POLICEMAN & THIEFS

Session: 2022 – 2026

Submitted by:

Mohammad Zaid 2022R\2021-CS-214

Supervised by:

Prof. Dr. Awais Hassan

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Table of Contents

Youtube Video Link

Complete Journey: https://youtu.be/PndvBJ_Ufa8

Story

The story of the game is that there is a policeman 3 treasure coins and thiefs policeman must have to Save these coins from thiefs.

SHORT DESCRIPTION

The goal of the game is to navigate the maze and catch treasure coins before Thiefs catch you. However, the maze is full of twists, turns, and obstacles that will make your catching treasure coins a challenging and exciting adventure.

As you move through the maze, enemies such as Chasing thiefs , vertical thiefs , and horizontals thiefs will be constantly searching for you, using their keen senses to try and track you down.

You'll need to be quick and clever to outsmart the enemies and collect the treasure coins. Smart and Chasing thiefs try to chase you and fire the treasure coin if they fire coin then coin disappear and you Loose. You need to be save the treasure coins. If you collect all three Coins then You Win And you Promoted to the Stage 3. Note: This is only 3 levels Game initially.

Game Characters Description

Player:

There is one player in the Game named as "Policeman".

Enemies

There are many types of enemies in my game such as:

1. **Vertical Thief**: It is one of the thiefs of the game which moves Verticals direction in the maze which cannot fire.

2. **Horizontal Thief:** It is one of the thiefs of the game which moves Horizontal direction in the maze which cannot fire .

3. **Chasing Thief:** It is a definitely the enemy you should be aware of, because it is constantly chasing you decrease your health and fire the treasure coins to disapears.

Game Objects Description

Following are the Objects in the Game

Walls:

Walls are the barriers in the game which the policeman and thiefs cannot cross.

Shooting System:

- The Player-Policeman will be able to shoot in left direction when moving in the left and right direction when moving right in the direction (Left/Right).
- The Chasing Thief will be able to shoot in LEFT and RIGHT direction according to Policeman movement.

Rules & Interactions

The rules and instructions of "Polceman and Thiefs" are as follows:

1. The goal of the game is to navigate the maze and collect the treasure coins before enemies catch you.

2. You control Policeman using the arrow keys on your keyboard. Use the up arrow to move upward, down arrow to move downward, the left arrow to turn left, and the right arrow to turn right.

3. Avoid hitting obstacles, such as walls and barriers, as they will slow you down and make it easier for enemies to catch you.

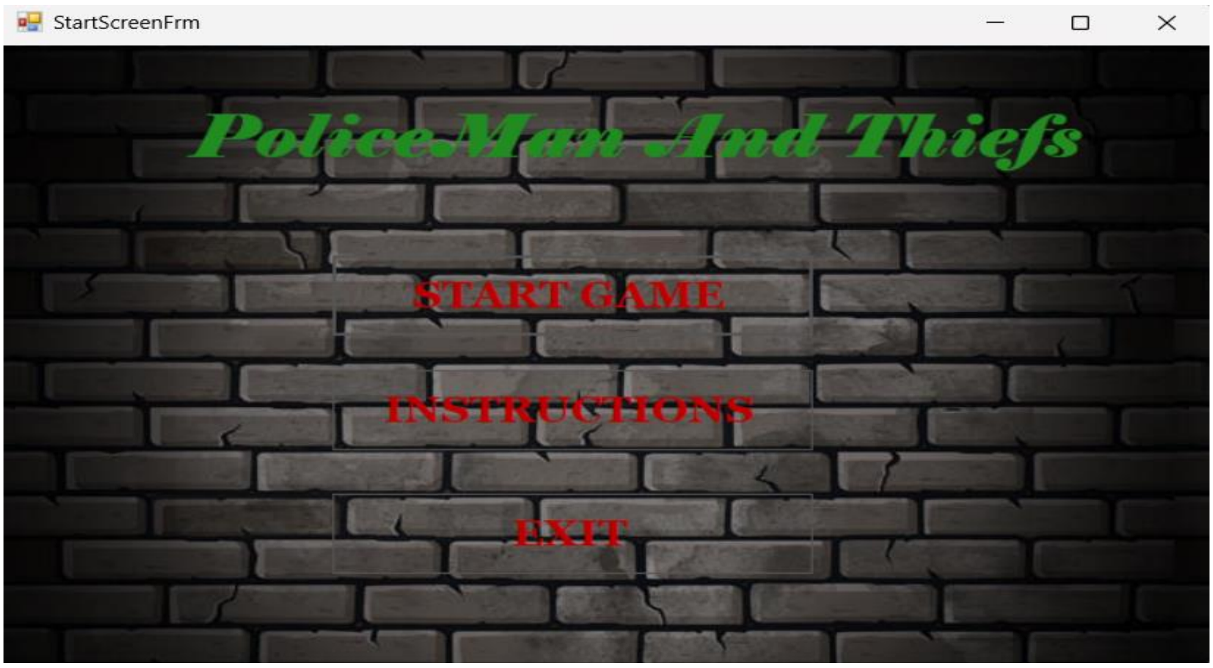
4. It's not necessary to kill All the Enemies you just need to save yourself and coins from chasing ghost
   And smart ghost and collect all these coins.

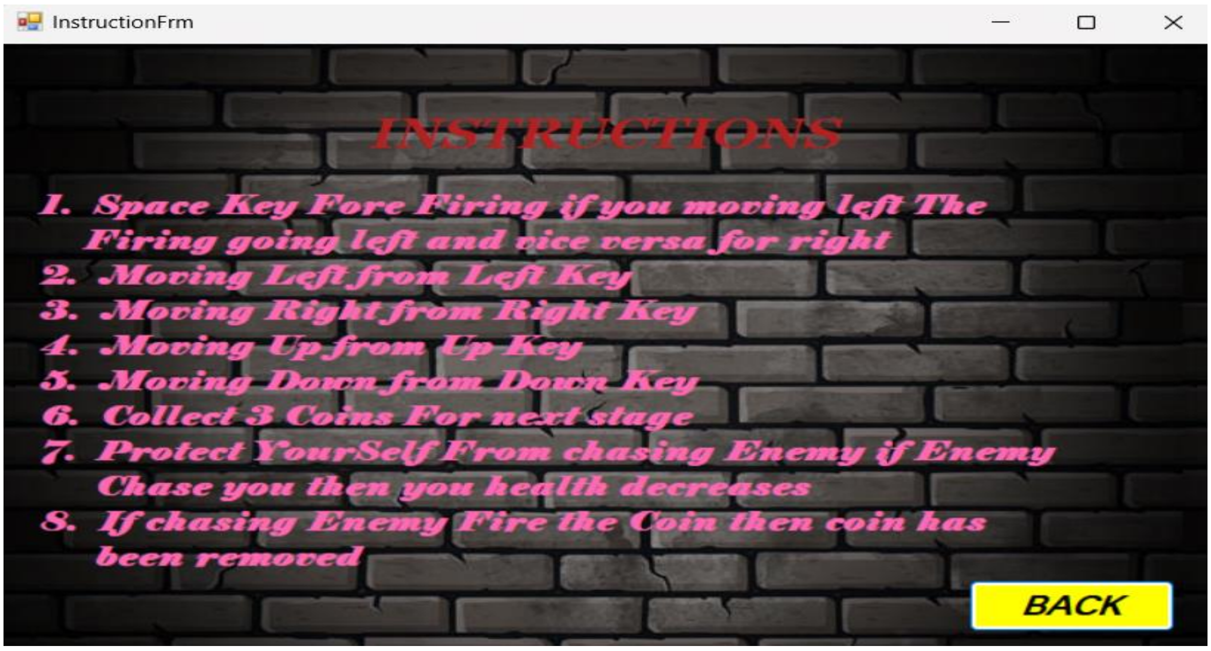
5. Once You collect all three treasure coins you win the Stage 1 same as Stage 2 and vice versa Stage 3.

