**Ship War**

****

Session: 2022 – 2026

**Submitted by:**

Robass Atif 2022-CS-150

**Supervised by:**

Prof. Dr. Muhammad Awais Hassan

Department of Computer Science

**University of Engineering and Technology**

**Lahore Pakistan**

**Table of Contents**

**Link………..………………………………………………………………………………...……3**

**Introduction………………………………………………………………………………...……3**

**Class Responsibility Collaboration Card……………………………...………………3**

**Object Oriented Programming………………………………………………………...…4**

**Design Pattern Implementation..………………………………………………………...5**

**Conclusion……........……………………………………………………………..…….......……7**

**Code……........……………………………………………………………..……........……………7**

**Images……........…………………………………………………………………..…….......……7**

* **Link**

https://youtu.be/yd7pYLfmX2s

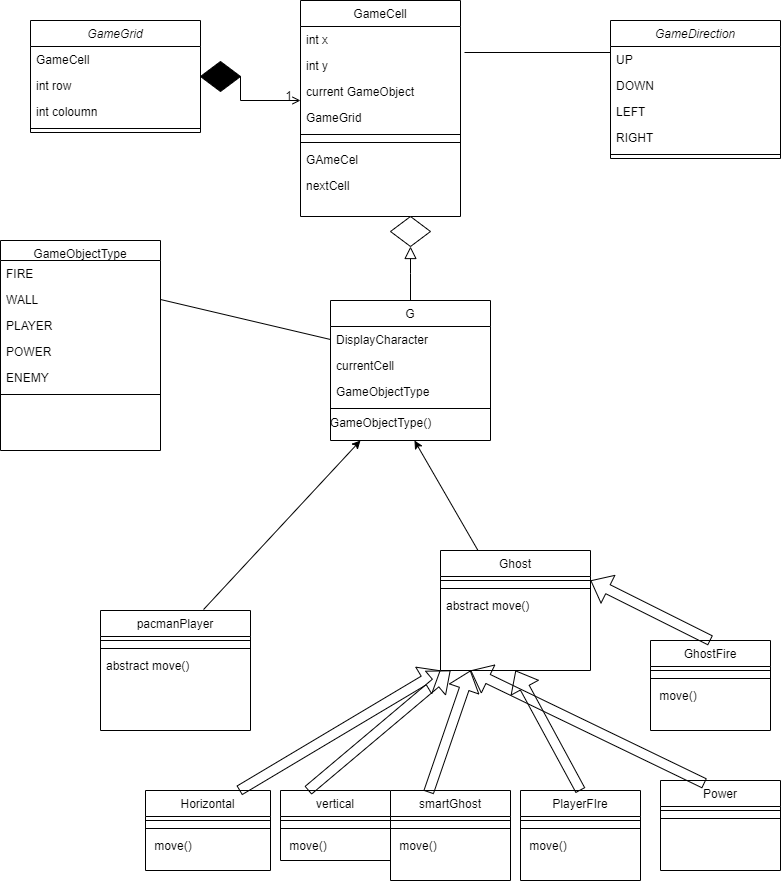
* **Introduction**
* **Overview**

Ship War is Game in which there are four main character one is player and three ghosts. Player in this game also able to fire and ghosts too. First of all there are two ghost one is horizontal and second is vertical if player kill the both enemy then both ghost disappear and the big Ghost appear which move vertical and there is also power booster which appear in some random places which increase a player life.

* **Functionality**

The intended functionality for the is there detection of fire and character and there is also detection between character and character.

* **Class Responsibility Collaboration Diagrams**

****

* **Object Oriented Programming**

Object-oriented programming (OOP) is a programming paradigm based on the concepts of the “objects”, which contains data and code called as attributes and behavior of the class respectively.

* **Association**

1. G with GameCell

**Advantage**

If I compare this with my procedural programming concepts, I can observe that there is a clear advantage of OOP.

* **Inheritance**

1. Palyer and ghost inherits from G.
2. All ghost inherits from ghost.

**Advantage**

Inheritance gives various advantages over procedural programming. It promotes code-reusability and reduces redundancy

* **Polymorphism**

1. move() function in ghost and in all in child ghost

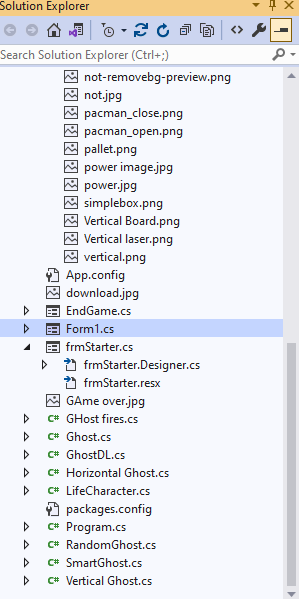
The other type of Polymorphism is Static Polymorphism. I have used this polymorphism only for the constructors.

