**Car Hurdle 2D Game**



Session: 2022 – 2026

**Submitted by:**

Muhammad Taha Saleem 2022-CS-139

**Supervised by:**

Prof. Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents**

**Introduction……………………………………………………….….………...……2-6**

**Wire Frames…........……………………………………………………..……..........7-9**

**Class Responsibility Collaboration Card……………………..………..….……..…10**

**Object Oriented Programming…………………………………….….………….…10**

**Design Pattern Implementation..…………………………………………….……...11**

**Conclusion……........……………………………………………………..…….....…..12**

**Code……........…………………………………………………………..….......….13-43**

## Short Description and Story of the Car Game

Basically my Game contains 4 Games in which Car Game is my main game in which there are total 5 cars. One is Player Car and other 4 cars are enemy Cars. (1) Random (2) Smart (3) Horizontal (4) Vertical enemy Cars. Player car and Horizontal Car are capable of Firing Bullets on the Other hand Smart car is capable of chasing Player car. There are 3 lives of player and its task is to destroy Horizontal car to win.

## YouTube Link

<https://youtu.be/uLZWHazYfYA>

## Game Characters Description

### Player:

There is one human player in the Game.

**Car:**

There is a Player Car in the game.

### Enemies

There are many enemy cars of three types in the game:

1. There is an enemy car that moves vertically without shooting.
2. There is an enemy car that moves horizontally with shooting
3. There is a Random Car.
4. There is finally a Smart Car.

## Game Objects Description

Following are the Objects in the Game

### Walls:

Walls are the barriers in the game which the Player car and the enemy cars cannot cross.

## Shooting System:

* The Player-Car will be able to shoot.
* The Horizontal enemy Car will be able to shoot in more than one direction.

## Rules & Interactions

Player Car can win the game by destroying Horizontal Car while keeping itself safe from other cars within its life span.

## Goal of the Game

The goal of the game is to Destroy Horizontal Car.

## 2. Short Description and Story of the Balloon Game

Basically in my Game there is a player with arrow and bow. His task is to shoot the balloons by his arrow till he wins.

## Game Characters Description

### Player:

There is one human player in the Game.

**Car:**

There is a Player with Bow and Arrow in the game.

## Game Objects Description

Following are the Objects in the Game

### Balloons:

Balloons are the objects in the Game which are being generated randomly.

## Shooting System:

* The Player-Car will be able to shoot balloons.

## Rules & Interactions

Player Car can win the game by reaching 50 score.

## Goal of the Game

The goal of the game is to Destroy Balloons.

## 3. Short Description and Story of the Whirly Ghost

Basically in my Game there is a player with Left Right movement. His task is to cross hurdles which are being randomly generated.

## Game Characters Description

### Player:

There is one human player in the Game.

**Car:**

There is a Player with Left Right movement in the game.

## Game Objects Description

Following are the Objects in the Game

### Hurdles:

Hurdles are arrow like objects in the Game which are being generated randomly.

## Shooting System:

* No shooting system in this game.

## Rules & Interactions

Player Car can win the game by crossing 35 randomly generated hurdles.

## Goal of the Game

The goal of the game is to Cross 35 Hurdles.

## 3. Short Description and Story of the Reach the Castle Game

Basically in my Game there is a player who can run and jump. His task is to run on the randomly generated hurdles by keeping itself safe from falling down on the ground. He can jump to move onto the hurdles which are at higher in position from current hurdle and again falling down with gravity. Player can win the game by moving onto the hurdles and eating the pallets so as to reach score 50 and the hidden Castle will be visible and finally after crossing some hurdles he will be able to reach Castle.

## Game Characters Description

### Player:

There is one human player in the Game.

**Car:**

There is a Player.

## Game Objects Description

Following are the Objects in the Game

### Hurdles:

Hurdles are the objects in the Game which are being generated randomly and the player has to move onto those hurdles.

### Pallets:

Pallets are the objects in the Game which are being generated randomly and which cause the score increase.

## Shooting system:

* No shooting system in the game.

## Rules & Interactions

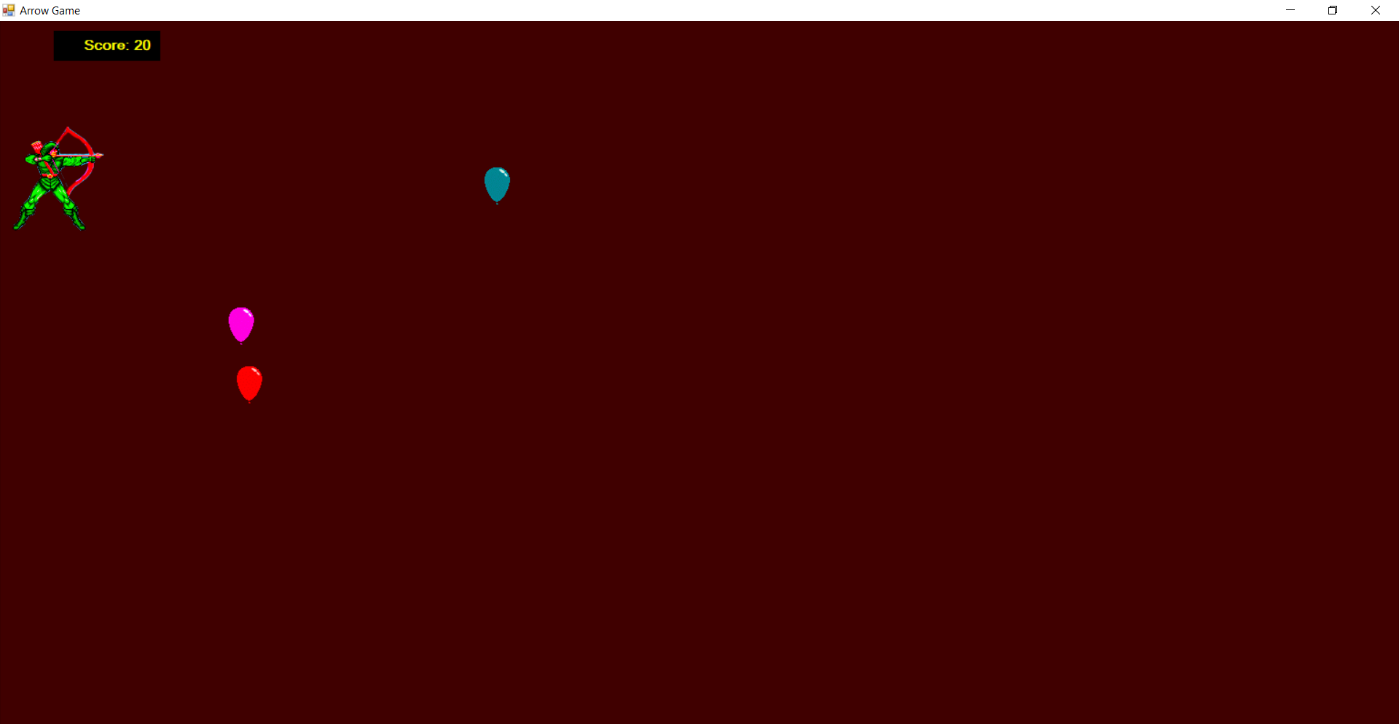
Player can win the game by moving onto the hurdles and eating the pallets so as to reach score 50 and the hidden Castle will be visible and finally after crossing some hurdles he will be able to reach Castle.