GOAL OF THE GAME

The goal of the game is simple yet challenging – to collect the coins By Smartness and save yourself and coins from thiefs.

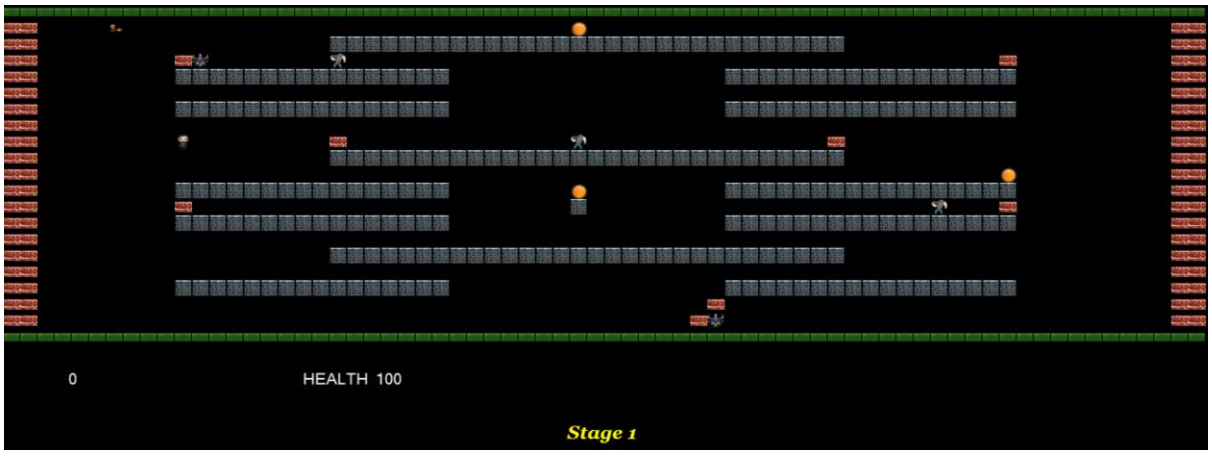
Wire Frames:

