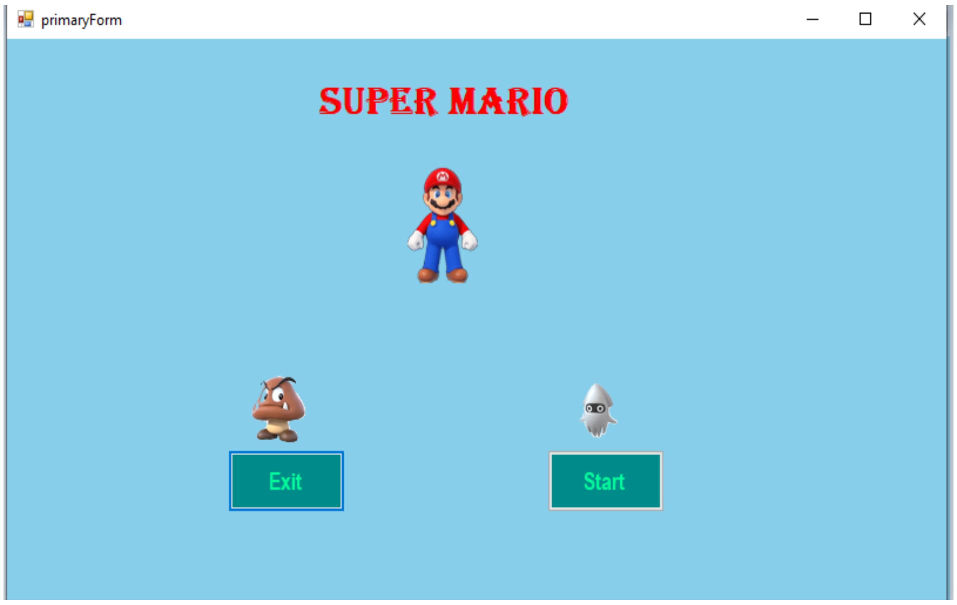
Game player:

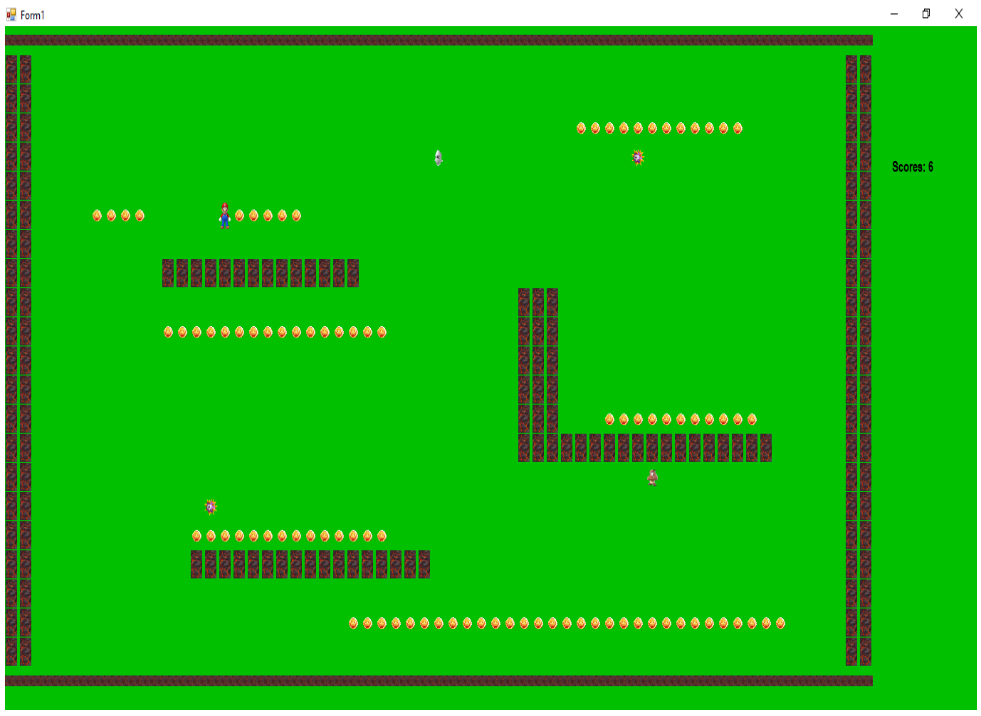
Horizontal Enemy:

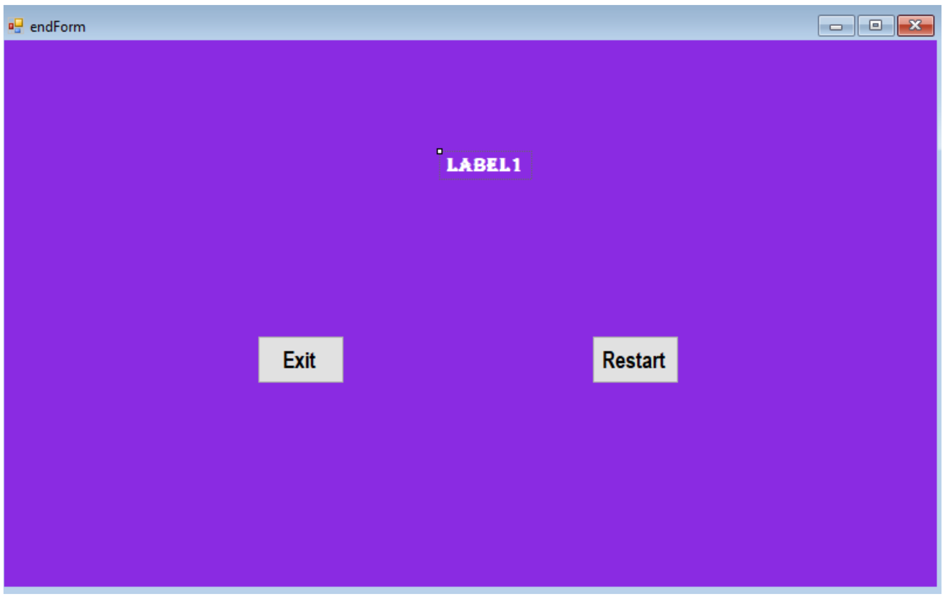


Vertical Enemy:

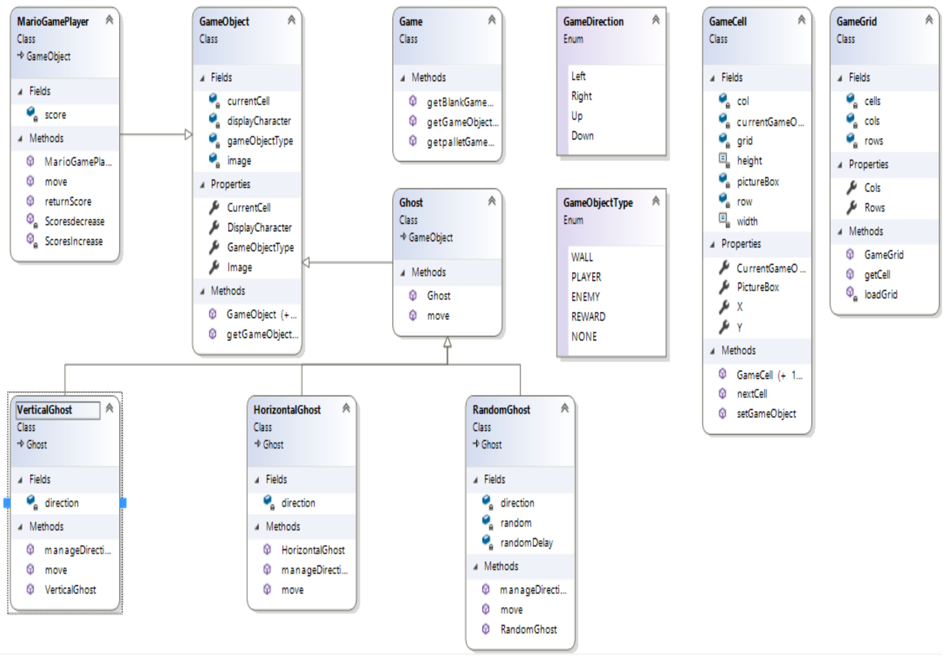
Smart Enemy:


primaryForm
SUPER MARIO
Exit
Start

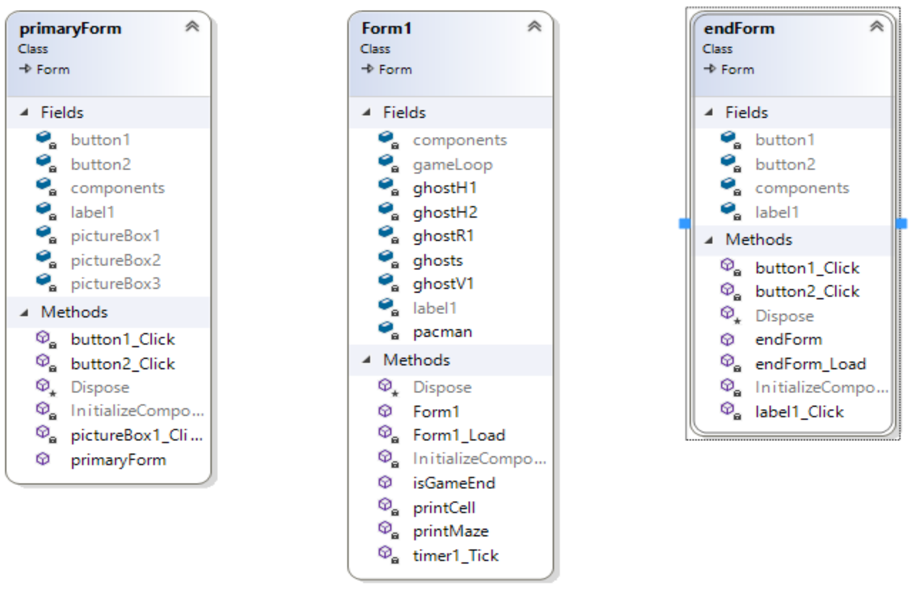

Form1
Scores: 6


endForm
LABEL1
Exit
Restart

CRC Diagram:


MarioGamePlayer
Class
GameObject
Fields
score
Methods
MarioGamePla...
move
returnScore
Scoresdecrease
ScoresIncrease
GameObject
Class
Fields
currentCell
displayCharacter
gameObjectType
image
Properties
CurrentCell
DisplayCharacter
GameObjectType
Image
Methods
GameObject (+...
getGameObject...
Game
Class
Methods
getBlankGame...
getGameObject...
getpalletGame...
Ghost
Class
GameObject
Methods
Ghost
move
GameDirection
Enum
Left
Right
Up
Down
GameObjectType
Enum
WALL
PLAYER
ENEMY
REWARD
NONE
GameCell
Class
Fields
col
currentGameO...
grid
height
pictureBox
row
width
Properties
CurrentGameO ...
PictureBox
X
Y
Methods
GameCell (+ 1...
nextCell
setGameObject
GameGrid
Class
Fields
cells
cols
rows
Properties
Cols
Rows
Methods
GameGrid
getCell
loadGrid
VerticalGhost
Class
Ghost
Fields
direction
Methods
manageDirecti...
move
VerticalGhost
HorizontalGhost
Class
Ghost
Fields
direction
Methods
HorizontalGhost
manageDirecti...
move
RandomGhost
Class
Ghost
Fields
direction
random
randomDelay
Methods
manageDirecti...
move
RandomGhost

