**Space Fighter Game**



Session: 2022 – 2026

**Submitted by:**

Muhammad Tayyab 2022-CS-133

**Supervised by:**

Prof. Dr. Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

Table of Contents

[Short Description and Story of the Space Fighter: 2](#_Toc141356534)

[Game Characters Description 3](#_Toc141356535)

[Player: 3](#_Toc141356536)

[Enemy: 3](#_Toc141356537)

[Game Objects Description 3](#_Toc141356538)

[Walls: 3](#_Toc141356539)

[Enemy: 3](#_Toc141356540)

[Fire: 3](#_Toc141356541)

[Enemy Fire: 3](#_Toc141356542)

[Player: 3](#_Toc141356543)

[Shooting System: 3](#_Toc141356544)

[Rules & Interactions: 4](#_Toc141356545)

[Goal of the Game: 4](#_Toc141356546)

[Wire Frames: 4](#_Toc141356547)

[Object Oriented Programming: 6](#_Toc141356548)

[Class Diagram: 7](#_Toc141356549)

[Design Pattern Implementation: 8](#_Toc141356550)

[Class Details: 9](#_Toc141356551)

[Game code: 10](#_Toc141356552)

[Conclusion: 23](#_Toc141356553)

# Short Description and Story of the Space Fighter:

* "Space Fighter" is an exhilarating and action-packed arcade-style game that takes players on an intergalactic adventure through the vast expanse of space.
* As a skilled space fighter pilot, players will engage in thrilling battles against hordes of menacing alien enemies.
* In this game there are 3 enemies 2 horizontal, one is random.
* After destroying all 3 enemies in the space the mission of hero completed. Bringing the peaceful environment after finishing every enemy.
* There are 3 lives of player each life has health bar of 100 which decrease when bullet of enemy collide with the player.
* Also,there are health bars of 100 of all enemy.
* The task is to finish all the enemies to win the game and safe the space.

# Game Characters Description

### Player:

It has a player with a spaceship. There is only one player in the game that can move right, left, up and down. It has shooting mechanism to finish all the enemies.

### Enemy:

There are 3 enemy spaceships of three types in the game:

* There is an enemy spaceship that moves horizontally and shooting continuously.
* There is an enemy spaceship that moves vertically and shooting continuously.
* There is a Random enemy also that has also a shooting mechanism.

# Game Objects Description

These are the following objects in my game:

### Walls:

The barriers or boundaries in the game through which either hero spaceship or enemy spaceship cannot cross.

### Enemy:

This game object is related to the enemies either horizontally, random and vertical.

### Fire:

It is related with the bullets of player.

### Enemy Fire:

It is related with the bullets of enemy.

### Player:

It is the main character of the game the one which is trying to save the space.

## Shooting System:

* The hero ship has a blue laser for shooting that kills the enemies.
* The Horizontal enemy, random enemy and vertical enemy has also a shooting mechanism.

## Rules & Interactions:

The responsibility of the user playing the game is to finish all the enemies with the given life span and health to win the game.

## Goal of the Game:

The objective of the game is to survive wave after wave of enemy attacks while strategically eliminating enemy ships and bosses.

# Wire Frames:



****

****

****

# Object Oriented Programming:

This game ensure all the OOP concepts to be implemented:

* **Polymorphism:**
* I also implemented polymorphism concepts in my project like moving function that moves the game objects.
* **Inheritance:**
* Inheritance, a fundamental concept of Object-Oriented Programming (OOP), plays a significant role in this project.
* The implementation utilizes inheritance to establish relationships between different classes.
* Both the Horizontal Ghosts and Random Ghosts classes inherit from a common parent class called Ghost, which, in turn, inherits from the Game Object class.
* Similarly, the Space Shooter class and Firing class also inherit directly from the Game Object class. Inheritance, a fundamental concept of Object-Oriented Programming (OOP), plays a significant role in this project. The implementation utilizes inheritance to establish relationships between different classes. Both the Horizontal Ghosts and Random Ghosts classes inherit from a common parent class called Ghost, which, in turn, inherits from the Game Object class.
* Similarly, the Space Shooter class and Firing class also inherit directly from the Game Object class
* **Encapsulation:**

Getters and setters are implemented in this game for getting and setting the values of attributes and attributes are private.