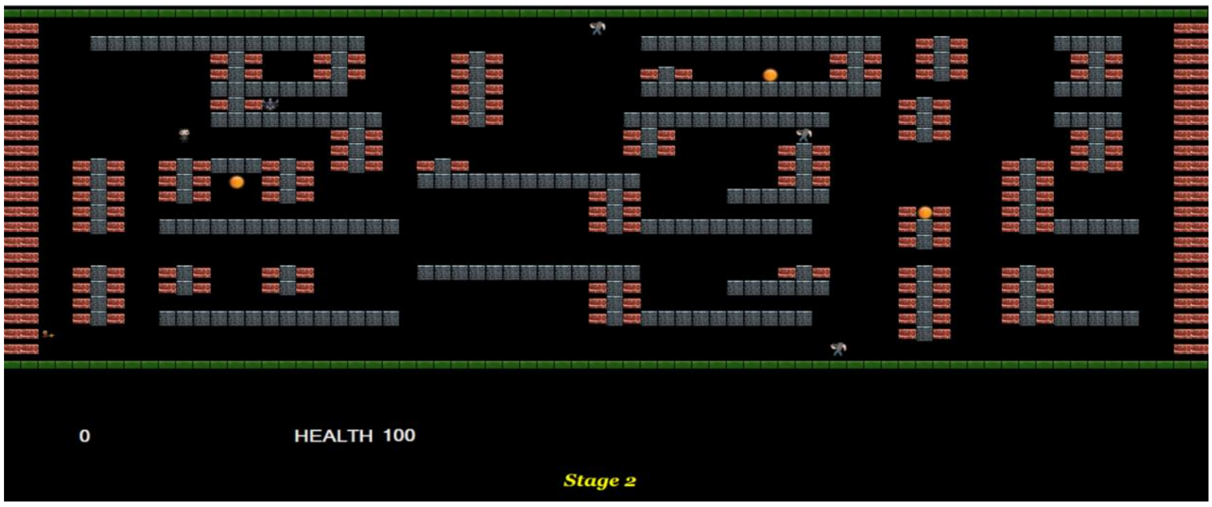


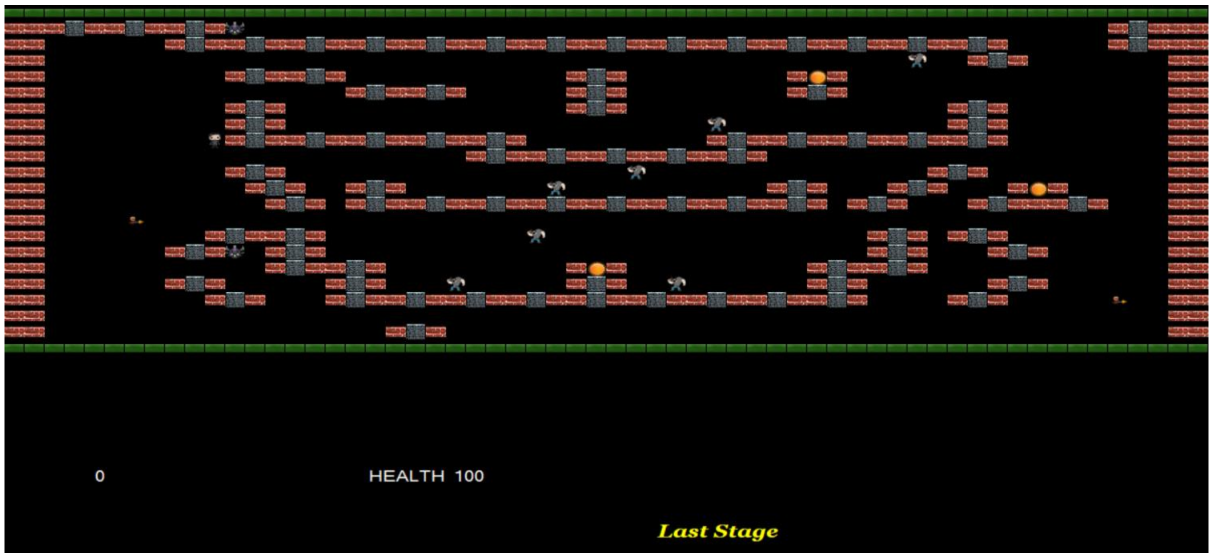

InstructionFrm

INSTRUCTIONS

1. Space Key Fore Firing if you moving left The Firing going left and vice versa for right
2. Moving Left from Left Key
3. Moving Right from Right Key
4. Moving Up from Up Key
5. Moving Down from Down Key
6. Collect 3 Coins For next stage
7. Protect YourSelf From chasing Enemy if Enemy Chase you then you health decreases
8. If chasing Enemy Fire the Coin then coin has been removed


0
HEALTH 100
Stage 1


0
HEALTH 100
Stage 2


0
HEALTH 100
Last Stage

Object Oriented Programming:

In this Game I have completely followed the OOP concepts.

Polymorphism:

I have implemented Polymorphism in my game. The move function in Player and Enemy classes

Inheritance:

I have implemented inheritance at multiple places like in GameObject and Enemy classes.

Encapsulation:

I have implemented encapsulation in my game I have made private attributes and getter setters.

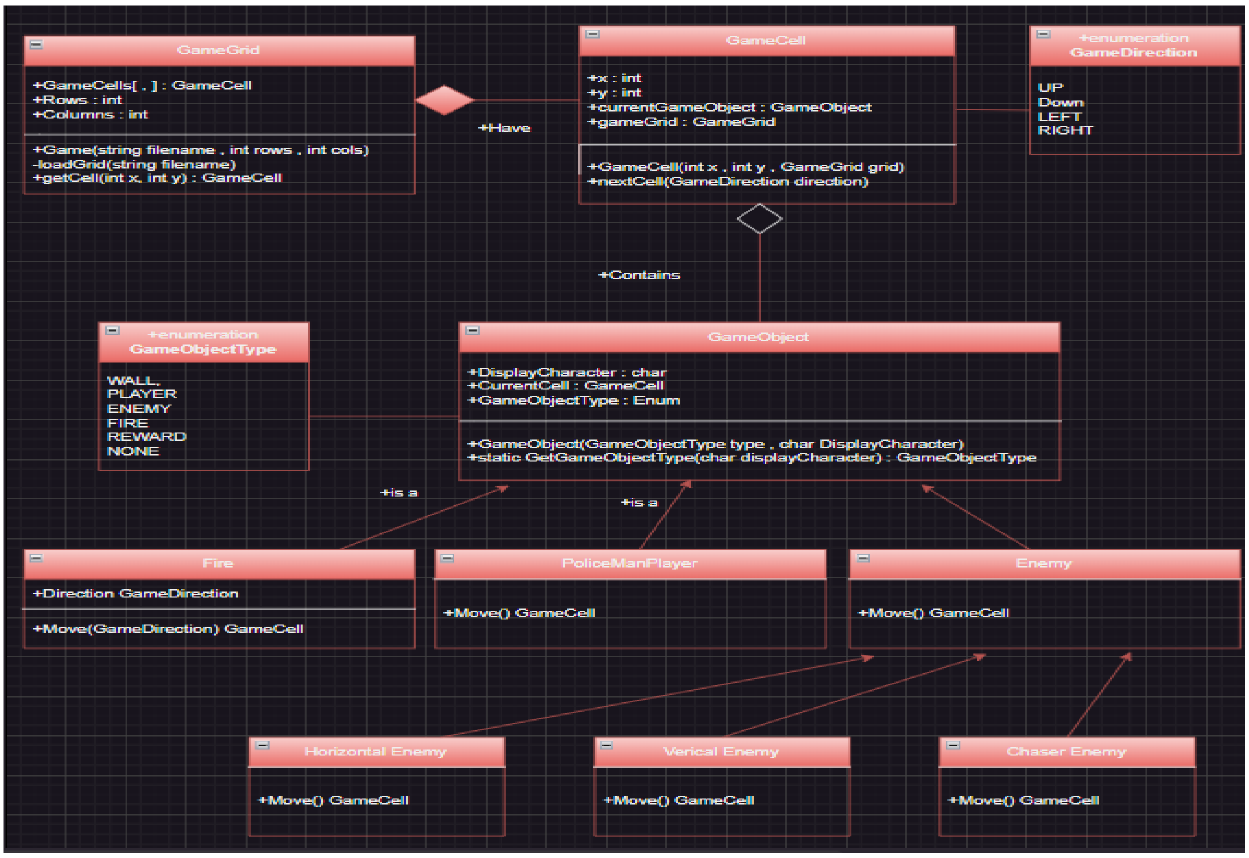
Abstraction:

I have implemented Abstract methods and abstract Enemy class.

Enumeration:

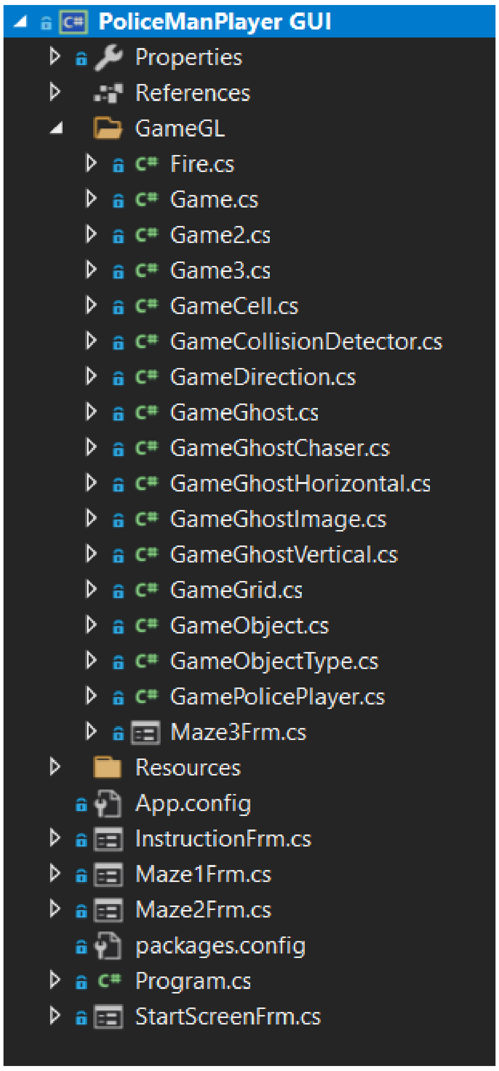
I have also implemented enumeration classes in my Game.

