**Lost World Hunt**



Session: 2022 – 2026

**Submitted by:**

Name: Hamza Waseem

Registration No: 2022-CS-54

**Supervised by:**

Mr. Laeeq uz Zaman Khan Niazi

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[**Story Writing of Game** 3](#_Toc141600794)

[**Short Description** 3](#_Toc141600795)

[**Game Characters Description** 3](#_Toc141600796)

[ **Player** 3](#_Toc141600797)

[o **Ryan the Brave** 3](#_Toc141600798)

[ **Enemies** 4](#_Toc141600799)

[o **Dimorphodon** 4](#_Toc141600800)

[o **CyberDino** 4](#_Toc141600801)

[o **T-REX** 4](#_Toc141600802)

[o **Velociraptor** 4](#_Toc141600803)

[ **Rules & Interactions** 4](#_Toc141600806)

[ **Goal of the Game** 4](#_Toc141600807)

[**CRC Diagram** 5](#_Toc141600808)

[**OOP Concepts** 5](#_Toc141600809)

[**Design Pattern Implementation** 6](#_Toc141600810)

[**Class Details** 6](#_Toc141600811)

[ **GameObject** 6](#_Toc141600812)

[ **Ghost** 7](#_Toc141600813)

[ **Bullet** 7](#_Toc141600814)

[**Wireframes** 7](#_Toc141600815)

[**Code** 8](#_Toc141600816)

[ **BL Classes** 8](#_Toc141600817)

[ **DL Classes** 16](#_Toc141600818)

[ **UI Classes** 16](#_Toc141600819)

[**Conclusion** 29](#_Toc141600820)

# **Back Story of Game**

One fateful day, while on an expedition deep in a dense and mysterious jungle, Ryan stumbles upon a hidden cave adorned with ancient hieroglyphics. Curiosity overtakes him, and against his better judgment, he steps inside the cave. Unbeknownst to him, the cave is no ordinary one; it acts as a portal to the distant past, transporting him to the Jurassic era, where dinosaurs once roamed the Earth.

As Ryan awakens in this prehistoric world, he realizes that he's not alone. The portal has brought forth more than just him, and now he finds himself trapped in an unfamiliar land teeming with colossal creatures from a bygone era. Determined to find a way back home, Ryan must survive the perils of the Jurassic wilderness and overcome relentless challenges.

With only his wits, survival skills, and trusty hunting gear at his disposal, Ryan sets out on a perilous quest. Along the way, he encounters various dinosaur species, each more menacing than the last. The hunter becomes the hunted as he battles ferocious carnivores, evades cunning predators, and navigates treacherous landscapes.

Through the unforgiving wilderness, epic encounters, and unforeseen challenges, the hunter learns valuable lessons about respect for nature, the balance of life, and the consequences of tampering with time. Armed with determination and newfound wisdom, Ryan battles on, hoping that one day, he will find a way to return to his world, while leaving behind a legacy that will forever alter the course of history.

**Short Description**

Is a 2-D game made through GUI with the help of C# using windows forms. The Game describes how a player (ridiculously named as Ryan) gets lost into a prehistoric world while he was on the journey for hunting. The objective of the game is to get the player out of that prehistoric world while defeating the dinosaurs and other mysterious creatures comes in his way. There are collectibles that the player can collect which increases the score. Also there are three glowing balls in the map, which increase the player score by 50. If the player (ryan) collects all three glowing balls he will have a very high score but it difficult to collect all of those considering that there are many dinosaurs in his was. If a player Comes out of the maze then he will be declared as Victorious.

**Game Characters Description**

* **Player**

There is one player in the Game.

* **Ryan – the brave:**

Ryan is the main Player of the game who is a hunter by profession and has the capability of killing all the enemies that come in his way.

* **Enemies**

There are 7 enemies in the game.

* **Dimorphodon:**

Dimorhodon is a flying dino who is a great threat to the hero player. Tackling him is a very task which makes the game more difficult.

* **Cyber-Dino:**

Cyber- Dino is a new kind of specie of dinosaurs. They have great agility that has never seen before.