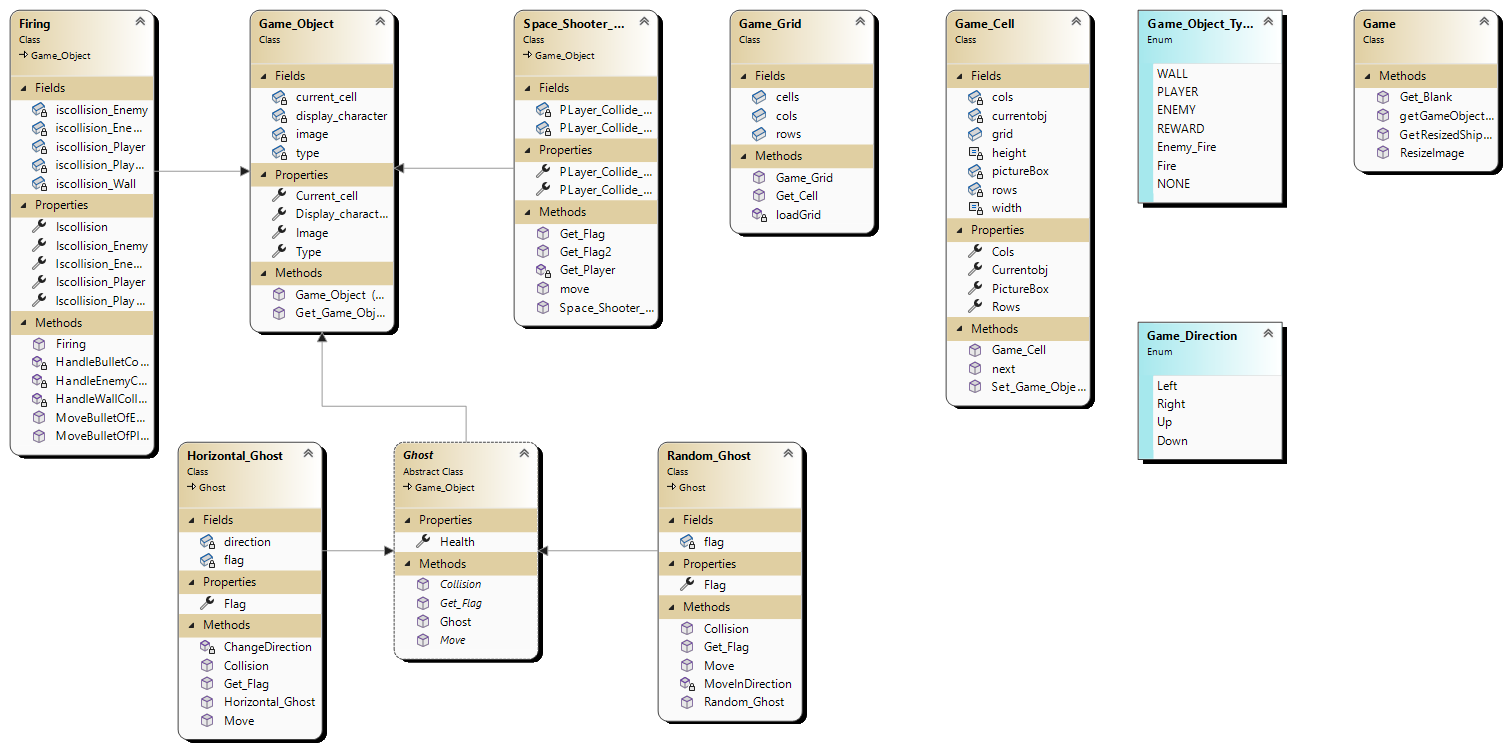
* **Abstraction:**

I have implemented many abstract classes and mechanism like abstract ghost class and its method of moving the ghost, get flag and collision detection.

* **Enumeration:**

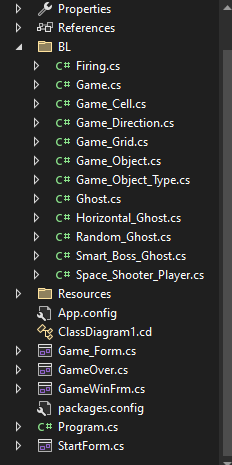
Game object and Game object type are two enumeration classes in my project.