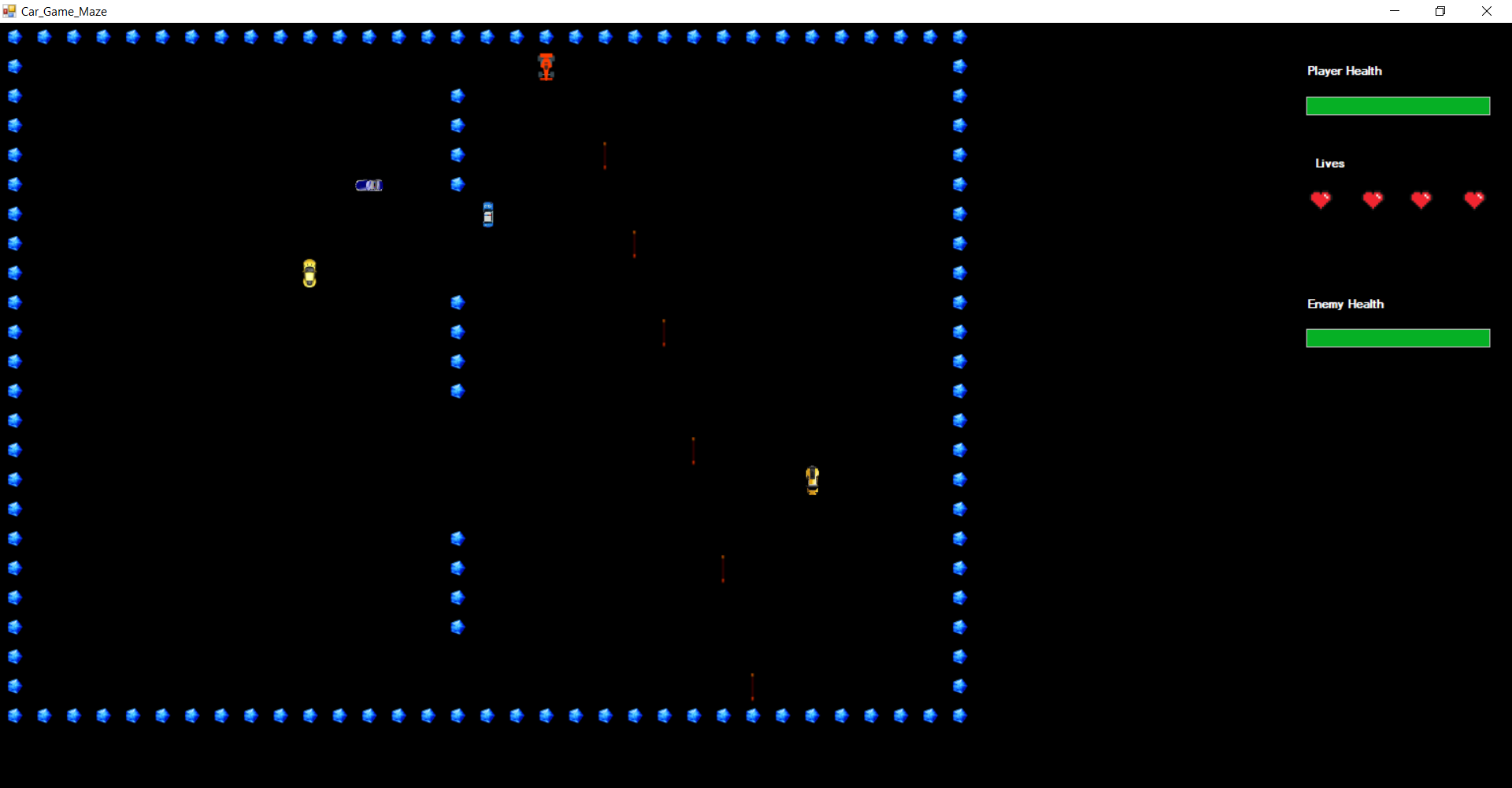
## Goal of the Game

The goal of the game is to reach the Castle.