**Advantage**

Polymorphism allowed us to extend the functionalities from the base class to use them for the child classes. Due to polymorphism, our code has become shorter because it didn’t required us to copy paste the whole code and then make changes to it.

* **Design Pattern Implementation**

The directory structure for the project is given below :-

****

* **Business Logic (BL)**

The Business Logic folder contains the main classes of the project. It conatins player,gamecell,ghost,power and all other.

* **Data Layer (DL)**

The Data Layer folder contains the static Lists and functions of the project. Only ghost DL is made for polymorphism.

* **User Interface (UI)**

This folder contains all the code for printing and input of the data. This folder deals with the interaction of user with the application. It includes the menus and other functions which are used for printing and taking inputs from the user.

* **Conclusion**

In conclusion, the Ship War is built using the object-oriented programming approach. It is interesting to play this Game.

* **Code**

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();

}

public class GameCell

{

int row;

int col;

public GameObject currentGameObject;

public 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; }

}

public class Game

{

public static int score=0;

public static int playerLife = 100;

public static int life = 3;

public static int Vghost = 10;

public static int bigGHost = 10;

public static bool flag1 = true;

public static bool powerFlag = true;

public static bool bigGhost = false;

public static List<GHost\_fires> bigGhostFires = new List<GHost\_fires>();

public static bool flag = false;

public static List<PlayerFires> fires = new List<PlayerFires>();

public static List<GHost\_fires> GHostFires = new List<GHost\_fires>();

public static GameObject getBlankGameObject(){

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

return blankGameObject;

}

public static Image getGameObjectImage(char displayCharacter)

{

Image img = PacManGUI.Properties.Resources.simplebox;

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

{

img = PacManGUI.Properties.Resources.vertical;

}

if(displayCharacter=='+')

{

img = PacManGUI.Properties.Resources.not\_removebg\_preview;

}

if(displayCharacter=='#')

{

img = PacManGUI.Properties.Resources.horizontal;

}

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

img = PacManGUI.Properties.Resources.image;

}

if(displayCharacter=='H')

{

img = PacManGUI.Properties.Resources.hullLarge\_\_2\_;

}

if(displayCharacter=='R')

{

img = PacManGUI.Properties.Resources.image\_\_2\_;

}

if(displayCharacter=='~')

{

img = PacManGUI.Properties.Resources.image\_\_3\_;

}

if(displayCharacter=='V')

{

img = PacManGUI.Properties.Resources.Vertical\_Board;

}

if(displayCharacter=='-')

{

img = PacManGUI.Properties.Resources.image\_\_5\_;

}

if (displayCharacter == '\*')

{

img = PacManGUI.Properties.Resources.laserRed01;

}

if (displayCharacter=='&')

{

img = PacManGUI.Properties.Resources.Vertical\_laser;

}

if(displayCharacter=='B')

{

img = PacManGUI.Properties.Resources.Big\_Ghost;

}

return img;

}

public static void removeFire()

{

for(int i=0;i<fires.Count;i++)

{

if(!fires[i].active)

{

fires[i].CurrentCell.setGameObject(Game.getBlankGameObject());

fires.Remove(fires[i]);

}

}

}

public static void moveFire()

{

foreach (PlayerFires x in fires)

{

x.move();

}

}

public static void moveGhostFire()

{

foreach (GHost\_fires x in GHostFires)

{

x.move();

}

}

public static void removeGhostFire()

{

for (int i = 0; i < GHostFires.Count; i++)

{

if (!GHostFires[i].active)

{

GHostFires.Remove(GHostFires[i]);

}

}

}

public static void movebigGhostFire()

{

foreach(GHost\_fires x in bigGhostFires)

{

x.move();

}

}

public static void removeBigGhostFire()

{

for (int i = 0; i < bigGhostFires.Count; i++)

{

if (!bigGhostFires[i].active)

{

bigGhostFires.Remove(bigGhostFires[i]);

}

}

}

public enum GameDirection

{

Left,

Right,

Up,

Down

}

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( 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; }

}

public enum GameObjectType

{

WALL,

PLAYER,

ENEMY,

REWARD,

FIRE,

POWER,

NONE

}

public class GamePacManPlayer : GameObject

{

public GamePacManPlayer(Image image,GameCell startCell):base (GameObjectType.PLAYER,image) {

this.CurrentCell = startCell;