# Class Diagram:



# Design Pattern Implementation:

The hierarchy structure of my game project is given by:

****

* **BL (Business Layer):**

This folder contains all the classes like Game Cell, Game Grid, Game Object, Player, Ghost, Horizontal, Random , Game Object Type, Game Directions, Game, Firing.

* **DL (Data Layer):**

This folder contains the lists of Ghosts, Firing, Player fire, enemy Fire.

* **UI (User Interface):**

It contains all the printing forms like printing of Map, Players, Ghosts, Bullets, Helath bar, Life span. Printing of levels also a part of this folder.

# Class Details:

* **Game Object:**

This class contains all the objects that are the parts of game like Player, Wall, all Ghosts, Fire. It is the parent class of many classes like Ghost, Player. Its has two constructors each takes two parameters.

* **Game Directions:**

This class contains the possible directions of the game it has four directions Right, Left, Up, Down.

* **Game Cell:**

This class contains the all the cells of the map through which the whole map made. It contains the object of Game Grid class showing association. Its constructor takes 3 parameters including row of cell column of cell and grid object.

* **Game Grid:**

This class contains the array of Game cell class and rows and columns of the map. It also contains the function of grid load that calls automatically in the constructor.

* **Game Object:**

This class contains different function like getting transparent image getter function, Check game over, check boss enemy and get game object image.

* **Game Object Type:**

This class contains the types of different object it has types of player, ghost, fire, enemy fire.

* **Ghost:**

This is an abstract class that inherits game object and has abstract functions like move, collision, get flag. Its constructor takes two attributes game cell and image.

* **Horizontal Ghost:**

This class contains ghost like horizontal and it inherits ghost class which is abstract. It takes image game cell, image and direction of move.

* **Random Ghost:**

This class contains ghost like random and it inherits ghost class which is abstract. It takes image game cell, image.

# Game code:

The code of my game project is given below:

**Game Grid:**

public class Game\_Grid

{

public static Game\_Cell[,] cells;

public int rows;

public int cols;

public Game\_Grid(string filename, int rows, int cols)

{

this.rows = rows;

this.cols = cols;

cells = new Game\_Cell[rows, cols];

this.loadGrid(filename);

}

public static Game\_Cell Get\_Cell(int x, int y)

{

return cells[x, y];

}

void loadGrid(string fileName)

{

StreamReader fp = new StreamReader("Maze.txt");

string record;

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

{

record = fp.ReadLine();

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

{

Game\_Cell cell = new Game\_Cell(row, col, this);

Char displayCharacter = record[col];

Game\_Object\_Type t = Game\_Object.Get\_Game\_Object\_Type(displayCharacter);

Image displayIamge = Game.getGameObjectImage(displayCharacter);

Game\_Object gameObject = new Game\_Object(t, displayIamge);

cell.Set\_Game\_Object(gameObject);

cells[row, col] = cell;

}

}

fp.Close();

}

}

**Game Cell:**

public class Game\_Cell

{

int rows;

int cols;

Game\_Object currentobj;

private PictureBox pictureBox;

public Game\_Grid grid;

const int width = 70;

const int height = 70;

public int Rows { get => rows; set => rows = value; }

public int Cols { get => cols; set => cols = value; }

public Game\_Object Currentobj { get => currentobj; set => currentobj = value; }

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

public Game\_Cell(int rows, int cols, Game\_Grid grid)

{

this.Rows = rows;

this.Cols = cols;

this.grid = grid;

PictureBox = new PictureBox();

PictureBox.Left = cols \* width;

PictureBox.Top = rows \* height;

PictureBox.BackColor = Color.Transparent;

PictureBox.Size = new Size(width, height);

PictureBox.SizeMode = PictureBoxSizeMode.Zoom;

}

public void Set\_Game\_Object(Game\_Object gameobject)

{

Currentobj = gameobject;

PictureBox.Image = gameobject.Image;

}

public Game\_Cell next(Game\_Direction direction)