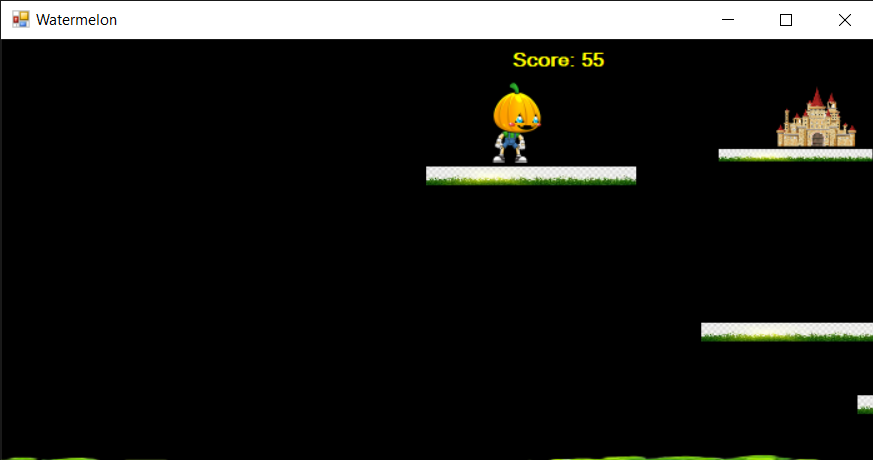
**Wire Frames:**

****

****

****

****

****

****

**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 Car 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.

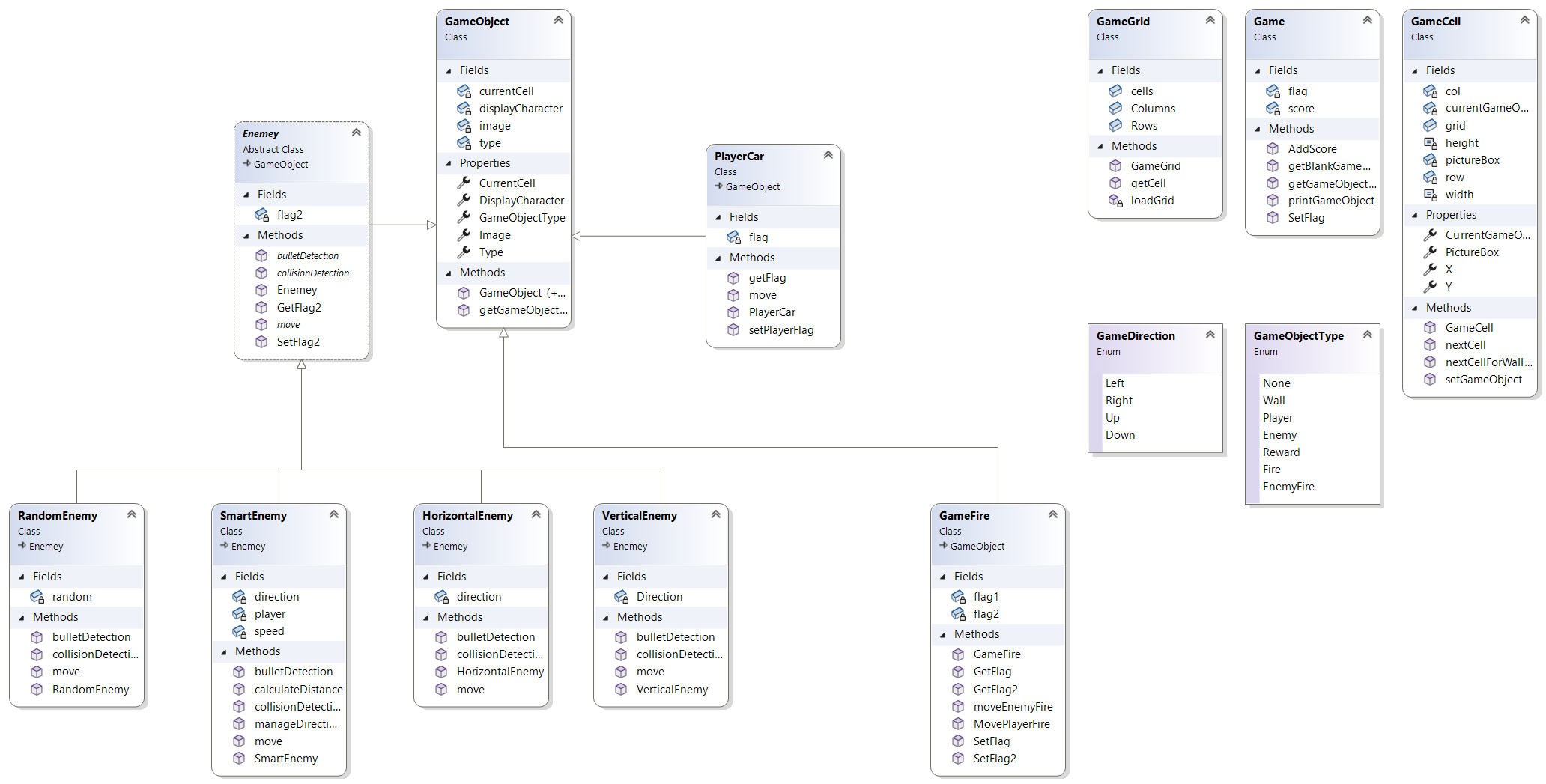
**Abstraction:**

I have implemented many Abstract methods and classes like collision detection and abstract Enemy class.

**Enumeration:**

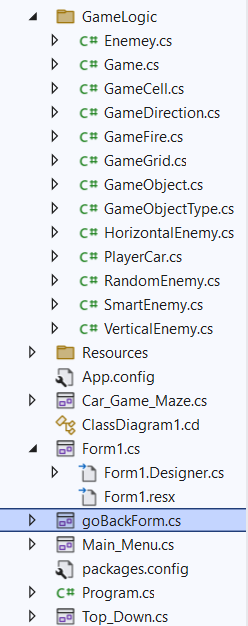
I have also implemented enumeration classes in my Game.

**Class Diagram:**

****

* **Design Pattern Implementation**

The directory structure for the project is given below:

****

**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:

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using EZInput;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

public class GameGrid

{

public int Rows;

public int Columns;

public GameCell[,] cells;

public GameGrid(string filename,int rows, int columns)

{

Rows = rows;

Columns = columns;

cells = new GameCell[Rows, Columns];

loadGrid(filename);

}

private void loadGrid(string filename)

{

StreamReader fp = new StreamReader(filename);

string record;

for (int row = 0; row < this.Rows; row++)

{

record = fp.ReadLine();

for (int col = 0; col < this.Columns; col++)

{

GameCell cell = new GameCell(row, col, this);

Char displayCharacter = record[col];

GameObjectType t = GameObject.getGameObjectType(displayCharacter);

Image displayIamge = Game.getGameObjectImage(displayCharacter);

GameObject gameObject = new GameObject(t, displayIamge);

cell.setGameObject(gameObject);

cells[row, col] = cell;