Class Diagram:



• Design Pattern Implementation

The directory structure for the project is given below:



```
PoliceManPlayer GUI
    Properties
    References
    GameGL
        Fire.cs
        Game.cs
        Game2.cs
        Game3.cs
        GameCell.cs
        GameCollisionDetector.cs
        GameDirection.cs
        GameGhost.cs
        GameGhostChaser.cs
        GameGhostHorizontal.cs
        GameGhostImage.cs
        GameGhostVertical.cs
        GameGrid.cs
        GameObject.cs
        GameObjectType.cs
        GamePolicePlayer.cs
        Maze3Frm.cs
    Resources
    App.config
    InstructionFrm.cs
    Maze1Frm.cs
    Maze2Frm.cs
    packages.config
    Program.cs
    StartScreenFrm.cs
```

Conclusion

In conclusion, my game is built using the object-oriented programming approach. Its key functionalities includes the Firing, Moving and implementation of Framework. Important concepts object-oriented concepts such as association, inheritance and polymorphism are used in this system. I faced several challenges during this phase. I faced difficulty in designing an effective class diagram collaboration model for the game and managing the key concepts of OOP paradigm. Throughout the period of designing, production and development of this project, I have learned how to create an effective game using object-oriented theory. The object-oriented approach can be really helpful in scaling of the project. It also helps the programmers in future to maintain and develop Games.

Code

GameGrid Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class GameGrid
    {
        GameCell[,] cells;
        int rows;
        int cols;

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

        void loadGrid(string fileName)
        {

            StreamReader fp = new StreamReader(fileName);
            string record;
            for (int row = 0; row < this.rows; row++)
            {
                record = fp.ReadLine();
                for (int col = 0; col < this.cols; col++)
                {
                    GameCell cell = new GameCell(row, col, this);
                    char displayCharacter = record[col];
                    GameObjectType type =
GameObject.getGameObjectType(displayCharacter);
                    Image displayIamge = Game.getGameObjectImage(displayCharacter);
                    GameObject gameObject = new GameObject(type, displayIamge);
                    cell.setGameObject(gameObject);
                    cells[row, col] = cell;
                }
            }

            fp.Close();
        }
    }
}
```

GameCell Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class GameCell
    {
        int row;
        int col;
        GameObject currentGameObject;
        GameGrid grid;
        PictureBox pictureBox;
        const int width = 20;
        const int height = 20;
        public GameCell(int row, int col, GameGrid grid)
        {
            this.row = row;
            this.col = col;
            pictureBox = new PictureBox();
            pictureBox.Left = col * width;
            pictureBox.Top = row * height;
            pictureBox.Size = new Size(width, height);
            pictureBox.SizeMode = PictureBoxSizeMode.Zoom;
            pictureBox.BackColor = Color.Transparent;
            this.grid = grid;
        }
        public void setGameObject(GameObject gameObject)
        {
            currentGameObject = gameObject;
            pictureBox.Image = gameObject.Image;

        }
        public GameCell nextCell(GameDirection direction)
        {
            if (direction == GameDirection.Left)
            {
                if (this.col > 0)
                {
                    GameCell ncell = grid.getCell(row, col - 1);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Right)
            {
                if (this.col < grid.Cols - 1)
                {
                    GameCell ncell = grid.getCell(this.row, this.col + 1);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Up)
            {
                if (this.row > 0)
                {
                    GameCell ncell = grid.getCell(this.row - 1, this.col);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }

            if (direction == GameDirection.Down)
            {
                if (this.row < grid.Rows - 1)
                {
                    GameCell ncell = grid.getCell(this.row + 1, this.col);
                    if (ncell.CurrentGameObject.GameObjectType != GameObjectType.WALL)
                    {
                        return ncell;
                    }
                }
            }
            return this; // if can not return next cell return its own reference
        }
        public 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; }
    }
}
```

GameObject Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class GameObject
    {
        char displayCharacter;
        GameObjectType gameObjectType;
        GameCell currentCell;
        Image image;
        public GameObject(GameObjectType type, Image image)
        {
            this.gameObjectType = type;
            this.Image = image;
        }
        public GameObject(GameObjectType type, char displayCharacter)
        {
            this.gameObjectType = type;
            this.displayCharacter = displayCharacter;
        }

        public static GameObjectType getGameObjectType(char displayCharacter)
        {

            if (displayCharacter == '|' || 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 Enum:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public enum GameObjectType
    {
        WALL,
        PLAYER,
        ENEMY,
        FIRE,
        REWARD,
        NONE
    }
}
```

GameDirection Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public enum GameDirection
    {
        Left,
        Right,
        Up,
        Down
    }
}
```

GameFire Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class Fire : GameObject
    {
        GameDirection direction;
        public Fire(Image ghostImage, GameCell startCell, GameDirection D) :
base(GameObjectType.FIRE, ghostImage)
        {
            this.CurrentCell = startCell;
            direction = D;
        }

        public void move(GameCell gameCell)
        {
            if (this.CurrentCell != null)
            {
                this.CurrentCell.setGameObject(Game.getBlankGameObject());

            }

            CurrentCell = gameCell;

        }

        public GameCell nextCell()
        {

            GameCell nextCell = this.CurrentCell;

            GameCell potentialNextCell = this.CurrentCell.nextCell(direction);

            if (potentialNextCell == nextCell)
            {
                return nextCell;
            }
            else
            {
                nextCell = potentialNextCell;
            }
            return nextCell;
        }
    }
}
```

GameGhost Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public abstract class GameGhost : GameObject
    {
        public GameGhost(Image ghostImage) : base(GameObjectType.ENEMY, ghostImage)
        {
        }


        public abstract GameCell nextCell();
        public abstract void move(GameCell gameCell);
    }
}
```

GameGhostHorizontal Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    class GameGhostHorizontal : GameGhost
    {
        GameDirection direction = GameDirection.Left;

        public GameGhostHorizontal(Image ghostImage, GameCell startCell) :
base(ghostImage)
        {
            this.CurrentCell = startCell;
        }

        public override void move(GameCell gameCell)
        {
            if (this.CurrentCell != null)
            {
                this.CurrentCell.setGameObject(Game.getBlankGameObject());