{

if (direction == Game\_Direction.Left)

{

if (this.Cols > 0)

{

Game\_Cell ncell = Game\_Grid.Get\_Cell(this.rows, this.cols - 1);

if (ncell.Currentobj.Type != Game\_Object\_Type.WALL)

{

return ncell;

}

}

}

if (direction == Game\_Direction.Right)

{

if (this.Cols < grid.cols - 1)

{

Game\_Cell ncell = Game\_Grid.Get\_Cell(this.rows, this.cols + 1);

if (ncell.Currentobj.Type != Game\_Object\_Type.WALL)

{

return ncell;

}

}

}

if (direction == Game\_Direction.Up)

{

if (this.rows > 0)

{

Game\_Cell ncell = Game\_Grid.Get\_Cell(this.rows - 1, this.cols);

if (ncell.Currentobj.Type != Game\_Object\_Type.WALL)

{

return ncell;

}

}

}

if (direction == Game\_Direction.Down)

{

if (this.rows < grid.rows - 1)

{

Game\_Cell ncell = Game\_Grid.Get\_Cell(this.rows + 1, this.cols);

if (ncell.Currentobj.Type != Game\_Object\_Type.WALL)

{

return ncell;

}

}

}

return this;

}

}

**Game Object:**

public class Game\_Object

{

private char display\_character;

private Game\_Object\_Type type;

private Game\_Cell current\_cell;

private Image image;

public Game\_Object(Game\_Object\_Type type, Image image)

{

this.Type = type;

this.Image = image;

}

public Game\_Object(char display\_character, Game\_Object\_Type type)

{

this.Display\_character = display\_character;

this.Type = type;

}

public static Game\_Object\_Type Get\_Game\_Object\_Type(char displayCharacter)

{

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

{

return Game\_Object\_Type.WALL;

}

return Game\_Object\_Type.NONE;

}

public char Display\_character { get => display\_character; set => display\_character = value; }

public Game\_Object\_Type Type { get => type; set => type = value; }

public Game\_Cell Current\_cell

{

get => current\_cell; set

{

current\_cell = value;

current\_cell.Set\_Game\_Object(this);

}

}

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

}

**Game Object Type:**

namespace Space\_Shooters.BL

{

public enum Game\_Object\_Type

{

WALL,

PLAYER,

ENEMY,

REWARD,

Enemy\_Fire,

Fire,

NONE

}

}

**Game Directions Type:**

namespace Space\_Shooters.BL

{

public enum Game\_Direction

{

Left,

Right,

Up,

Down

}

}

**Firing:**

public class Firing:Game\_Object

{

bool iscollision\_Wall=false;

bool iscollision\_Player = false;

bool iscollision\_Enemy = false;

bool iscollision\_EnemyFire = false;

bool iscollision\_PlayerFire = false;

public Firing(Image image, Game\_Cell Cell,Game\_Object\_Type type) : base(type, image)

{

this.Current\_cell = Cell;

}

public bool Iscollision { get => iscollision\_Wall; set => iscollision\_Wall = value; }

public bool Iscollision\_Player { get => iscollision\_Player; set => iscollision\_Player = value; }

public bool Iscollision\_Enemy { get => iscollision\_Enemy; set => iscollision\_Enemy = value; }

public bool Iscollision\_PlayerFire { get => iscollision\_PlayerFire; set => iscollision\_PlayerFire = value; }

public bool Iscollision\_EnemyFire { get => iscollision\_EnemyFire; set => iscollision\_EnemyFire = value; }

public Game\_Cell MoveBulletOfEnemy()

{

Game\_Cell currentCell = this.Current\_cell;

Game\_Cell nextCell = currentCell.next(Game\_Direction.Down);

if (nextCell != currentCell)

{

if (nextCell != null)

{

currentCell.Set\_Game\_Object(Game.Get\_Blank());

Current\_cell = nextCell;

return nextCell;

}

}

if (nextCell == currentCell)

{

HandleBulletCollision(currentCell);

}

if (nextCell != null && nextCell.Currentobj.Type == Game\_Object\_Type.WALL)

{

HandleWallCollision();

}

return null;

}

private void HandleBulletCollision(Game\_Cell currentCell)