Super Mario Game



Working Code

Game Class:

```
                    class Game
{
    public static GameObject getBlankGameObject()
    {
        GameObject blankGameObject = new GameObject(GameObjectType.NONE, gamePacOop.Properties.Resources.simplebox);
        return blankGameObject;
    }

    public static GameObject getpalletGameObject()
    {
        GameObject palletGameObject = new GameObject(GameObjectType.REWARD, gamePacOop.Properties.Resources.coin);
        return palletGameObject;
    }
    public static Image getGameObjectImage(char displayCharacter)
    {
        Image img = gamePacOop.Properties.Resources.simplebox;
        if (displayCharacter == '|' || displayCharacter == '%')
        {
            img = gamePacOop.Properties.Resources._2;
        }

        if (displayCharacter == '#')
        {
            img = gamePacOop.Properties.Resources._1;
        }

        if (displayCharacter == '.')
        {
            img = gamePacOop.Properties.Resources.coin;
        }
        if (displayCharacter == 'P' || displayCharacter == 'p')
        {
            img = gamePacOop.Properties.Resources.Mario;
        }
        if (displayCharacter == 'H' || displayCharacter == 'h')
        {
            img = gamePacOop.Properties.Resources.HGhost;
        }
        if (displayCharacter == 'V' || displayCharacter == 'v')
        {
            img = gamePacOop.Properties.Resources.VGhost;
        }


        if (displayCharacter == 'R' || displayCharacter == 'r')
        {
            img = gamePacOop.Properties.Resources.RGhost;
        }

        return img;
    }
}
```

GameCell Class:

```
class GameCell
{
    int row;
    int col;
    GameObject currentGameObject;
```

```
        GameGrid grid;
        PictureBox pictureBox;
        const int width = 20;
        const int height = 30;
        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 GameCell()
        {

        }
        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.NONE || ncell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
                    {
                        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.NONE || ncell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Up)
            {
                if (this.row > 0)
                {
                    GameCell ncell = grid.getCell(this.row - 1, this.col);
                    if (ncell.CurrentGameObject.GameObjectType == GameObjectType.NONE || ncell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
                    {
                        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.NONE || ncell.CurrentGameObject.GameObjectType ==
GameObjectType.REWARD)
                    {
                        return ncell;
                    }
                }
            }
            return this; // if can not return next cell return its own reference
        }
        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; }


}
```

GameDirection Class:

```
public enum GameDirection
    {
        Left,
        Right,
        Up,
        Down
    }
```

GameGrid Class:

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

        public GameGrid(String fileName, int rows, int cols)
        {
            //Numbers of rows and cols should load from the text file
            this.rows = rows;
            this.cols = cols;
            cells = new GameCell[rows, cols];
            this.loadGrid(fileName);
        }
        public GameCell getCell(int x, int y)
        {
            return cells[x, y];
        }
        public int Rows { get => rows; set => rows = value; }
        public int Cols { get => cols; set => cols = value; }

        void loadGrid(string fileName)
        {

            StreamReader fp = new StreamReader(fileName);
            string record;
            for (int row = 0; row < this.rows; row++)
            {
                record = fp.ReadLine();
                for (int col = 0; col < this.cols; col++)
                {
```

```
                GameCell cell = new GameCell(row, col, this);
                char displayCharacter = record[col];
                GameObjectType type = GameObject.getGameObjectType(displayCharacter);
                Image displayIamge = Game.getGameObjectImage(displayCharacter);
                GameObject gameObject = new GameObject(type, displayIamge);
                cell.setGameObject(gameObject);
                cells[row, col] = cell;
            }
        }

        fp.Close();
    }

}
```

GameObject Class:

```
  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 == '|' || displayCharacter == '%' || displayCharacter == '#')
        {
            return GameObjectType.WALL;
        }

        if (displayCharacter == '.')
        {
            return GameObjectType.REWARD;
        }

        return GameObjectType.NONE;
    }
    public char DisplayCharacter { get => displayCharacter; set => displayCharacter = value; }
    public GameObjectType GameObjectType { get => gameObjectType; set => gameObjectType = value; }
    public GameCell CurrentCell
    {
        get => currentCell;
        set
        {
            currentCell = value;
            currentCell.setGameObject(this);
        }
    }

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

GameObjectType:

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

MarioGamePlayer Class:

```
class MarioGamePlayer : GameObject
{

    int score;
    public MarioGamePlayer(Image image, GameCell startCell) : base(GameObjectType.PLAYER, image)
    {
        this.CurrentCell = startCell;
    }
    public GameCell move(GameDirection direction)
    {
        GameCell currentCell = this.CurrentCell;
        GameCell nextCell = currentCell.nextCell(direction);
        ScoresIncrease(nextCell);
        Scoresdecrease(nextCell);

        this.CurrentCell = nextCell;


        if (currentCell != nextCell)
        {
            currentCell.setGameObject(Game.getBlankGameObject());

        }
        return nextCell;
    }


    private void ScoresIncrease(GameCell nextCell)
    {
        if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.REWARD)
        {
            score++;
        }

    }
    private void Scoresdecrease(GameCell nextCell)
    {
        if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.ENEMY)
        {
            score--;
        }

    }
    public   int returnScore()
    {
        return this.score;
    }


}
```

Ghost Class:

```
class Ghost : GameObject
{
    public Ghost(GameObjectType gameObjectType, Image image) : base(GameObjectType.ENEMY, image)
    {

    }

    public virtual GameCell move()
    {
        GameCell a = new GameCell();
        return a;

    }


}
```

HorizontalGhost Class:

```
class HorizontalGhost : Ghost
 {
    GameDirection direction;
    public HorizontalGhost(GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.ENEMY, image)
    {
        this.direction = direction;
        this.CurrentCell = startCell;
    }

    public override GameCell move()
    {
        GameCell currentCell = this.CurrentCell;
        GameCell nextCell = currentCell.nextCell(direction);
        GameObject previousObject = nextCell.CurrentGameObject;
        this.CurrentCell = nextCell;

        if (currentCell == nextCell)
        {
            manageDirections();
            currentCell.setGameObject(currentCell.CurrentGameObject);
        }

        if (currentCell != nextCell)
        {
            currentCell.setGameObject(previousObject);

        }
        else
        {

        }
        return nextCell;
    }

    public void manageDirections()
    {
        if (direction == GameDirection.Left)
        {
            direction = GameDirection.Right;
        }
        else if (direction == GameDirection.Right)
        {
            direction = GameDirection.Left;
        }
    }

 }
```

VerticalGhost Class:

```
 class VerticalGhost : Ghost
{
   GameDirection direction;
   public VerticalGhost(GameDirection direction, Image image, GameCell startCell) : base(GameObjectType.ENEMY, image)
   {
       this.direction = direction;
       this.CurrentCell = startCell;
   }

   public override GameCell move()
   {
       GameCell currentCell = this.CurrentCell;
       GameCell nextCell = currentCell.nextCell(direction);
       GameObject previousObject = nextCell.CurrentGameObject;
       this.CurrentCell = nextCell;

       if (currentCell == nextCell)
       {
           manageDirections();
       }

       if (currentCell != nextCell)
       {
           currentCell.setGameObject(previousObject);

       }
       return nextCell;
   }

   public void manageDirections()
   {
       if (direction == GameDirection.Up)
       {
           direction = GameDirection.Down;
       }
       else if (direction == GameDirection.Down)
       {
           direction = GameDirection.Up;
       }
   }
}
```

RandomGhost Class:

```
  class RandomGhost : Ghost
{

   int randomDelay;
   int random;
   GameDirection direction;

   public RandomGhost( Image image, GameCell startCell) : base(GameObjectType.ENEMY, image)
   {
       this.CurrentCell = startCell;

   }

   public override GameCell move()
   {
       manageDirections();
       GameCell currentCell = this.CurrentCell;
```

```
            GameCell nextCell = currentCell.nextCell(direction);
            GameObject previousObject = nextCell.CurrentGameObject;
            this.CurrentCell = nextCell;




            if (currentCell != nextCell)
            {
                currentCell.setGameObject(previousObject);

            }
            return nextCell;
        }

        public void manageDirections()
        {
            if (randomDelay % 5 == 0)
            {
                Random r = new Random();
                random = r.Next(4);
            }

            if (random == 0)
            {
                direction = GameDirection.Right;
            }
            else if (random == 1)
            {
                direction = GameDirection.Left;
            }
            else if (random == 2)
            {
                direction = GameDirection.Up;
            }
            else if (random == 3)
            {
                direction = GameDirection.Down;
            }
            randomDelay++;

        }
    }
```