            }
            CurrentCell = gameCell;
        }

        public override GameCell nextCell()
        {

            GameCell nextCell = this.CurrentCell;
            GameCell potentialNextCell = this.CurrentCell.nextCell(direction);

            if (potentialNextCell == nextCell)
            {
                if (direction == GameDirection.Left)
                {
                    direction = GameDirection.Right;
                }
                else if (direction == GameDirection.Right)
                {
                    direction = GameDirection.Left;
                }
            }
            else
            {
                nextCell = potentialNextCell;
            }
            return nextCell;
        }
    }
}
```

GameGhostVertical Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class GameGhostVertical : GameGhost
    {
        GameDirection direction = GameDirection.Down;

        public GameGhostVertical(Image ghostImage, GameCell startCell) :
base(ghostImage)
        {
            this.CurrentCell = startCell;
        }

        public override void move(GameCell gameCell)
        {
            if (this.CurrentCell != null)
            {
                this.CurrentCell.setGameObject(Game.getBlankGameObject());

            }
            CurrentCell = gameCell;
        }

        public override GameCell nextCell()
        {

            GameCell nextCell = this.CurrentCell;

            GameCell potentialNextCell = this.CurrentCell.nextCell(direction);

            if (potentialNextCell == nextCell)
            {
                if (direction == GameDirection.Up)
                {
                    direction = GameDirection.Down;
                }
                else if (direction == GameDirection.Down)
                {
                    direction = GameDirection.Up;
                }
            }
            else
            {
                nextCell = potentialNextCell;
            }
            return nextCell;
        }

    }
}
```

GameGhostChaser Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class GameGhostChaser : GameGhost
    {
        GameDirection direction = GameDirection.Left;
        GameObject Object;
        public GameGhostChaser(Image ghostImage, GameCell startCell, GameObject 
@object) : base(ghostImage)
        {
            this.CurrentCell = startCell;
            Object = @object;
        }

        public override void move(GameCell gameCell)
        {
            if (this.CurrentCell != null)
            {
                this.CurrentCell.setGameObject(Game.getBlankGameObject());

            }
            CurrentCell = gameCell;
        }

        public override GameCell nextCell()
        {

            GameCell nextCell = this.CurrentCell;

            GameCell potentialNextCell = this.CurrentCell.nextCell(direction);

            SetDirection();
            if (potentialNextCell != null)
            {

                nextCell = potentialNextCell;
            }
            return nextCell;
        }
        public void SetDirection()
        {
            double[] distancebf = new double[4] { 100000, 100000, 100000, 100000 };
            distancebf[0] = calculatedistancebf(Object.CurrentCell, 
CurrentCell.nextCell(GameDirection.Up));
            distancebf[1] = calculatedistancebf(Object.CurrentCell, 
CurrentCell.nextCell(GameDirection.Down));
            distancebf[2] = calculatedistancebf(Object.CurrentCell, 
CurrentCell.nextCell(GameDirection.Right));
            distancebf[3] = calculatedistancebf(Object.CurrentCell, 
CurrentCell.nextCell(GameDirection.Left));
            if (distancebf[0] == distancebf.Min())
            {
                direction = GameDirection.Up;
            }
            else if (distancebf[1] == distancebf.Min())
            {
                direction = GameDirection.Down;
            }
            else if (distancebf[2] == distancebf.Min())
            {
                direction = GameDirection.Right;
            }
            else if (distancebf[3] == distancebf.Min())
            {
                direction = GameDirection.Left;
            }
        }
        public double calculatedistancebf(GameCell cell1, GameCell cell2)
        {
            if (cell2 == null) { return 10000000; }
            return Math.Sqrt(Math.Pow(cell1.Y - cell2.Y, 2) + Math.Pow(cell1.X - 
cell2.X, 2));
        }
    }
}
```

PoliceManPlayer Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class GamePolicePlayer : GameObject
    {
        public GamePolicePlayer(Image image, GameCell startCell) : 
base(GameObjectType.PLAYER, image)
        {
            this.CurrentCell = startCell;
        }

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

    }
}
```

Game Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class Game
    {
        GamePolicePlayer policeman;
        GameGrid grid;
        public List<GameGhost> ghosts;
        int score = 0;
        int health = 100;
        List<Fire> Bullets = new List<Fire>();
        Form gameGUI;

        public List<Fire> Bullets1 { get => Bullets; set => Bullets = value; }
        public int Health { get => health; set => health = value; }

        public Game(Form gameGUI)
        {
            this.gameGUI = gameGUI;
            grid = new GameGrid("Maze.txt", 21, 70);
            Image policemanImage = Game.getGameObjectImage('P');
            ghosts = new List<GameGhost>();
            GameCell startCell = grid.getCell(8, 10);
            policeman = new GamePolicePlayer(policemanImage, startCell);
            printMaze(grid);

        }
        public void ProduceBullet(GameCell gameCell, GameDirection D)
        {
            if (gameCell.CurrentGameObject.GameObjectType == GameObjectType.NONE)
            {
                Fire f = new Fire(Properties.Resources.bullet, gameCell, D);
                addBullet(f);
            }
        }
        public GameCell getCell(int x, int y)
        {
            return grid.getCell(x, y);
        }
        public void addBullet(Fire F)
        {
            Bullets1.Add(F);
        }
        public void addGhost(GameGhost ghost)
        {
            ghosts.Add(ghost);
        }
        public void RemoveGhost(GameGhost ghost)
        {
            ghosts.Remove(ghost);
        }
        public GamePolicePlayer getpolicemanPlayer()
        {
            return policeman;
        }
        public void addScorePoints(int points)
        {
            this.score = score + points;
        }
        public void DecreaseHealth(int points)
        {

            this.Health = Health + points;


        }
        public int getScore()
        {
            return score;
        }
        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);
                    gameGUI.Controls.Add(cell.PictureBox);

                }

            }
        }

        public static GameObject getBlankGameObject()
        {
            GameObject blankGameObject = new GameObject(GameObjectType.NONE,
Properties.Resources.simplebox);
            return blankGameObject;
        }
        public Image getUpDown()
        {
            return Properties.Resources.upDownThief;
        }

        public Image getRightLeft()
        {
            return Properties.Resources.thief;
        }
        public Image getChaser()
        {
            return Properties.Resources.smart;
        }

        public static Image getGameObjectImage(char displayCharacter)
        {
            Image img = Properties.Resources.simplebox;

            if (displayCharacter == '|' || displayCharacter == '#')
            {
                img = Properties.Resources.vertical;
            }

            if (displayCharacter == '%')
            {
                img = Properties.Resources.zamin1;
            }

            if (displayCharacter == '.')
            {
                img = Properties.Resources.pallet;
            }
            if (displayCharacter == 'P' || displayCharacter == 'p')
            {
                img = Properties.Resources.pngegg__1_;
            }