{

currentCell.Set\_Game\_Object(Game.Get\_Blank());

Iscollision = true;

}

private void HandleWallCollision()

{

Iscollision\_PlayerFire = true;

}

public Game\_Cell MoveBulletOfPlayer()

{

Game\_Cell currentCell = this.Current\_cell;

Game\_Cell nextCell = currentCell.next(Game\_Direction.Up);

if (nextCell != null && nextCell.Currentobj.Type == Game\_Object\_Type.ENEMY)

{

HandleEnemyCollision();

}

if (nextCell != currentCell)

{

if (nextCell != null)

{

currentCell.Set\_Game\_Object(Game.Get\_Blank());

Current\_cell = nextCell;

return nextCell;

}

}

if (nextCell == currentCell)

{

HandleBulletCollision(currentCell);

}

return null;

}

private void HandleEnemyCollision()

{

Iscollision\_Enemy = true;

}

}

**Game:**

public class Game

{

public static Image ResizeImage(Image originalImage, int newWidth, int newHeight)

{

Bitmap resizedImage = new Bitmap(newWidth, newHeight);

using (Graphics graphics = Graphics.FromImage(resizedImage))

{

graphics.DrawImage(originalImage, 0, 0, newWidth, newHeight);

}

return resizedImage;

}

public static Image GetResizedShipImage(int newWidth, int newHeight)

{

Image shipImage = My\_Game.Properties.Resources.Ship;

return ResizeImage(shipImage, newWidth, newHeight);

}

public static Image getGameObjectImage(char displayCharacter)

{

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

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

{

img = My\_Game.Properties.Resources.vertical2;

}

if (displayCharacter == '#')

{

img = My\_Game.Properties.Resources.horizontal;

}

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

{

img= My\_Game.Properties.Resources.images\_\_1\_\_\_1\_\_removebg\_preview;

}

if (displayCharacter == 'E')

{

img = My\_Game.Properties.Resources.Ghost;

}

if (displayCharacter == 'e')

{

img = My\_Game.Properties.Resources.Ghost2;

}

if (displayCharacter == 'R')

{

img = My\_Game.Properties.Resources.Enemy;

}

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

{

img = My\_Game.Properties.Resources.\_27803\_91fd7c56c0817b211ea4827f9d6b324e;

}

if (displayCharacter == '.')

{

img = My\_Game.Properties.Resources.laserRed13;

}

if (displayCharacter == ' ')

{

img = My\_Game.Properties.Resources.transparent1;

}

if (displayCharacter == '.')

{

img = My\_Game.Properties.Resources.laserRed13;

}

if (displayCharacter == '\*')

{

img = My\_Game.Properties.Resources.bullet;

}

return img;

}

public static Game\_Object Get\_Blank()

{

Game\_Object blankGameObject = new Game\_Object(Game\_Object\_Type.NONE, My\_Game.Properties.Resources.transparent1);

return blankGameObject;

}

}

**Game Ghost:**

namespace Space\_Shooters.BL

{

public abstract class Ghost : Game\_Object

{

public Ghost(Image image, Game\_Cell Cell) : base(Game\_Object\_Type.ENEMY, image)

{

}

public abstract Game\_Cell Move();

public abstract bool Get\_Flag();

public abstract bool Collision(Space\_Shooter\_Player player);

}

}

**Horizontal Ghost:**

public class Horizontal\_Ghost : Ghost

{

Game\_Direction direction;

bool flag = false;

public Horizontal\_Ghost(Image image, Game\_Cell cell, Game\_Direction d) : base(image, cell)

{

this.direction = d;

this.Current\_cell = cell;

}

public bool Flag { get => flag; set => flag = value; }

public override Game\_Cell Move()

{

Game\_Cell currentCell = this.Current\_cell;

Game\_Cell next = currentCell.next(direction);

if (next.Currentobj.Type == Game\_Object\_Type.Fire)

{

Flag = true;

}

if (currentCell == next)

{

ChangeDirection();

}

else

{

currentCell.Set\_Game\_Object(next.Currentobj);

this.Current\_cell = next;

return next;

}

return null;

}

private void ChangeDirection()