* **T-REX**

T-REX is also a terrifying dino. Which is an enemy in the game. Only a true warrior can escape from his powerful Jaws.

* **VelociRaptor**

Velociraptor is a small but very fast dino. If the hunter kills it then he can escape from that mysterious prehistoric world.

* **Insects**

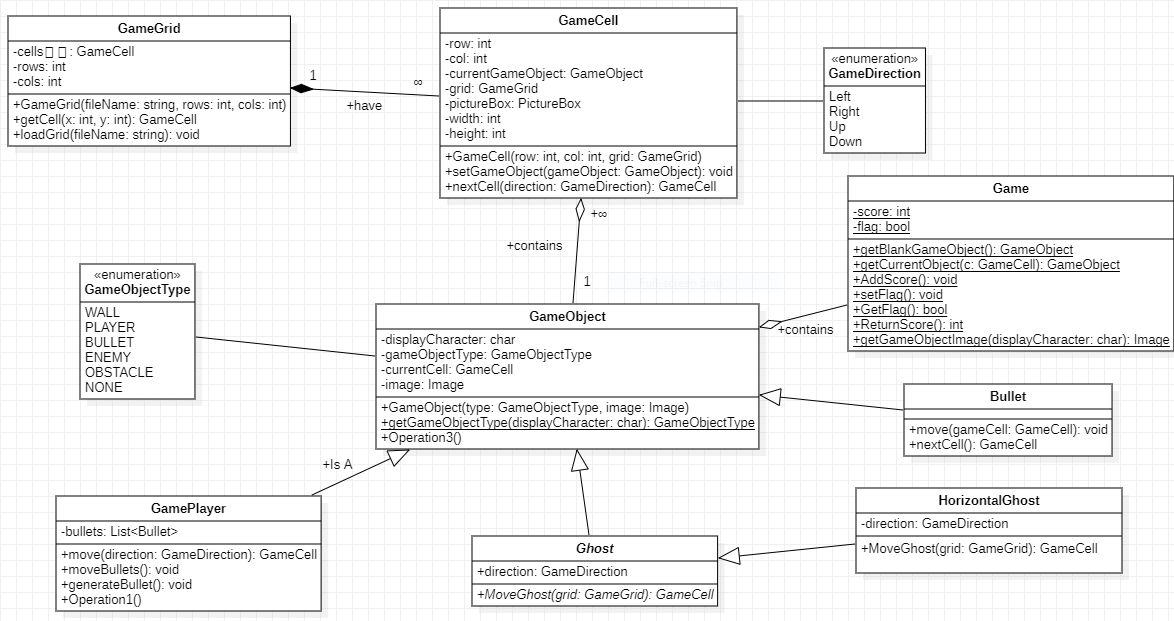
**Rules & Interactions**

Ryan the brave can gain score if he collect the pallets. He will die if he comes near a dino or insects. Avoid the cutter walls. Collect the pallets. Avoid the dinosaurs.

**Goal of the Game**

Kill all the dinos and insects and come out that maze to be victorious.

# **CRC Diagram**



<enumeration>

GameObjectType

WALL,

PLAYER

BULLET,

ENEMY,

ENDINGSTAR

GLOWY

# **OOP Concepts**

There is a parent class named **GameObject** and **Ghost** classes which includes the displayCharacter, gameObjectType, currentCell and image attributes that is common to the **Bullet** class. Inheritance is achieved by using this approach. And it is easy to by using same attributes from the parent class and the code repetition is minimized by using this approach. It is way better approach to use rather than using procedural programming.

The classes are associated with each other either of **composition** or aggregation. For example, the BL and DL classes of **BulletDL** are linked with each other by composition. It means that these classes exists only when the Bullet class is used in the program. Besides these, the Ghost class linked with Passenger class with **aggregation**. It means that the both classes existence doesn’t depends on each other. There is a polymorphism in the code to store the data in the list depending on the function call. By using this, we can easily store the parent and child class data on the same time. The **abstraction** is achieved by **Enemy** class.

**Polymorphism** is used to show different behavior when the child class want to store something the in the lists. The polymorphism automatically execute the child classes functions to write in the lists.