            return img;
        }
    }
}
```

Game2 Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public class Game2
    {
        GamePolicePlayer policeman;
        GameGrid grid;
        public List<GameGhost> ghosts;
        int score = 0;
        int health = 100;
        List<Fire> Bullets = new List<Fire>();
        Form gameGUI;

        public List<Fire> Bullets1 { get => Bullets; set => Bullets = value; }
        public int Health { get => health; set => health = value; }

        public Game2(Form gameGUI)
        {
            this.gameGUI = gameGUI;
            grid = new GameGrid("Maze2.txt", 24, 70);
            Image policemanImage = Game.getGameObjectImage('P');
            ghosts = new List<GameGhost>();
            GameCell startCell = grid.getCell(8, 10);
            policeman = new GamePolicePlayer(policemanImage, startCell);
            printMaze(grid);

        }
        public void ProduceBullet(GameCell gameCell, GameDirection D)
        {
            if (gameCell.CurrentGameObject.GameObjectType == GameObjectType.NONE)
            {
                Fire f = new Fire(Properties.Resources.bullet, gameCell, D);
                addBullet(f);
            }
        }
        public GameCell getCell(int x, int y)
        {
            return grid.getCell(x, y);
        }
        public void addBullet(Fire F)
        {
            Bullets1.Add(F);
        }
        public void addGhost(GameGhost ghost)
        {
            ghosts.Add(ghost);
        }
        public void RemoveGhost(GameGhost ghost)
        {
            ghosts.Remove(ghost);
        }
        public GamePolicePlayer getpolicemanPlayer()
        {
            return policeman;
        }
        public void addScorePoints(int points)
        {
            this.score = score + points;
        }
        public void DecreaseHealth(int points)
        {

            this.Health = Health + points;


        }
        public int getScore()
        {
            return score;
        }
        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);
                    gameGUI.Controls.Add(cell.PictureBox);

                }

            }
        }

        public static GameObject getBlankGameObject()
        {
            GameObject blankGameObject = new GameObject(GameObjectType.NONE,
Properties.Resources.simplebox);
            return blankGameObject;
        }
        public Image getUpDown()
        {
            return Properties.Resources.upDownThief;
        }

        public Image getRightLeft()
        {
            return Properties.Resources.thief;
        }
        public Image getChaser()
        {
            return Properties.Resources.smart;
        }

        public static Image getGameObjectImage(char displayCharacter)
        {

            Image img = Properties.Resources.simplebox;


            if (displayCharacter == '|' || displayCharacter == '#')
            {
                img = Properties.Resources.vertical;
            }

            if (displayCharacter == '%')
            {
                img = Properties.Resources.zamin1;
            }

            if (displayCharacter == '.')
            {
                img = Properties.Resources.pallet;
            }
            if (displayCharacter == 'P' || displayCharacter == 'p')
            {
                img = Properties.Resources.pngegg__1_;
            }

            return img;
        }
    }
}
```

Game3 Class:

```
namespace PoliceManPlayer_GUI.GameGL
{
    class Game3
    {
        GamePolicePlayer policeman;
        GameGrid grid;
        public List<GameGhost> ghosts;
        int score = 0;
        int health = 100;
        List<Fire> Bullets = new List<Fire>();
        Form gameGUI;

        public List<Fire> Bullets1 { get => Bullets; set => Bullets = value; }
        public int Health { get => health; set => health = value; }

        public Game3(Form gameGUI)
        {
            this.gameGUI = gameGUI;
            grid = new GameGrid("Maze3.txt", 22, 60);
            Image policemanImage = Game.getGameObjectImage('P');
            ghosts = new List<GameGhost>();
            GameCell startCell = grid.getCell(8, 10);
            policeman = new GamePolicePlayer(policemanImage, startCell);
            printMaze(grid);

        }
        public void ProduceBullet(GameCell gameCell, GameDirection D)
        {
            if (gameCell.CurrentGameObject.GameObjectType == GameObjectType.NONE)
            {
                Fire f = new Fire(Properties.Resources.bullet, gameCell, D);
                addBullet(f);
            }
        }
        public GameCell getCell(int x, int y)
        {
            return grid.getCell(x, y);
        }
        public void addBullet(Fire F)
        {
            Bullets1.Add(F);
        }
        public void addGhost(GameGhost ghost)
        {
            ghosts.Add(ghost);
        }
        public void RemoveGhost(GameGhost ghost)
        {
            ghosts.Remove(ghost);
        }
        public GamePolicePlayer getpolicemanPlayer()
        {
            return policeman;
        }
        public void addScorePoints(int points)
        {
            this.score = score + points;
        }
        public void DecreaseHealth(int points)
        {

            this.Health = Health + points;


        }
        public int getScore()
        {
            return score;
        }
        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);
                    gameGUI.Controls.Add(cell.PictureBox);

                }

            }
        }

        public static GameObject getBlankGameObject()
        {
            GameObject blankGameObject = new GameObject(GameObjectType.NONE,
Properties.Resources.simplebox);
            return blankGameObject;
        }
        public Image getUpDown()
        {
            return Properties.Resources.upDownThief;
        }

        public Image getRightLeft()
        {
            return Properties.Resources.thief;
        }
        public Image getChaser()
        {
            return Properties.Resources.smart;
        }

        public static Image getGameObjectImage(char displayCharacter)
        {

            Image img = Properties.Resources.simplebox;


            if (displayCharacter == '|')
            {
                img = Properties.Resources.bricks;
            }

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

            if (displayCharacter == '%')
            {
                img = Properties.Resources.zamin1;
            }
            if (displayCharacter == '.')
            {
                img = Properties.Resources.pallet;
            }
            if (displayCharacter == 'P' || displayCharacter == 'p')
            {
                img = Properties.Resources.pngegg__1_;
            }

            return img;
        }
    }
}
```

StartScreenFrm Form:

```
namespace PoliceManPlayer_GUI
{
    public partial class StartScreenFrm : Form
    {
        public StartScreenFrm()
        {
            InitializeComponent();
        }

        private void button1_Click(object sender, EventArgs e)
        {
            this.Hide();
            Maze1Frm mf = new Maze1Frm();
            mf.Show();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            string op1 = "1.  Space Key Fore Firing if you moving left The Firing
going left and vice versa for right    " +
                "2.  Moving Left from Left Key    " +
                "3.  Moving Right from Right Key     " +
                "4.  Moving Up from Up Key    " +
                "5.  Moving Down from Down Key    " +
                "6.  Collect 3 Coins For next stage    " +
                "7.  Protect YourSelf From chasing Enemy if Enemy Chase you then you
health decreases    " +
                "8.  If chasing Enemy Fire the Coin then coin has been removed    ";
            MessageBox.Show(op1);
        }

        private void button3_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }
    }
}
```