}

}

fp.Close();

}

public GameCell getCell(int x, int y)

{

return cells[x, y];

}

}

}

GameCell Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

public class GameCell

{

private int row;

private int col;

private GameObject currentGameObject;

PictureBox pictureBox;

public GameGrid grid;

const int width = 30;

const int height = 30;

public int X { get => row; set => row = value; }

public int Y { get => col; set => col = value; }

public GameObject CurrentGameObject { get => currentGameObject; set => currentGameObject = value; }

public PictureBox PictureBox { get => pictureBox; set => pictureBox = value; }

public GameCell(int x, int y, GameGrid grid)

{

this.X = x;

this.Y = y;

PictureBox = new PictureBox();

PictureBox.Left = width\*y;

PictureBox.Top = height\*x;

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.Y > 0)

{

GameCell ncell = grid.getCell(X, Y - 1);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

}

if (direction == GameDirection.Right)

{

if (this.Y < grid.Columns - 1)

{

GameCell ncell = grid.getCell(this.X, this.Y + 1);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

}

if (direction == GameDirection.Up)

{

if (this.X > 0)

{

GameCell ncell = grid.getCell(this.X - 1, this.Y);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

}

if (direction == GameDirection.Down)

{

if (this.X < grid.Rows - 1)

{

GameCell ncell = grid.getCell(this.X + 1, this.Y);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

}

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

}

public GameCell nextCellForWallBreaker(GameDirection direction)

{

if (direction == GameDirection.Left)

{

GameCell ncell = grid.getCell(X, Y - 1);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

if (this.Y < grid.Columns - 1)

{

GameCell ncell = grid.getCell(this.X, this.Y + 1);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

if (direction == GameDirection.Up)

{

if (this.X > 0)

{

GameCell ncell = grid.getCell(this.X - 1, this.Y);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

}

if (this.X < grid.Rows - 1)

{

GameCell ncell = grid.getCell(this.X + 1, this.Y);

if (ncell.CurrentGameObject.Type != GameObjectType.Wall)

{

return ncell;

}

}

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

}

}

}

GameObject Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

public class GameObject

{

private char displayCharacter;

private GameObjectType type;

private GameCell currentCell;

private Image image;

public GameObject(GameObjectType type, Image image)

{

this.Type = type;

this.Image = image;

}

public GameObject(char displayCharacter, GameObjectType gameObjectType)

{

this.displayCharacter = displayCharacter;

this.Type = gameObjectType;

}

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 => Type; set => Type = value; }

public GameCell CurrentCell

{

get => currentCell;

set

{

currentCell = value;

currentCell.setGameObject(this);

}

}

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

public GameObjectType Type { get => type; set => type = value; }

}

}

GameObjectType Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

namespace My\_Car\_Game\_GUI.GameLogic

{

public enum GameObjectType

{

None,Wall,Player,Enemy,Reward,Fire, EnemyFire

}

}

GameDirection Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace My\_Car\_Game\_GUI.GameLogic

{

public enum GameDirection

{

Left,Right,Up,Down

}

}

GameFire Class:

using System;

using System.Collections.Generic;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

internal class GameFire : GameObject

{

bool flag1 = true;

bool flag2 = true;

public GameFire(GameCell startCell, Image image,GameObjectType type) : base(type, image)

{

this.CurrentCell = startCell;

this.Image = image;

this.Type = type;

}

public GameCell moveEnemyFire(GameGrid grid)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = grid.getCell(CurrentCell.X + 1, CurrentCell.Y);

if (nextCell.CurrentGameObject.Type != GameObjectType.Wall)

{

if (nextCell!=null)

{

currentCell.setGameObject(Game.getBlankGameObject());

CurrentCell = nextCell;

return nextCell;

}

}

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.Wall)

{

SetFlag();

}

return null;

}

public GameCell MovePlayerFire(GameGrid grid)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = grid.getCell(CurrentCell.X - 1, CurrentCell.Y);

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.Enemy)

{

SetFlag2();

}

if (nextCell.CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

if (nextCell != null)

{

currentCell.setGameObject(Game.getBlankGameObject());

CurrentCell = nextCell;

return nextCell;

}

}

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.Wall)

{

SetFlag();

}

return null;

}

public void SetFlag2()

{

flag2 = false;

}

public bool GetFlag2()

{

return flag2;

}

public void SetFlag()

{

flag1 = false;

}

public bool GetFlag()

{

return flag1;

}

}

}

Enemy Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

abstract class Enemey :GameObject

{

bool flag2;

public Enemey(Image image, GameCell Cell) : base(GameObjectType.Enemy, image)

{

}

public abstract GameCell move();

public abstract bool collisionDetection(PlayerCar player);

public abstract bool bulletDetection(PlayerCar player);

public void SetFlag2()

{

flag2 = false;

}

public bool GetFlag2()

{

return flag2;

}

}

}

PlayerCar Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

internal class PlayerCar : GameObject

{

bool flag = true;

public PlayerCar(GameCell startCell, Image image) : base(GameObjectType.Player, image)

{

this.CurrentCell = startCell;

flag = true;

}

public GameCell move(GameDirection direction)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.Reward)

{

Game.AddScore();

}

if (CurrentCell.nextCell(GameDirection.Up).CurrentGameObject.GameObjectType == GameObjectType.EnemyFire || CurrentCell.nextCell(GameDirection.Right).CurrentGameObject.GameObjectType == GameObjectType.EnemyFire

|| CurrentCell.nextCell(GameDirection.Left).CurrentGameObject.GameObjectType == GameObjectType.EnemyFire || CurrentCell.nextCell(GameDirection.Down).CurrentGameObject.GameObjectType == GameObjectType.EnemyFire)

{

setPlayerFlag(false);

}

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return nextCell;

}

public void setPlayerFlag(bool flag)

{

this.flag = flag;

}

public bool getFlag()

{

return this.flag;

}

}

}

Game Class:

using My\_Car\_Game\_GUI.GameLogic;

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Drawing;

namespace My\_Car\_Game\_GUI

{

public class Game

{

static int score = 0;

static bool flag = true;

public static void printGameObject(GameObject gameObject)