The use of OOP in programming increases the **readability and efficiency** of code. The repetition of code is much lesser in this programming approach. The variables, attributes and operations are arranged in very better which creates a positive impact when we are dealing with large code. It is much easier to find errors by using OOP concepts.

# **Design Pattern Implementation**

The code of the project is arranged in three different folders named as BL, DL and UI. The BL folder contains the classes in which we write the simple attributes and the related functions of the classes. The main classes are coded in this section such as Bullet, Game, GameCell, GameDirection, GameGrid, GameObject, GameObjectType, GameRyanPlayer, Ghost and Horizontal\_insect and verticle\_insect.

­­The DL folder contains classes in which we store the lists and the functions which is related to the file handling and storing or retrieving data in lists such as BulletDL.

The UI folder contains classes in which we write code to show something on the console screen and take input from the users. In this folder I have added the forms that are needed in the game.

# **Class Details**

The major classes are GameObject, Ghost and Bullet. The GameObject and Ghost is the parent class in this scenario.

# **GameObject**

This class contains all the data about the GameObject like its displayCharacter, gameObjectType, currentCell and image.

# **Enemy**

This class is responsible for handling the enemies information like its direction. Dinosaurs and the insects are enemies.

# **Bullet**

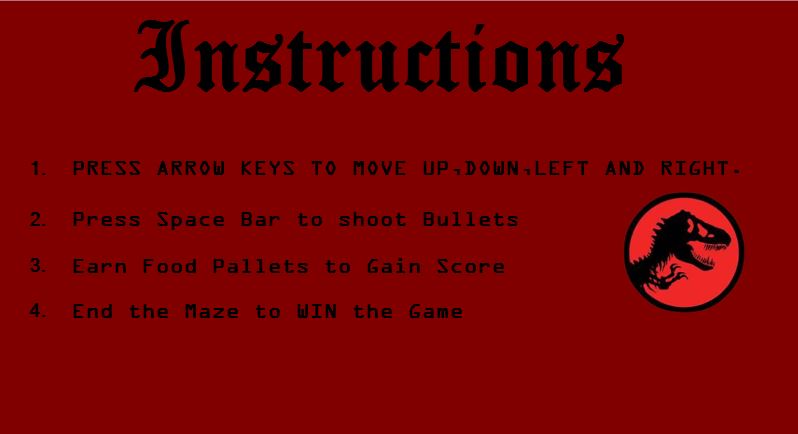
This class contains the information about the bullet and the operation to move the bullet in the game grid.

# **Wireframes**



**Figure**: Welcome Menu

# 

****

# **Figure**: Instructions Menu

# 

# 

# **Figure**: The Actual Game

# **Code**

* **BL Classes**

**Bullet.cs**

class Bullet : GameObject

{

public Bullet(Image img, GameCell cell) : base(GameObjectType.BULLET, img)

{

this.Image = img;

this.CurrentCell = cell;

}

public void move(GameCell gameCell)

{

if (this.CurrentCell != null)

{

this.CurrentCell.setGameObject(Game.getBlankGameObject());

}

CurrentCell = gameCell;

}

public GameCell nextCell()

{

return this.CurrentCell.nextCell(GameDirection.Right);

}

}

**Game.cs**

public class Game

{

static int score = 0;

static bool flag = true;

public static GameObject getBlankGameObject()

{

GameObject blankGameObject = new GameObject(GameObjectType.NONE, DinoHunter.Properties.Resources.simplebox);

return blankGameObject;

}

public static GameObject getCurrentObject(GameCell c)

{

GameObject Object = new GameObject(c.CurrentGameObject.GameObjectType, c.CurrentGameObject.Image);

return Object;

}

public static void AddScore()

{

score++;

}

public static void DecreaseScore()

{

score--;

}

public static void SetFlag()

{

flag = false;

}

public static bool GetFlag()

{

return flag;

}

public static int ReturnScore()

{

return score;

}

public static Image getGameObjectImage(char displayCharacter)

{

Image img = DinoHunter.Properties.Resources.simplebox;

if (displayCharacter == '|' || displayCharacter == '%')

{

img = DinoHunter.Properties.Resources.CroppedBoundary;

}

if (displayCharacter == '#')

{

img = DinoHunter.Properties.Resources.CroppedBoundary;

}

if (displayCharacter == '.')