Maze1Frm Form:

```
namespace PoliceManPlayer_GUI
{
    public partial class Maze1Frm : Form
    {
        Game game;
        GameCollisionDetector collider;
        GameDirection d = GameDirection.Right;
        public Maze1Frm()
        {
            InitializeComponent();
            game = new Game(this);
            collider = new GameCollisionDetector();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            GameGhostVertical gv1 = new GameGhostVertical(game.getUpDown(),
game.getCell(3, 6));

            GameGhostChaser gv2 = new GameGhostChaser(game.getChaser(),
game.getCell(3, 22), game.getpolicemanPlayer());
            GameGhostHorizontal gv3 = new GameGhostHorizontal(game.getRightLeft(),
game.getCell(8, 41));
            GameGhostHorizontal gv4 = new GameGhostHorizontal(game.getRightLeft(),
game.getCell(12, 48));
            GameGhostHorizontal gv5 = new GameGhostHorizontal(game.getRightLeft(),
game.getCell(3, 53));
            GameGhostChaser gv6 = new GameGhostChaser(game.getChaser(),
game.getCell(19, 53), game.getpolicemanPlayer());

            game.addGhost(gv1);
            game.addGhost(gv2);
            game.addGhost(gv3);
            game.addGhost(gv4);
            game.addGhost(gv5);
            game.addGhost(gv6);
        }

        public void moveGhosts()
        {
            foreach (GameGhost g in game.ghosts)
            {
                if (collider.isGhostCollideWithpoliceman(g))
                {
                    game.DecreaseHealth(-1);
                }
                g.move(g.nextCell());
            }
        }

        public void moveBullets()
        {
            for (int i = 0; i < game.Bullets1.Count; i++)
            {
                if (collider.isBulletCollideWithGhost(game.Bullets1[i]) != null)
                {
game.RemoveGhost(collider.isBulletCollideWithGhost(game.Bullets1[i]));
                }
                game.Bullets1[i].move(game.Bullets1[i].nextCell());
            }
        }

        public void RemoveBullets()
        {
            for (int i = 0; i < game.Bullets1.Count; i++)
            {
                if (game.Bullets1[i].nextCell() == game.Bullets1[i].CurrentCell)
                {
                    game.Bullets1[i].Image = Properties.Resources.simplebox;
                }
            }
        }

        public void GhostProduceBullet()
        {
            if (game.ghosts[1].CurrentCell.X ==
game.getpolicemanPlayer().CurrentCell.X)
            {
                game.ProduceBullet(game.ghosts[1].CurrentCell.nextCell(d), d);
            }
        }

        private void showScore()
        {

            lblScoreValue.Text = game.getScore().ToString();
            if (int.Parse(lblScoreValue.Text) == 3)
            {
                timer1.Stop();
                timer2.Stop();
                MessageBox.Show("YOU WIN... NEXT STAGE !!!");
                this.Hide();
                Maze2Frm f = new Maze2Frm();
                f.Show();
            }
        }

        private void showHealth()
        {

            label1.Text = game.Health.ToString();
            if (int.Parse(label1.Text) <= 0)
            {
                timer1.Stop();
                timer2.Stop();
                MessageBox.Show("YOU LOOSE...  GAME IS ENDED !!!");
                Thread.Sleep(1000);
                this.Hide();
                StartScreenFrm f = new StartScreenFrm();
                f.Show();
            }
        }

        private void movepoliceman()
        {
            GamePolicePlayer policeman = game.getpolicemanPlayer();
            GameCell potentialNewCell = policeman.CurrentCell;

            if (Keyboard.IsKeyPressed(Key.LeftArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Left);
                d = GameDirection.Left;
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Left))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.RightArrow))
            {
                potentialNewCell = 
policeman.CurrentCell.nextCell(GameDirection.Right);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Right))
                {
                    game.addScorePoints(1);
                }
                d = GameDirection.Right;
            }
            if (Keyboard.IsKeyPressed(Key.UpArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Up);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Up))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.DownArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Down);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Down))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.Space))
            {
                game.ProduceBullet(policeman.CurrentCell.nextCell(d), d);
            }


            GameCell currentCell = policeman.CurrentCell;
            currentCell.setGameObject(Game.getBlankGameObject());

            policeman.move(potentialNewCell);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            movepoliceman();
            showScore();
            showHealth();
            GhostProduceBullet();
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            moveGhosts();
            RemoveBullets();
            moveBullets();
        }
    }
}
```

Maze2Frm Form:

```
namespace PoliceManPlayer_GUI
{
    public partial class Maze2Frm : Form
    {
        Game2 game;
        GameCollisionDetector collider;
        GameDirection d = GameDirection.Right;
        public Maze2Frm()
        {
            InitializeComponent();
            game = new Game2(this);
            collider = new GameCollisionDetector();
        }

        private void Form2_Load(object sender, EventArgs e)
        {
            GameGhostVertical gv1 = new GameGhostVertical(game.getUpDown(),
game.getCell(2, 2));

            GameGhostChaser gv2 = new GameGhostChaser(game.getChaser(),
game.getCell(3, 22), game.getpolicemanPlayer());
            GameGhostHorizontal gv3 = new GameGhostHorizontal(game.getRightLeft(),
game.getCell(8, 41));
            GameGhostHorizontal gv4 = new GameGhostHorizontal(game.getRightLeft(),
game.getCell(22, 67));
            GameGhostHorizontal gv5 = new GameGhostHorizontal(game.getRightLeft(),
game.getCell(1, 53));

            game.addGhost(gv1);
            game.addGhost(gv2);
            game.addGhost(gv3);
            game.addGhost(gv4);
            game.addGhost(gv5);
        }

        public void moveGhosts()
        {
            foreach (GameGhost g in game.ghosts)
            {
                if (collider.isGhostCollideWithpoliceman(g))
                {
                    game.DecreaseHealth(-1);
                }
                g.move(g.nextCell());


            }
        }

        public void moveBullets()
        {
            for (int i = 0; i < game.Bullets1.Count; i++)
            {
                if (collider.isBulletCollideWithGhost(game.Bullets1[i]) != null)
                {

game.RemoveGhost(collider.isBulletCollideWithGhost(game.Bullets1[i]));

                }
                game.Bullets1[i].move(game.Bullets1[i].nextCell());


            }
        }

        public void RemoveBullets()
        {
            for (int i = 0; i < game.Bullets1.Count; i++)
            {
                if (game.Bullets1[i].nextCell() == game.Bullets1[i].CurrentCell)
                {
                    game.Bullets1[i].Image = Properties.Resources.simplebox;
                }
            }
        }

        public void GhostProduceBullet()
        {
            if (game.ghosts[1].CurrentCell.X ==
game.getpolicemanPlayer().CurrentCell.X)
            {
                game.ProduceBullet(game.ghosts[1].CurrentCell.nextCell(d), d);
            }
        }

        private void showScore()
        {

            lblScore.Text = game.getScore().ToString();
            if (int.Parse(lblScore.Text) == 3)
            {
                timer1.Stop();
                timer2.Stop();
                MessageBox.Show("YOU WIN !!!");
                this.Hide();
                StartScreenFrm f = new StartScreenFrm();
                f.Show();
            }
        }

        private void showHealth()
        {

            label1.Text = game.Health.ToString();
            if (int.Parse(label1.Text) <= 0)
            {
                timer1.Stop();
                timer2.Stop();
                MessageBox.Show("YOU LOOSE... GAME IS ENDED !!!");
                this.Hide();
                StartScreenFrm f = new StartScreenFrm();
                f.Show();
            }

        }

        private void movepoliceman()
        {
            GamePolicePlayer policeman = game.getpolicemanPlayer();
            GameCell potentialNewCell = policeman.CurrentCell;

            if (Keyboard.IsKeyPressed(Key.LeftArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Left);
                d = GameDirection.Left;
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Left))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.RightArrow))
            {
                potentialNewCell = 
policeman.CurrentCell.nextCell(GameDirection.Right);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Right))
                {
                    game.addScorePoints(1);
                }
                d = GameDirection.Right;
            }
            if (Keyboard.IsKeyPressed(Key.UpArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Up);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Up))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.DownArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Down);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Down))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.Space))
            {
                game.ProduceBullet(policeman.CurrentCell.nextCell(d), d);
            }


            GameCell currentCell = policeman.CurrentCell;
            currentCell.setGameObject(Game.getBlankGameObject());

            policeman.move(potentialNewCell);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            movepoliceman();
            showScore();
            showHealth();
            GhostProduceBullet();
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            moveGhosts();
            RemoveBullets();
            moveBullets();
        }
    }
}
```