{

Console.SetCursorPosition(gameObject.CurrentCell.Y, gameObject.CurrentCell.X);

Console.Write(gameObject.DisplayCharacter);

}

public static GameObject getBlankGameObject()

{

GameObject blankGameObject = new GameObject(GameObjectType.None, My\_Car\_Game\_GUI.Properties.Resources.simplebox);

return blankGameObject;

}

public static Image getGameObjectImage(char displayCharacter)

{

Image img = My\_Car\_Game\_GUI.Properties.Resources.simplebox;

if (displayCharacter == '#' || displayCharacter == '|' || displayCharacter == '-')

{

img = My\_Car\_Game\_GUI.Properties.Resources.\_1;

}

if (displayCharacter == 'l')

{

img = My\_Car\_Game\_GUI.Properties.Resources.arrowLeft;

}

if (displayCharacter == 'u')

{

img = My\_Car\_Game\_GUI.Properties.Resources.arrowUp;

}

if (displayCharacter == 'd')

{

img = My\_Car\_Game\_GUI.Properties.Resources.arrowDown;

}

if (displayCharacter == 'C' || displayCharacter == 'c')

{

img = My\_Car\_Game\_GUI.Properties.Resources.car;

}

if (displayCharacter == 'P' || displayCharacter == 'p')

{

img = My\_Car\_Game\_GUI.Properties.Resources.police;

}

if (displayCharacter == 'H' || displayCharacter == 'h')

{

img = My\_Car\_Game\_GUI.Properties.Resources.horCar2;

}

if (displayCharacter == 'V' || displayCharacter == 'v')

{

img = My\_Car\_Game\_GUI.Properties.Resources.vertical1;

}

if (displayCharacter == 'R' || displayCharacter == 'r')

{

img = My\_Car\_Game\_GUI.Properties.Resources.rand;

}

return img;

//return null;

}

public static void SetFlag()

{

flag = false;

}

public static void AddScore()

{

score++;

}

}

}

Horizontal Enemy Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

internal class HorizontalEnemy : Enemey

{

GameDirection direction;

public HorizontalEnemy(Image image, GameCell cell, GameDirection d) : base(image, cell)

{

this.direction = d;

this.CurrentCell = cell;

}

public override GameCell move()

{

GameCell current = this.CurrentCell;

GameCell next = CurrentCell.nextCell(direction);

this.CurrentCell = next;

if (current == next)

{

if (this.direction == GameDirection.Left)

{

this.direction = GameDirection.Right;

}

else if (this.direction == GameDirection.Right)

{

this.direction = GameDirection.Left;

}

}

else

{

current.setGameObject(Game.getBlankGameObject());

}

return next;

}

public override bool collisionDetection(PlayerCar player)

{

bool flag = false;

if (player.CurrentCell.X == this.CurrentCell.X && player.CurrentCell.Y == this.CurrentCell.Y)

{

flag = true;

}

return flag;

}

public override bool bulletDetection(PlayerCar player)

{

bool flag = false;

if (CurrentCell.nextCell(GameDirection.Down).CurrentGameObject.GameObjectType == GameObjectType.Fire || CurrentCell.nextCell(GameDirection.Right).CurrentGameObject.GameObjectType == GameObjectType.Fire || CurrentCell.nextCell(GameDirection.Left).CurrentGameObject.GameObjectType == GameObjectType.Fire)

{

flag = true;

}

return flag;

}

}

}

Random Enemy Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

internal class RandomEnemy : Enemey

{

int random;

public RandomEnemy(Image image, GameCell cell) : base(image, cell)

{

this.CurrentCell = cell;

this.CurrentCell.PictureBox.Height = 20;

this.CurrentCell.PictureBox.Width = 20;

}

public override GameCell move()

{

Random rand = new Random();

random = rand.Next(4);

if (random == 0)

{

GameCell current = this.CurrentCell;

GameCell next = current.nextCell(GameDirection.Left);

if (next.CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

this.CurrentCell = next;

if (next != current)

{

current.setGameObject(Game.getBlankGameObject());

}

}

}

else if (random == 1)

{

GameCell current = this.CurrentCell;

GameCell next = current.nextCell(GameDirection.Right);

if (next.CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

this.CurrentCell = next;

if (next != current)

{

current.setGameObject(Game.getBlankGameObject());

}

}

}

else if (random == 2)

{

GameCell current = this.CurrentCell;

GameCell next = current.nextCell(GameDirection.Up);

if (next.CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

this.CurrentCell = next;

if (next != current)

{

current.setGameObject(Game.getBlankGameObject());

}

}

}

else if (random == 3)

{

GameCell current = this.CurrentCell;

GameCell next = current.nextCell(GameDirection.Down);

if (next.CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

this.CurrentCell = next;

if (next != current)

{

current.setGameObject(Game.getBlankGameObject());

}

}

}

return null;

}

public override bool collisionDetection(PlayerCar player)

{

bool flag = false;

if (GameObjectType.Player == this.CurrentCell.CurrentGameObject.Type)

{

flag = true;

}

return flag;

}

public override bool bulletDetection(PlayerCar player)

{

return false;

}

}

}

Vertical Enemy Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

internal class VerticalEnemy : Enemey

{

GameDirection Direction;

public VerticalEnemy(Image image, GameCell cell, GameDirection d) : base(image, cell)

{

this.Direction = d;

this.CurrentCell = cell;

this.CurrentCell.PictureBox.Height = 20;

this.CurrentCell.PictureBox.Width = 20;

}

public override GameCell move()

{

GameCell current = this.CurrentCell;

GameCell next = CurrentCell.nextCell(Direction);

this.CurrentCell = next;

if (current == next)

{

if (this.Direction == GameDirection.Up)

{

this.Direction = GameDirection.Down;

}

else if (this.Direction == GameDirection.Down)

{

this.Direction = GameDirection.Up;

}

}

if (current != next)

{

current.setGameObject(Game.getBlankGameObject());

}

return next;

}

public override bool collisionDetection(PlayerCar player)

{

bool flag = false;

if (player.CurrentCell.X == this.CurrentCell.X && player.CurrentCell.Y == this.CurrentCell.Y)

{

flag = true;

}

return flag;

}

public override bool bulletDetection(PlayerCar player)