{

img = DinoHunter.Properties.Resources.Burger;

}

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

{

img = DinoHunter.Properties.Resources.NewHunter;

}

if (displayCharacter == ' ')

{

img = DinoHunter.Properties.Resources.simplebox;

}

if(displayCharacter == 's' || displayCharacter == 'S')

{

img = DinoHunter.Properties.Resources.StarRemoved;

}

if (displayCharacter == 'B' || displayCharacter == 'b')

{

img = DinoHunter.Properties.Resources.Glowy;

}

return img;

}

}**GameCell.cs**

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+1 != GameObjectType.WALL)

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; }

}**GameDirection.cs**

public enum GameDirection

{

Left,

Right,

Up,

Down

}

**GameGrid.cs**

public class GameGrid

{

GameCell[,] cells;

int rows;

int cols;

public GameGrid(String fileName, int rows, int cols)

{

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.cs**

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;

}

if (displayCharacter == 'B')

{

return GameObjectType.GOLD;

}

if (displayCharacter == 'S')

{

return GameObjectType.ENDSTAR;

}

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.cs**

public enum GameObjectType

{

WALL,

PLAYER,

ENEMY,

REWARD,

ENDSTAR,

GOLD,

NONE

}**GamePlayer.cs**

class GameRyanPlayer : GameObject

{

public GameRyanPlayer(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);

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

{

Game.AddScore();

}

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(Game.getBlankGameObject());

}

return nextCell;

}

} }

**Ghost.cs**

abstract class Enemy : GameObject

{

public GameDirection direction;

public Enemy(char DisplayCharacter, GameObjectType type) : base(type, DisplayCharacter)

{

this.DisplayCharacter = DisplayCharacter;

this.GameObjectType = type;

}

public Enemy(Image img, GameObjectType type) : base(type, img)

{

this.GameObjectType = type;

this.Image = img;

}

public abstract GameCell MoveGhost(GameGrid grid);

}**HorizontalGhost.cs**

class Horizontal\_insect :Enemy

{

GameCell previous;

public Horizontal\_insect(char DisplayCharacter, GameCell cell, GameObjectType type, GameDirection direction) : base(DisplayCharacter, type)

{

this.DisplayCharacter = DisplayCharacter;

this.CurrentCell = cell;

this.direction = direction;

this.GameObjectType = type;

}

public Horizontal\_insect(Image img, GameCell cell, GameObjectType type, GameDirection direction) : base(img, type)

{

this.Image = img;

this.CurrentCell = cell;

this.direction = direction;

this.GameObjectType = type;

}

public override GameCell MoveGhost(GameGrid grid)

{

GameCell currentCell = this.CurrentCell;

if (direction == GameDirection.Left)

{

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

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

{

Game.SetFlag();

}

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

{

if (nextCell != null)

{

currentCell.setGameObject(nextCell.CurrentGameObject);

CurrentCell = nextCell;

return nextCell;

}

}

else if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

direction = GameDirection.Right;

}

}

if (direction == GameDirection.Right)

{

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

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

{

Game.SetFlag();

}

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

{

if (nextCell != null)

{

currentCell.setGameObject(Game.getCurrentObject(nextCell));

CurrentCell = nextCell;

return nextCell;

}

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

{

Game.SetFlag();

}

}

else if (nextCell.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

direction = GameDirection.Left;

}

}

return null;

}

class Vertical\_insect :Enemy

{

public Vertical\_insect(char DisplayCharacter, GameCell cell, GameObjectType type, GameDirection direction) : base(DisplayCharacter, type)

{

this.DisplayCharacter = DisplayCharacter;

this.CurrentCell = cell;

this.direction = direction;

this.GameObjectType = type;

}

public Vertical\_insect(Image img, GameCell cell, GameObjectType type, GameDirection direction) : base(img, type)

{

this.Image = img;

this.CurrentCell = cell;

this.direction = direction;

this.GameObjectType = type;