{

if (this.direction == Game\_Direction.Left)

{

this.direction = Game\_Direction.Right;

}

else if (this.direction == Game\_Direction.Right)

{

this.direction = Game\_Direction.Left;

}

}

public override bool Get\_Flag()

{

return this.Flag;

}

public override bool Collision(Space\_Shooter\_Player player)

{

return (Game\_Object\_Type.PLAYER == this.Current\_cell.Currentobj.Type);

}

}

**Random Ghost:**

public class Random\_Ghost : Ghost

{

bool flag = false;

public bool Flag { get => flag; set => flag = value; }

public Random\_Ghost(Image image, Game\_Cell cell) : base(image, cell)

{

this.Current\_cell = cell;

}

public override Game\_Cell Move()

{

Random random = new Random();

int number = random.Next(10);

if (number % 4 == 0)

return MoveInDirection(Game\_Direction.Down);

if (number % 3 == 0)

return MoveInDirection(Game\_Direction.Up);

if (number % 7 == 0)

return MoveInDirection(Game\_Direction.Right);

if (number % 9 == 0)

return MoveInDirection(Game\_Direction.Left);

return null;

}

private Game\_Cell MoveInDirection(Game\_Direction direction)

{

Game\_Cell currentCell = this.Current\_cell;

Game\_Cell nextCell = currentCell.next(direction);

if (nextCell.Currentobj.Type == Game\_Object\_Type.Fire)

Flag = true;

if (nextCell.Currentobj.Type != Game\_Object\_Type.WALL && currentCell != nextCell)

{

currentCell.Set\_Game\_Object(nextCell.Currentobj);

this.Current\_cell = nextCell;

return nextCell;

}

return null;

}

public override bool Get\_Flag()

{

return this.Flag;

}

public override bool Collision(Space\_Shooter\_Player player)

{

return (Game\_Object\_Type.PLAYER == this.Current\_cell.Currentobj.Type);

}

}

**Player:**

public class Space\_Shooter\_Player : Game\_Object

{

bool PLayer\_Collide\_Enemy\_Fire = false;

bool PLayer\_Collide\_Enemy = false;

public bool PLayer\_Collide\_Enemy\_Fire1 { get => PLayer\_Collide\_Enemy\_Fire; set => PLayer\_Collide\_Enemy\_Fire = value; }

public bool PLayer\_Collide\_Enemy1 { get => PLayer\_Collide\_Enemy; set => PLayer\_Collide\_Enemy = value; }

public Space\_Shooter\_Player(Image image, Game\_Cell startCell) : base(Game\_Object\_Type.PLAYER, image)

{

this.Current\_cell = startCell;

}

public bool Get\_Flag()

{

return PLayer\_Collide\_Enemy\_Fire1;

}

public bool Get\_Flag2()

{

return PLayer\_Collide\_Enemy1;

}

private Space\_Shooter\_Player Get\_Player()

{

return this;

}

public Game\_Cell move(Game\_Direction direction)

{

Game\_Cell currentCell = this.Current\_cell;

Game\_Cell nextCell = currentCell.next(direction);

if (Current\_cell.Currentobj.Type == Game\_Object\_Type.Enemy\_Fire)

{

PLayer\_Collide\_Enemy\_Fire1 = true;

}

if (Current\_cell.Currentobj.Type == Game\_Object\_Type.ENEMY)

{

PLayer\_Collide\_Enemy1 = true;

}

this.Current\_cell = nextCell;

if (currentCell != nextCell)

{

currentCell.Set\_Game\_Object(Game.Get\_Blank());

}

return nextCell;

}

}

# Conclusion:

In this game development project, I fully embraced the principles of Object-Oriented Programming (OOP) and harnessed its main pillars: inheritance, association, abstraction, and enumeration. Though it presented a challenging task, I dedicated myself to resolving all the issues I encountered during the procedural programming phase. By doing so, I succeeded in crafting an engaging and interactive game experience for players.

Throughout the development process, I implemented a wide range of functionalities and focused on delivering a polished and enjoyable game. Learning how to resolve the challenges I faced was an invaluable experience, and it enriched my understanding of GUI development. I discovered how to create captivating and immersive games that captivate players' interest.