}

public void move(GameDirection direction) {

GameCell currentCell = this.CurrentCell;

GameCell nextCell= currentCell.nextCell(direction);

if(nextCell.currentGameObject.GameObjectType==GameObjectType.POWER)

{

Game.life++;

Game.powerFlag = true;

}

if (nextCell.currentGameObject.GameObjectType == GameObjectType.ENEMY)

{

Game.flag1 = false;

Game.playerLife = 0;

}

this.CurrentCell = nextCell;

if (currentCell != nextCell) {

currentCell.setGameObject(Game.getBlankGameObject());

}

}

}

public class PlayerFires:GameObject

{

private bool active = true;

GameDirection d;

public PlayerFires( Image image,GameCell start,GameDirection d): base (GameObjectType.FIRE,image)

{

this.CurrentCell = start;

this.d = d;

}

public bool Active { get => active; set => active = value; }

public void move()

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(d);

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

{

Game.Vghost--;

if(Game.bigGhost)

{

Game.bigGHost--;

}

Game.score+=2;

}

if(currentCell==nextCell || nextCell.CurrentGameObject.GameObjectType==GameObjectType.ENEMY )

{

Active = false;

}

if(Active)

{

this.CurrentCell = nextCell;

}

currentCell.setGameObject(Game.getBlankGameObject());

}

public partial class Form1 : Form

{

GamePacManPlayer pacman;

Horizontal\_Ghost H;

Vertical\_Ghost V;

Vertical\_Ghost bigG;

int timer = 0;

LifeCharacter power;

Random rand = new Random();

public Form1()

{

InitializeComponent();

}

GameGrid grid;

private void Form1\_Load(object sender, EventArgs e)

{

grid = new GameGrid("maze.txt", 24, 69);

createPacman();

createHGhost();

createVGhost();

printMaze(grid);

creationOfLAbelThings();

}

private void startCellOfPacman()

{

if(!Game.flag1)

{

GameCell curr = pacman.CurrentCell;

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

pacman.CurrentCell = startCell;

curr.setGameObject(Game.getBlankGameObject());

Game.flag1 = true;

}

}

private void start()

{

this.Controls.Clear();

gameLoop.Enabled = true;

Game.bigGhost = false;

Game.bigGHost = 10;

grid = new GameGrid("maze.txt", 24, 69);

createPacman();

createHGhost();

createVGhost();

Game.playerLife = 100;

Game.life = 3;

printMaze(grid);

creationOfLAbelThings();

}

private void createPacman()

{

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

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

pacman = new GamePacManPlayer(pacManImage, startCell);

}

private void createPowerBooster()

{

int element = rand.Next(1, 4);

if (element == 1)

{

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

GameCell startCell = grid.getCell(9, 10);

power = new LifeCharacter(pacManImage, startCell);

}

if (element == 2)

{

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

GameCell startCell = grid.getCell(9, 14);

power = new LifeCharacter(pacManImage, startCell);

}

if (element == 3)

{

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

GameCell startCell = grid.getCell(14, 5);

power = new LifeCharacter(pacManImage, startCell);

}

if (element == 4)

{

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

GameCell startCell = grid.getCell(4, 40);

power = new LifeCharacter(pacManImage, startCell);

}

}

private void createHGhost()

{

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

GameCell startCell = grid.getCell(2, 63);

H = new Horizontal\_Ghost(pacManImage, startCell);

GhostDL.ghosts.Add(H);

}

private void createVGhost()

{

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

GameCell startCell = grid.getCell(5, 64);

V = new Vertical\_Ghost(pacManImage, startCell);

GhostDL.ghosts.Add(V);

}

private void createBigGHost()

{

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

GameCell startCell1 = grid.getCell(6, 62);

bigG = new Vertical\_Ghost(pacManImage, startCell1);

GhostDL.ghosts.Add(bigG);

Game.bigGhost = true;

}

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);

}

int timerOfPower = 0;

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

{

if(Keyboard.IsKeyPressed(Key.LeftArrow)) {

pacman.move(GameDirection.Left);

}

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

pacman.move(GameDirection.Right);

}

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

pacman.move(GameDirection.Up);

}

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

pacman.move(GameDirection.Down);

}

if(Keyboard.IsKeyPressed(Key.Space))

{

if (timer == 2)

{

createFire();

timer = 0;

}

timer++;

}

if(timerOfPower==150)

{

if (Game.powerFlag)

{

createPowerBooster();

Game.powerFlag = false;

}

timerOfPower = 0;

}

timerOfPower++;

score1.Text = Game.score.ToString();

if (V.active)