{

return false;

}

}

}

Smart Enemy Class:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

namespace My\_Car\_Game\_GUI.GameLogic

{

internal class SmartEnemy : Enemey

{

GameDirection direction;

PlayerCar player;

int speed;

public SmartEnemy(Image image, GameCell cell, int speed, PlayerCar player) : base(image, cell)

{

this.CurrentCell = cell;

this.speed = speed;

this.player = player;

}

public override GameCell move()

{

if (speed % 1029 == 0)

{

manageDirections();

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return nextCell;

}

speed++;

return this.CurrentCell;

}

public void manageDirections()

{

double[] distance = new double[4] { 10000, 10000, 10000, 10000 };

if (this.CurrentCell.nextCell(GameDirection.Left).CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

distance[0] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Left));

}

if (this.CurrentCell.nextCell(GameDirection.Right).CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

distance[1] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Right));

}

if (this.CurrentCell.nextCell(GameDirection.Up).CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

distance[2] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Up));

}

if (this.CurrentCell.nextCell(GameDirection.Down).CurrentGameObject.GameObjectType != GameObjectType.Wall)

{

distance[3] = calculateDistance(this.CurrentCell.nextCell(GameDirection.Down));

}

if (distance[0] <= distance[1] && distance[0] <= distance[2] && distance[0] <= distance[3])

{

this.direction = GameDirection.Left;

}

if (distance[1] <= distance[0] && distance[1] <= distance[2] && distance[1] <= distance[3])

{

this.direction = GameDirection.Right;

}

if (distance[2] <= distance[0] && distance[2] <= distance[1] && distance[2] <= distance[3])

{

this.direction = GameDirection.Up;

}

if (distance[3] <= distance[0] && distance[3] <= distance[1] && distance[3] <= distance[2])

{

this.direction = GameDirection.Down;

}

}

public double calculateDistance(GameCell nextcell)

{

return Math.Sqrt(Math.Pow((player.CurrentCell.X - nextcell.X), 2) + Math.Pow((player.CurrentCell.Y - nextcell.Y), 2));

}

public override bool collisionDetection(PlayerCar player)

{

bool flag = false;

if (player.CurrentCell.X == this.CurrentCell.X && player.CurrentCell.Y == this.CurrentCell.Y)

{

flag = true;

}

return flag;

}

public override bool bulletDetection(PlayerCar player)

{

return false;

}

}

}

Car Game Form CS file:

using EZInput;

using My\_Car\_Game\_GUI.GameLogic;

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace My\_Car\_Game\_GUI

{

public partial class Car\_Game\_Maze : Form

{

GameGrid grid;

List<Enemey> enemyCars = new List<Enemey>();

List<GameFire> fires = new List<GameFire>();

List<GameFire> enemyFires = new List<GameFire>();

HorizontalEnemy Horizontal;

VerticalEnemy Vertical;

RandomEnemy Random;

SmartEnemy Smart;

PlayerCar player;

int playerFireCounter = 0;

int playerLivesCount = 4;

GameCell playerFireCell;

GameFire playerFire;

GameCell horizontalEnemy1FireCell;

GameFire horizontalEnemy1Fire;

int horizontalEnemyFireCounter = 0;

public Car\_Game\_Maze()

{

InitializeComponent();

gameOverLBL.Visible= false;

}

private void Car\_Game\_Maze\_Load(object sender, EventArgs e)

{

grid = new GameGrid("carMaze.txt", 24, 33);

Image playerCar = Game.getGameObjectImage('c');

Image verticalEnemyCar = Game.getGameObjectImage('v');

Image horizontalEnemyCar = Game.getGameObjectImage('h');

Image randomEnemyCar = Game.getGameObjectImage('r');

Image smartEnemyCar = Game.getGameObjectImage('p');

GameCell startCell = grid.getCell(8, 10);

GameCell Vertical\_Enemy\_Cell = grid.getCell(3, 27);

GameCell Horizontal\_Enemy\_Cell = grid.getCell(1, 30);

GameCell Random\_Enemy\_Cell = grid.getCell(7, 14);

GameCell Smart\_Enemy\_Cell = grid.getCell(3, 25);

player = new PlayerCar(startCell, playerCar);

Horizontal = new HorizontalEnemy(horizontalEnemyCar, Horizontal\_Enemy\_Cell, GameDirection.Left);

Vertical = new VerticalEnemy(verticalEnemyCar, Vertical\_Enemy\_Cell, GameDirection.Down);

Random = new RandomEnemy(randomEnemyCar, Random\_Enemy\_Cell);

Smart = new SmartEnemy(smartEnemyCar, Smart\_Enemy\_Cell, 0, player);

enemyCars.Add(Horizontal);

enemyCars.Add(Vertical);

enemyCars.Add(Random);

enemyCars.Add(Smart);

printMaze(grid);

}

void printMaze(GameGrid gameGrid)

{

for (int i = 0; i < gameGrid.Rows; i++)

{

for (int j = 0; j < gameGrid.Columns; j++)

{

GameCell cell = gameGrid.getCell(i, j);

this.Controls.Add(cell.PictureBox);

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

collisionHealthManaging();

if (Keyboard.IsKeyPressed(Key.LeftArrow))

{

player.move(GameDirection.Left);

}

else if (Keyboard.IsKeyPressed(Key.RightArrow))

{

player.move(GameDirection.Right);

}

else if (Keyboard.IsKeyPressed(Key.UpArrow))

{

player.move(GameDirection.Up);

}

else if (Keyboard.IsKeyPressed(Key.DownArrow))

{

player.move(GameDirection.Down);

}

if (Keyboard.IsKeyPressed(Key.Space))

{

generatePlayerBullet();

}

manageHealth();

//Bullet Detection

foreach (Enemey item in enemyCars)

{

if (item.bulletDetection(player))

{

if (progressBar2.Value == 0)

{

goBackMenu();

}

else

{

progressBar2.Value -= 15;

}

}

}

foreach (Enemey item in enemyCars)

{

if (item.collisionDetection(player))

{

progressBar1.Value -= 25;

manageHealth();

}

}

//Firing

foreach (GameFire item in fires)