Maze3Frm Form:

```
namespace PoliceManPlayer_GUI.GameGL
{
    public partial class Maze3Frm : Form
    {
        Game3 game;
        GameCollisionDetector collider;
        GameDirection d = GameDirection.Right;
        public Maze3Frm()
        {
            InitializeComponent();
            game = new Game3(this);
            collider = new GameCollisionDetector();
        }

        private void Maze3Frm_Load(object sender, EventArgs e)
        {
            GameGhostVertical gv1 = new GameGhostVertical(game.getUpDown(),
game.getCell(3, 6));
            GameGhostVertical gv11 = new GameGhostVertical(game.getUpDown(),
game.getCell(6, 55));
```

```
            GameGhostChaser gv2 = new GameGhostChaser(game.getChaser(), 
game.getCell(1, 12), game.getpolicemanPlayer());
            GameGhostHorizontal gv3 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(10, 15));
            GameGhostHorizontal gv4 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(7, 17));
            GameGhostHorizontal gv5 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(11, 25));
            GameGhostHorizontal gv7 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(14, 20));
            GameGhostHorizontal gv8 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(3, 5));
            GameGhostHorizontal gv9 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(17, 20));
            GameGhostHorizontal gv10 = new GameGhostHorizontal(game.getRightLeft(), 
game.getCell(17, 31));
            GameGhostChaser gv6 = new GameGhostChaser(game.getChaser(), 
game.getCell(20, 24), game.getpolicemanPlayer());

            game.addGhost(gv1);
            game.addGhost(gv2);
            game.addGhost(gv3);
            game.addGhost(gv4);
            game.addGhost(gv5);
            game.addGhost(gv6);
            game.addGhost(gv7);
            game.addGhost(gv8);
            game.addGhost(gv9);
            game.addGhost(gv10);
            game.addGhost(gv11);
        }

        public void moveGhosts()
        {
            foreach (GameGhost g in game.ghosts)
            {
                if (collider.isGhostCollideWithpoliceman(g))
                {
                    game.DecreaseHealth(-1);
                }
                g.move(g.nextCell());
            }
        }

        public void moveBullets()
        {
            for (int i = 0; i < game.Bullets1.Count; i++)
            {
                if (collider.isBulletCollideWithGhost(game.Bullets1[i]) != null)
                {

game.RemoveGhost(collider.isBulletCollideWithGhost(game.Bullets1[i]));

                }
                game.Bullets1[i].move(game.Bullets1[i].nextCell());
            }
        }

        public void RemoveBullets()
        {
            for (int i = 0; i < game.Bullets1.Count; i++)
            {
                if (game.Bullets1[i].nextCell() == game.Bullets1[i].CurrentCell)
                {
                    game.Bullets1[i].Image = Properties.Resources.simplebox;
                }
            }
        }

        public void GhostProduceBullet()
        {
            if (game.ghosts[1].CurrentCell.X ==
game.getpolicemanPlayer().CurrentCell.X)
            {
                game.ProduceBullet(game.ghosts[1].CurrentCell.nextCell(d), d);
            }
        }

        private void showScore()
        {

            lblScoreValue.Text = game.getScore().ToString();
            if (int.Parse(lblScoreValue.Text) == 3)
            {
                timer1.Stop();
                timer2.Stop();
                MessageBox.Show("Congratulations... YOU WIN ALL STAGES !!!");
                this.Hide();
                StartScreenFrm f = new StartScreenFrm();
                f.Show();
            }
        }

        private void showHealth()
        {

            label1.Text = game.Health.ToString();
            if (int.Parse(label1.Text) <= 0)
            {
                timer1.Stop();
                timer2.Stop();
                MessageBox.Show("YOU LOOSE...  GAME IS ENDED !!!");
                Thread.Sleep(1000);
                this.Hide();
                StartScreenFrm f = new StartScreenFrm();
                f.Show();
            }

        }

        private void movepoliceman()
        {
            GamePolicePlayer policeman = game.getpolicemanPlayer();
            GameCell potentialNewCell = policeman.CurrentCell;

            if (Keyboard.IsKeyPressed(Key.LeftArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Left);
                d = GameDirection.Left;
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Left))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.RightArrow))
            {
                potentialNewCell = 
policeman.CurrentCell.nextCell(GameDirection.Right);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Right))
                {
                    game.addScorePoints(1);
                }
                d = GameDirection.Right;
            }
            if (Keyboard.IsKeyPressed(Key.UpArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Up);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Up))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.DownArrow))
            {
                potentialNewCell = policeman.CurrentCell.nextCell(GameDirection.Down);
                if (collider.ispolicemanCollideWithPallet(potentialNewCell, 
GameDirection.Down))
                {
                    game.addScorePoints(1);
                }
            }
            if (Keyboard.IsKeyPressed(Key.Space))
            {
                game.ProduceBullet(policeman.CurrentCell.nextCell(d), d);
            }


            GameCell currentCell = policeman.CurrentCell;
            currentCell.setGameObject(Game.getBlankGameObject());

            policeman.move(potentialNewCell);
        }

        private void timer1_Tick(object sender, EventArgs e)
        {
            movepoliceman();
            showScore();
            showHealth();
            GhostProduceBullet();
        }

        private void timer2_Tick(object sender, EventArgs e)
        {
            moveGhosts();
            RemoveBullets();
            moveBullets();
        }
    }
}
```

InstrunctionFrm Form:

```
namespace PoliceManPlayer_GUI
{
    public partial class InstructionFrm : Form
    {
        public InstructionFrm()
        {
            InitializeComponent();
        }

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

THE END ☺