{

createVFire();

}

Game.removeGhostFire();

Game.moveGhostFire();

if (H.active)

{

createGhostFire(GameDirection.Down);

}

Game.moveFire();

GhostDL.move();

Game.removeFire();

health.Value = Game.playerLife;

removevghost();

if (Game.bigGhost == true)

{

if (timer2 == 3)

{

BigGHostFire();

timer2 = 0;

}

timer2++;

}

Game.movebigGhostFire();

Game.removeBigGhostFire();

startCellOfPacman();

life();

YouWin();

life1.Text = Game.life.ToString();

}

int timer2 = 0;

private void removevghost()

{

if (Game.Vghost <= 0)

{

GhostDL.ghosts.Remove(V);

GhostDL.ghosts.Remove(H);

H.active = false;

V.active = false;

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

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

Game.Vghost = 10;

createBigGHost();

}

}

private void createFire()

{

if (pacman.CurrentCell.Y + 2 < 67)

{

Image fireImage = Game.getGameObjectImage('-');

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

PlayerFires f = new PlayerFires(fireImage, startCell,GameDirection.Right);

Game.fires.Add(f);

}

}

int x = 0;

private void createVFire()

{

if (x==5)

{

Image fireImage = Game.getGameObjectImage('&');

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

GHost\_fires f = new GHost\_fires(fireImage, startCell, GameDirection.Left);

Game.GHostFires.Add(f);

x = 0;

}

x++;

}

int timerGhostFire = 0;

private void createGhostFire(GameDirection d1)

{

GameDirection d=d1;

if (timerGhostFire == 3)

{

Image fireImage = Game.getGameObjectImage('\*');

GameCell startCell = grid.getCell(H.CurrentCell.X + 2, H.CurrentCell.Y);

GHost\_fires f = new GHost\_fires(fireImage, startCell, d);

Game.GHostFires.Add(f);

timerGhostFire = 0;

}

timerGhostFire++;

}

private void BigGHostFire()

{

Image fireImage = Game.getGameObjectImage('~');

GameCell startCell = grid.getCell(bigG.CurrentCell.X , bigG.CurrentCell.Y-2);

GHost\_fires f = new GHost\_fires(fireImage, startCell,GameDirection.Left );

Game.bigGhostFires.Add(f);

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

public Label score;

Label life2;

Label health1;

TextBox score1;

TextBox life1;

ProgressBar health;

private void creationOfLAbelThings()

{

score = new Label();

score.ForeColor = Color.Yellow;

score.Left = 100;

score.Top = this.Height - 250;

score.Text = "Score : ";

score.Font = new Font(score.Font.FontFamily, 14);

score1 = new TextBox();

score1.ForeColor = Color.Yellow;

score1.Left = (score.Left+score.Width)+ 20;

score1.Top = this.Height - 250;

score1.ReadOnly = true;

score1.Text = "0";

life2 = new Label();

life2.ForeColor = Color.Yellow;

life2.Left = 100;

life2.Top = this.Height - 180;

life2.Text = "Life : ";

life2.Font = new Font(score.Font.FontFamily, 14);

life1 = new TextBox();

life1.ForeColor = Color.Yellow;

life1.Left = (life2.Left + life2.Width) + 5;

life1.Top = this.Height - 180;

life1.ReadOnly = true;

life1.Text = Game.life.ToString();

health1 = new Label();

health1.ForeColor = Color.Yellow;

health1.Left = score1.Left+150;

health1.Top = this.Height - 250;

health1.Text = "Health: ";

health1.Font = new Font(score.Font.FontFamily, 14);

health = new ProgressBar();

health.Left = health1.Left + health1.Width + 10;

health.Top = health1.Top;

health.Value = Game.playerLife;

this.Controls.Add(health);

this.Controls.Add(life1);

this.Controls.Add(life2);

this.Controls.Add(score);

this.Controls.Add(score1);

this.Controls.Add(health1);

}

public class Vertical\_Ghost:Ghost

{

public Vertical\_Ghost(Image image, GameCell startCell) : base(GameObjectType.ENEMY, image)

{

this.CurrentCell = startCell;

this.direction = GameDirection.Down;

objects = Game.getBlankGameObject();

}

public bool active = true;

public GameDirection direction;

public GameObject objects;

public override void move()

{

if (direction == GameDirection.Down)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

if (nextCell.currentGameObject.GameObjectType == GameObjectType.ENEMY)

{

direction = GameDirection.Up;

}

else

{

{

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(objects);

}

else

{

direction = GameDirection.Up;

}

}

}

}

else if (direction == GameDirection.Up)