{

item.MovePlayerFire(grid);

if (!item.GetFlag())

{

fires.Remove(item);

item.CurrentCell.PictureBox.Image = My\_Car\_Game\_GUI.Properties.Resources.transparent;

break;

}

}

//Enemy

foreach (GameFire i in enemyFires)

{

i.moveEnemyFire(grid);

if (!i.GetFlag())

{

enemyFires.Remove(i);

i.CurrentCell.PictureBox.Image = Properties.Resources.transparent;

break;

}

}

generateHorizontalEnemyBullet();

playerFireCounter++;

horizontalEnemyFireCounter++;

}

private void generatePlayerBullet()

{

if (playerFireCounter>=4)

{

playerFireCell = new GameCell(player.CurrentCell.X, player.CurrentCell.Y,grid);

playerFire = new GameFire(playerFireCell, My\_Car\_Game\_GUI.Properties.Resources.arrowUp,GameObjectType.Fire);

fires.Add(playerFire);

playerFireCounter = 0;

}

}

private void generateHorizontalEnemyBullet()

{

if (horizontalEnemyFireCounter >= 4)

{

if (Horizontal != null && enemyCars.Contains(Horizontal))

{

horizontalEnemy1FireCell = new GameCell(Horizontal.CurrentCell.X, Horizontal.CurrentCell.Y, grid);

horizontalEnemy1Fire = new GameFire(horizontalEnemy1FireCell, Properties.Resources.arrowDown,GameObjectType.EnemyFire);

enemyFires.Add(horizontalEnemy1Fire);

}

horizontalEnemyFireCounter = 0;

}

}

private void label2\_Click(object sender, EventArgs e)

{

}

private void label1\_Click(object sender, EventArgs e)

{

}

private void timer2\_Tick(object sender, EventArgs e)

{

timer2.Stop();

goBackForm backForm = new goBackForm();

backForm.Show();

}

private void gameOverLBL\_Click(object sender, EventArgs e)

{

}

void collisionHealthManaging()

{

if (!player.getFlag() && progressBar1.Value >= 0)

{

progressBar1.Value -= 25;

player.setPlayerFlag(true);

}

if (!player.getFlag() && progressBar1.Value >= 0)

{

progressBar1.Value -= 25;

player.setPlayerFlag(true);

}

if (!player.getFlag() && progressBar1.Value >= 0)

{

progressBar1.Value -= 25;

player.setPlayerFlag(true);

}

if (!player.getFlag() && progressBar1.Value >= 0)

{

progressBar1.Value -= 25;

player.setPlayerFlag(true);

}

}

void manageHealth()

{

if (progressBar1.Value == 0 && playerLivesCount == 4)

{

pbHeart3.Hide();

progressBar1.Value = 100;

playerLivesCount--;

}

if (progressBar1.Value == 0 && playerLivesCount == 3)

{

pbHeart2.Hide();

playerLivesCount--;

progressBar1.Value = 100;

}

if (progressBar1.Value == 0 && playerLivesCount == 2)

{

pbHeart1.Hide();

playerLivesCount--;

progressBar1.Value = 100;

}

if (progressBar1.Value == 0 && playerLivesCount == 1)

{

pbHeart1.Hide();

playerLivesCount--;

}

if ( playerLivesCount <= 0)

{

goBackMenu();

}

}

void goBackMenu()

{

timer1.Stop();

gameOverLBL.Visible = true;

timer2.Start();

}

private void timer3\_Tick(object sender, EventArgs e)

{

foreach (Enemey g in enemyCars)

{

g.move();

}

}

}

}

Balloon Game Form CS file:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace My\_Car\_Game\_GUI

{

public partial class Player : Form

{

public Player()

{

if (Program.arow\_Game\_Start==true)

{

InitializeComponent();

gameOverLBL.Visible = false;

}

}

int score = -5;

void collision()

{

if (Arrow.Bounds.IntersectsWith(b1.Bounds))

{

b1.Top = 400;

score += 5;

label1.Text = "Score: " + score;

Arrow.Image = My\_Car\_Game\_GUI.Properties.Resources.Explosion;

}

if (Arrow.Bounds.IntersectsWith(b2.Bounds))

{

b2.Top = 400;

score += 5;

label1.Text = "Score: " + score;

Arrow.Image = My\_Car\_Game\_GUI.Properties.Resources.Explosion;

}

if (Arrow.Bounds.IntersectsWith(b3.Bounds))

{

b3.Top = 400;

score += 5;

label1.Text = "Score: " + score;

Arrow.Image = My\_Car\_Game\_GUI.Properties.Resources.Explosion;

}

}

void baloon()

{

Random random = new Random();

int x, y, z;

if (b1.Top<20)

{

b1.Top = 300;

x = random.Next(200,600);

b1.Location = new Point(x, 500);

}

if (b2.Top < 20)

{

b2.Top = 300;

y = random.Next(200, 600);

b2.Location = new Point(y, 500);

}

if (b3.Top < 20)

{

b3.Top = 300;

z = random.Next(200, 600);

b3.Location = new Point(z, 500);

}

else

{

b1.Top -= 5;

b2.Top -= 8;

b3.Top -= 10;

}

}

private void Player\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Space)

{

Arrow.Left = pictureBox1.Left;

pictureBox1.Image = My\_Car\_Game\_GUI.Properties.Resources.shoot;

}

}

void shoot()

{

Arrow.Left += 100;

if (Arrow.Left>600)

{

pictureBox1.Image = My\_Car\_Game\_GUI.Properties.Resources.idle;

Arrow.Image = My\_Car\_Game\_GUI.Properties.Resources.arrow;

Arrow.Top = pictureBox1.Top + 35;

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

collision();

shoot();

baloon();

if (score == 35)

{

timer1.Stop();

gameOverLBL.Visible = true;

timer2.Start();

}

}

private void timer2\_Tick(object sender, EventArgs e)

{

timer2.Stop();

goBackForm backForm = new goBackForm();

backForm.Show();

}

}

}

Whirly Ghost Game Form CS file:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace My\_Car\_Game\_GUI

{

public partial class Top\_Down : Form

{

public Top\_Down()

{

if (Program.top\_Down\_Start==true)