}

public override GameCell MoveGhost(GameGrid grid)

{

GameCell currentCell = this.CurrentCell;

if (direction == GameDirection.Up)

{

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

if (next.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

Game.SetFlag();

}

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

{

if (next != null)

{

currentCell.setGameObject(Game.getCurrentObject(next));

CurrentCell = next;

return next;

}

}

else if (next.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

direction = GameDirection.Down;

}

}

if (direction == GameDirection.Down)

{

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

if (next.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

Game.SetFlag();

}

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

{

if (next != null)

{

currentCell.setGameObject(Game.getCurrentObject(next));

CurrentCell = next;

return next;

}

}

else if (next.CurrentGameObject.GameObjectType == GameObjectType.WALL)

{

direction = GameDirection.Up;

}

}

return null;

}

}

class Random\_insect : Enemy

{

Random r = new Random();

public Random\_insect(char DisplayCharacter, GameCell cell, GameObjectType type, GameDirection direction) : base(DisplayCharacter, type)

{

this.DisplayCharacter = DisplayCharacter;

this.CurrentCell = cell;

this.direction = direction;

this.GameObjectType = type;

}

public Random\_insect(Image img, GameCell cell, GameObjectType type, GameDirection direction) : base(img, type)

{

this.Image = img;

this.CurrentCell = cell;

this.direction = direction;

this.GameObjectType = type;

}

public override GameCell MoveGhost(GameGrid gameGrid)

{

int value = r.Next(4);

GameCell currentCell = this.CurrentCell;

if (value == 0)

{

GameCell next = gameGrid.getCell(CurrentCell.X - 1, CurrentCell.Y);

if (next.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

Game.SetFlag();

}

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

{

if (next != null)

{

currentCell.setGameObject(Game.getCurrentObject(next));

CurrentCell = next;

return next;

}

}

}

else if (value == 1)

{

GameCell next = gameGrid.getCell(CurrentCell.X + 1, CurrentCell.Y);

if (next.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

Game.SetFlag();

}

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

{

if (next != null)

{

currentCell.setGameObject(Game.getCurrentObject(next));

CurrentCell = next;

return next;

}

}

}

else if (value == 2)

{

GameCell next = gameGrid.getCell(CurrentCell.X, CurrentCell.Y - 1);

if (next.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

Game.SetFlag();

}

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

{

if (next != null)

{

currentCell.setGameObject(Game.getCurrentObject(next));

CurrentCell = next;

return next;

}

}

}

else if (value == 3)

{

GameCell next = gameGrid.getCell(CurrentCell.X, CurrentCell.Y + 1);

if (next.CurrentGameObject.GameObjectType == GameObjectType.PLAYER)

{

Game.SetFlag();

}

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

{

if (next != null)

{

currentCell.setGameObject(Game.getCurrentObject(next));

CurrentCell = next;

return next;

}

}

}

return null;

}

}

* }**DL Classes**
* **UI Classes**

**Form1.cs**

public partial class Form2 : Form

{

List<Enemy> ghost = new List<Enemy>();

GameRyanPlayer ryan;

Horizontal\_insect h;

Vertical\_insect v;

Random\_insect r;

GameGrid grid;

GameDinoPlayer dino;

GameDinoPlayer hero;

int score = 0;

public Form2()

{

InitializeComponent();

}

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

{

ArrowKeyMovement();

BulletMovement();

GameResult();

if (Keyboard.IsKeyPressed(Key.LeftArrow))

{

dino.move(GameDirection.Left);

}

if (Keyboard.IsKeyPressed(Key.RightArrow))

{

dino.move(GameDirection.Right);

}

if (Keyboard.IsKeyPressed(Key.UpArrow))

{

dino.move(GameDirection.Up);

}

if (Keyboard.IsKeyPressed(Key.DownArrow))

{

dino.move(GameDirection.Down);

}

if (Keyboard.IsKeyPressed(Key.Space))

{

AddBullet();

}

foreach (Enemy i in ghost)

{

i.MoveGhost(grid);