{

GameCell currentCell = this.CurrentCell;

GameCell nextCell = currentCell.nextCell(direction);

if (nextCell.currentGameObject.GameObjectType == GameObjectType.ENEMY)

{

direction = GameDirection.Down;

}

else

{

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

{

Game.flag1 = false;

Game.life--;

}

else

{

this.CurrentCell = nextCell;

if (currentCell != nextCell)

{

currentCell.setGameObject(objects);

}

else

{

direction = GameDirection.Down;

}

}

}

}

}

* **Images**



Figure: 1

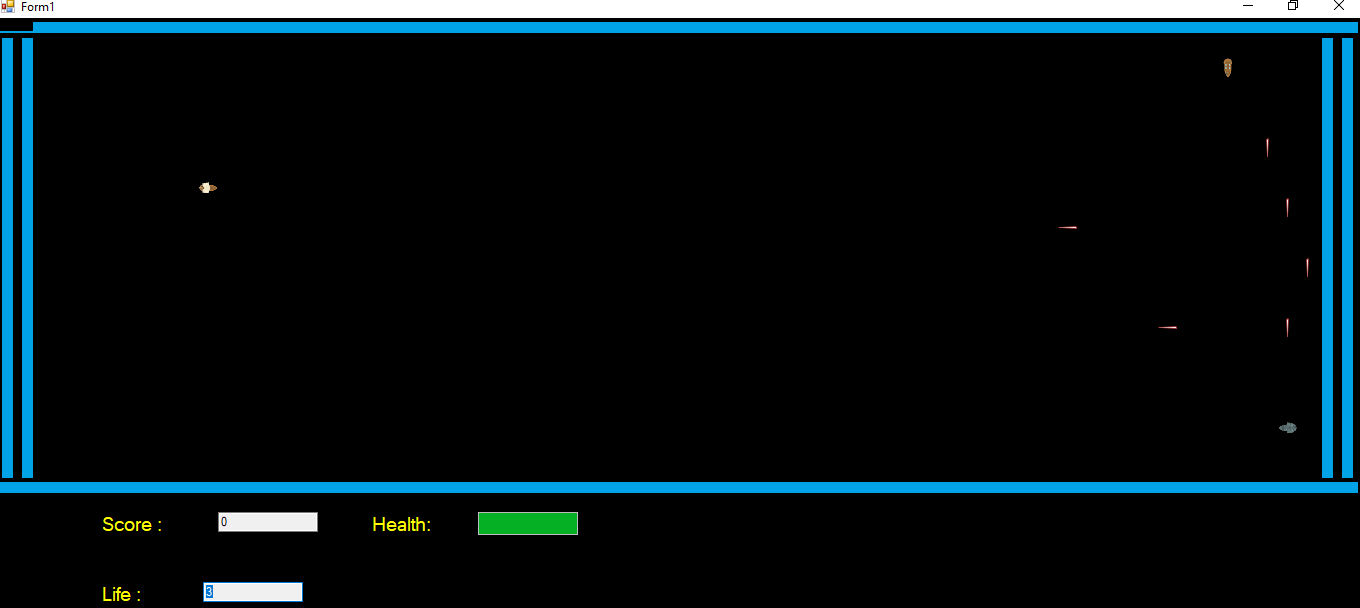


Figure : 2



Figure :3

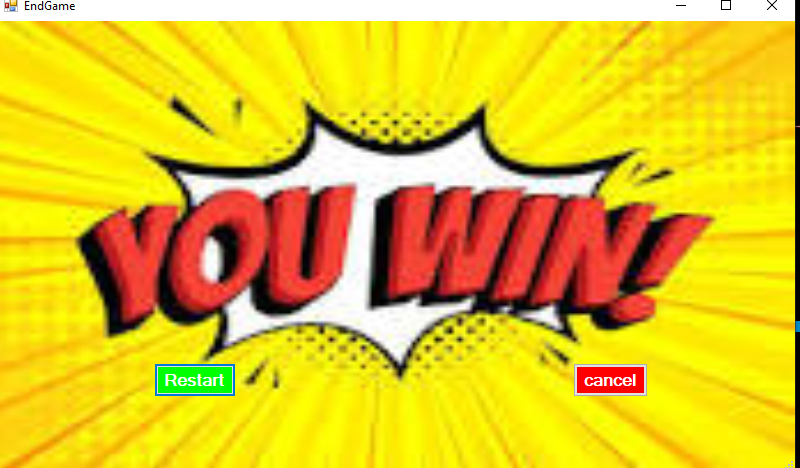


Figure : 4