{

InitializeComponent();

gameOverLBL.Visible = false;

}

}

void movePlayer()

{

if (right==true)

{

if (player.Left<200)

{

player.Left += 5;

}

}

if (left==true)

{

if (player.Left>20)

{

player.Left -= 5;

}

}

}

bool left,right;

Random Random = new Random();

int score = 0;

void hurdles()

{

foreach (Control item in this.Controls)

{

if (item is PictureBox && item.Tag == "Block")

{

if (item.Top>450)

{

item.Top = 25;

item.Width = Random.Next(50, 200);

score++;

scoreLBL.Text = "Score: " + score;

}

if (player.Bounds.IntersectsWith(item.Bounds))

{

timer1.Stop();

goBackForm backForm = new goBackForm();

backForm.Show();

}

item.Top += 5;

}

}

}

private void Top\_Down\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode==Keys.Right)

{

right = true;

}

if (e.KeyCode == Keys.Left)

{

left = true;

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

movePlayer();

hurdles();

}

private void Top\_Down\_Load(object sender, EventArgs e)

{

timer1.Enabled = true;

timer1.Interval = 20;

}

private void timer2\_Tick(object sender, EventArgs e)

{

timer2.Stop();

goBackForm backForm = new goBackForm();

backForm.Show();

}

private void Top\_Down\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Right)

{

right = false;

}

if (e.KeyCode == Keys.Left)

{

left = false;

}

}

}

}

Reach the Castle Form CS file:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Reflection.Emit;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace My\_Car\_Game\_GUI

{

public partial class Watermelon : Form

{

bool jump;

int gravity;

int score;

int speed;

public Watermelon()

{

if (Program.water\_Melon\_Start == true)

{

InitializeComponent();

gameOverLBL.Visible = false;

castle.Hide();

}

}

void moveBases()

{

foreach (Control item in this.Controls)

{

if (item is PictureBox && item.Tag == "Base")

{

if (item.Left<-200)

{

item.Left = 900;

}

else if (player.Bounds.IntersectsWith(item.Bounds))

{

player.Top = item.Top - player.Height;

}

else if (player.Bounds.IntersectsWith(finalBase.Bounds))

{

player.Size = new Size(40, 50);

player.Top = finalBase.Top - player.Height;

speed = 0;

}

else

{

item.Left -= speed;

}

}

}

}

void jumpControl()

{

player.Top += speed;

if (jump&&gravity<0)

{

jump = false;

}

else if (jump)

{

player.Top = speed;

gravity -= 20;

}

else

{

speed = 5;

gravity = 0;

}

}

private void Watermelon\_KeyDown(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Left)

{

player.Left -= 10;

}

if (e.KeyCode == Keys.Right)

{

player.Image = My\_Car\_Game\_GUI.Properties.Resources.Walk\_\_8\_;

player.Left += 10;

}

if (e.KeyCode == Keys.Space && !jump)

{

player.Image = My\_Car\_Game\_GUI.Properties.Resources.Idle\_\_2\_;

jump = true;

}

}

void coinControll()

{

foreach (Control x in this.Controls)

{

if (x is PictureBox && x.Tag == "coin")

{

coin1.Top = base1.Width;

Random random = new Random();

int a;

if (x.Left <-900)

{

x.Left -= speed;

a = random.Next(1000);

x.Location = new Point(a, 200);

}

else if (player.Bounds.IntersectsWith(x.Bounds))

{

a = random.Next(1000);

x.Location = new Point(a, 100);

score+=5;

scoreLBL.Text = "Score: " + score;

}

else

{

x.Left -= speed;

}

}

}

}

void gameOver()

{

if (player.Bounds.IntersectsWith(ground.Bounds))

{

player.Image = My\_Car\_Game\_GUI.Properties.Resources.Dead\_\_8\_;

scoreLBL.Text = "score: " + score;

timer1.Stop();

gameOverLBL.Visible = true;

timer2.Start();

}

else if (score>50)

{

castle.Show();

if (player.Bounds.IntersectsWith(castle.Bounds))

{

player.Hide();

timer1.Stop();

gameOverLBL.Visible = true;

timer2.Start();

}

}

}

private void timer1\_Tick(object sender, EventArgs e)

{

jumpControl();

moveBases();

coinControll();

gameOver();

}

private void timer2\_Tick(object sender, EventArgs e)

{

timer2.Stop();

goBackForm backForm = new goBackForm();

backForm.Show();

}

private void Watermelon\_Load(object sender, EventArgs e)

{

}

}

}

Main Form CS file:

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace My\_Car\_Game\_GUI

{

public partial class Main\_Menu : Form

{

public Main\_Menu()

{

InitializeComponent();

}

private void btnCarGame\_Click(object sender, EventArgs e)

{

this.Hide();

Program.car\_Game\_Start = true;

Car\_Game\_Maze car\_Game\_Maze = new Car\_Game\_Maze();

car\_Game\_Maze.Show();

}

private void btnExit\_Click(object sender, EventArgs e)

{

Application.Exit();

}

private void btnArrow\_Game\_Click(object sender, EventArgs e)

{

this.Hide();

Program.arow\_Game\_Start = true;

Player player = new Player();

player.Show();

}

private void btnTopDown\_Game\_Click(object sender, EventArgs e)

{

this.Hide();

Program.top\_Down\_Start = true;

Top\_Down top\_Down = new Top\_Down();

top\_Down.Show();

}

private void btnCastle\_Game\_Click(object sender, EventArgs e)

{

this.Hide();

Program.water\_Melon\_Start = true;

Watermelon watermelon = new Watermelon();

watermelon.Show();

}

}

}

Go Back Form CS file:

using System;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace My\_Car\_Game\_GUI

{

public partial class goBackForm : Form

{

public goBackForm()

{

InitializeComponent();

}

public static Form currentForm;

private void button1\_Click(object sender, EventArgs e)

{

backToMenu();

}

void backToMenu()

{

this.Close();

ActiveForm.Hide();

Main\_Menu main\_Menu = new Main\_Menu();

main\_Menu.Show();

}

private void goBackForm\_Load(object sender, EventArgs e)

{

}

}

}