}

score = Game.ReturnScore();

lblScore.Text = score.ToString();

if (score >= 200)

{

GameLoop.Enabled = false;

MessageBox.Show("You Win");

}

if (!Game.GetFlag())

{

GameLoop.Enabled = false;

MessageBox.Show("You Loose");

}

}

private void Form2\_Load(object sender, EventArgs e)

{

grid = new GameGrid("D:\\OOP\\GameProjectDinoHunter\\DinoHunter\\DinoHunter\\bin\\Debug\\nMaze.txt", 23, 69);

Image pacManImage = Game.getGameObjectImage('P');

GameCell startCell = grid.getCell(4, 8);

GameCell cell1 = grid.getCell(2, 20);

GameCell cell2 = grid.getCell(4, 14);

GameCell cell3 = grid.getCell(18, 3);

dino = new GameDinoPlayer(pacManImage, startCell);

h = new Horizontal\_insect(Properties.Resources.macharr, cell1, GameObjectType.ENEMY, GameDirection.Left);

v = new Vertical\_insect(Properties.Resources.ladybug, cell2, GameObjectType.ENEMY, GameDirection.Up);

r = new Random\_insect(Properties.Resources.YellowMachar, cell3, GameObjectType.ENEMY, GameDirection.Up);

ghost.Add(h);

ghost.Add(v);

ghost.Add(r);

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);

}

}

}

static void printCell(GameCell cell)

{

Console.SetCursorPosition(cell.Y, cell.X);

Console.Write(cell.CurrentGameObject.DisplayCharacter);

}

// FOR big Player MOvement

bool right, left, up,down,space;

void GameResult()

{

foreach(Control j in this.Controls)

{

foreach(Control i in this.Controls )

{

if(j is PictureBox && j.Tag == "bullet")

{

if (i is PictureBox && i.Tag == "enemy")

{

if (j.Bounds.IntersectsWith(i.Bounds))

{

// i.Top = -100;

// i.Visible = false;

i.Enabled = false;

}

}

}

}

}

}

void AddBullet()

{

PictureBox bullet = new PictureBox();

bullet.SizeMode = PictureBoxSizeMode.AutoSize;

bullet.Image = Properties.Resources.firey;

bullet.BackColor = System.Drawing.Color.Transparent;

bullet.Tag = "bullet";

bullet.Left = Player.Left + 48;

// bullet.Left = Player.Left ;

bullet.Top = Player.Top - 30;

// bullet.Top = Player.Top ;

this.Controls.Add(bullet);

bullet.BringToFront();

}

void BulletMovement()

{

foreach (Control x in this.Controls)

{

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

{

x.Top -= 10;

if(x.Top<100)

{

this.Controls.Remove(x);

}

}

}

}

private void Form2\_KeyPress(object sender, KeyPressEventArgs e)

{

}

void ArrowKeyMovement()

{

if(right == true)

{

if(Player.Left<425)

{

Player.Left += 30;

}

}

if(left== true)

{

if(Player.Left>10)

{

Player.Left -= 20;

}

}

/\* if(up == true)

{

}\*/

}

private void Form2\_KeyDown(object sender, KeyEventArgs e)

{

if(e.KeyCode == Keys.Right)

{

right = true;

}

if (e.KeyCode == Keys.Down)

{

down = true;

}

if (e.KeyCode == Keys.Up)

{

up = true;

}

if (e.KeyCode == Keys.Left)

{

left = true;

}

if(e.KeyCode == Keys.Space)

{

space = true;

AddBullet();

}

}

private void Form2\_KeyUp(object sender, KeyEventArgs e)

{

if (e.KeyCode == Keys.Right)

{

right = false;

}

if(e.KeyCode == Keys.Down)

{

down = false;

}

if(e.KeyCode == Keys.Up)

{

up = false;

}

if (e.KeyCode == Keys.Left)

{

left = false;

}

if (e.KeyCode == Keys.Space)

{

space = false;

}

}

}

# **Conclusion**

The game is a unique and friendly way to learn coding concepts. The use of GUI made the whole experience even better. It is capable of adding maze information using file handling. The concepts of OOPs made the whole journey really smooth.