Forms:

Primary Form:

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

        private void pictureBox1_Click(object sender, EventArgs e)
        {

        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Close();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            Form moreform = new Form1();
            moreform.Show();
        }
    }
```

Main Form:

```
public partial class Form1 : Form
    {
        MarioGamePlayer pacman;
        HorizontalGhost ghostH1;
        HorizontalGhost ghostH2;
        VerticalGhost ghostV1;
        RandomGhost ghostR1;


        List<Ghost> ghosts = new List<Ghost>();
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            GameGrid grid = new GameGrid("maze.txt", 23, 61);
            Image pacManImage = GameGL.Game.getGameObjectImage('P');
            GameCell startCell = grid.getCell(8, 10);
            pacman = new MarioGamePlayer(pacManImage, startCell);

            Image ghostH1Img = GameGL.Game.getGameObjectImage('H');
            GameCell startH1 = grid.getCell(16, 45);
            ghostH1 = new HorizontalGhost(GameDirection.Left, ghostH1Img, startH1);
            Image ghostH2Img = GameGL.Game.getGameObjectImage('H');
            GameCell startH2 = grid.getCell(4, 15);
            ghostH2 = new HorizontalGhost(GameDirection.Left, ghostH2Img, startH2);

            Image ghostV1Img = GameGL.Game.getGameObjectImage('V');
            GameCell startV1 = grid.getCell(6, 30);
            ghostV1 = new VerticalGhost(GameDirection.Up, ghostV1Img, startV1);

            Image ghostR1Img = GameGL.Game.getGameObjectImage('R');
            GameCell startR1 = grid.getCell(10, 23);
            ghostR1 = new RandomGhost( ghostR1Img, startR1);



            ghosts.Add(ghostH1);
            ghosts.Add(ghostH2);
            ghosts.Add(ghostV1);
            ghosts.Add(ghostR1);


            printMaze(grid);
        }

        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);
                    //        printCell(cell);
                }
            }
        }
        static void printCell(GameCell cell)
        {
            Console.SetCursorPosition(cell.Y, cell.X);
            Console.Write(cell.CurrentGameObject.DisplayCharacter);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {

            if (!isGameEnd())
            {


                if (Keyboard.IsKeyPressed(Key.LeftArrow))
                {
                    pacman.move(GameDirection.Left);

                }
                else if (Keyboard.IsKeyPressed(Key.RightArrow))
                {
                    pacman.move(GameDirection.Right);

                }
                else if (Keyboard.IsKeyPressed(Key.UpArrow))
                {
                    pacman.move(GameDirection.Up);

                }
                else if (Keyboard.IsKeyPressed(Key.DownArrow))
                {
                    pacman.move(GameDirection.Down);

                }

                foreach (Ghost g in ghosts)
                {

                    g.move();
                }

                label1.Text = "Scores: " + pacman.returnScore();
            }
            else
            {
                gameLoop.Stop();
                this.Hide();
                endForm gameover = new endForm();
                gameover.Show();
            }

        }
       public bool isGameEnd()
        {

            int score = pacman.returnScore();
            if (score == -1)
            {
                return true;
            }
            else
            {
                return false;
            }
        }
    }
```

End Form:

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

        private void label1_Click(object sender, EventArgs e)
        {

        }

        private void endForm_Load(object sender, EventArgs e)
        {
            label1.Text = "You lost";
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
            primaryForm primary = new primaryForm();
            primary.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            this.Hide();
            Form1 newForm = new Form1();
            newForm.Show();
        }
    }
```

Game Objects Description

Followings are some objects in the game:

Coin pallet:

There is a line of round circles known as coin. When Mario collects each coin, his score will increase by 1.

Walls:

Walls are the barrier in the game which Mario and his enemies cannot cross.

Rules & Interactions

Mario can collect coins that have been put across the platform. Mario will lose his one life he will collides with walls, turtle or sun. Mario can kill turtle by jumping on it. As Mario collects coin his score will increase more and more. The Game will be end if Mario saves princess.

Goal of the Game

The goal of the game is to save princess without collision with any of the enemy.