Bro vs Aliens

Session: 2023 – 2027

Submitted by:

Abu Tayyab 2023-CS-54

Supervised by:

Prof. Maida Shahid

Course:

CSC-103 Object Oriented Programming

Department of Computer Science

University of Engineering and Technology

Lahore Pakistan

Contents

Table of wireframes:

Bro vs Aliens

1) Background Story:

Bro, our cool space explorer, got lost in a super weird galaxy filled with unfriendly aliens. Now, he needs to shoot his way through these space bullies to find his way back home to Earth. Armed with a cool spaceship, Bro faces tough fights, dodges asteroids, and collects power-ups to beat the big boss alien, General Zorgon.

Along the journey, Bro finds special space gadgets that make his ship even cooler. It's like a space adventure where Bro needs to be a hero and defeat all the aliens to unlock the path back to Earth. Will Bro be able to do it? Let's help Bro make an epic comeback in his Space Rescue!

2) Description of game:

"Bro's Galactic Odyssey" thrusts you into space, maneuvering Bro with arrows and firing at alien foes with the Space key.Close encounters with enemies decrease health, so strategize your moves. Gravity affects bullets, limiting their range to 20 points. Clear levels to confront General Zorgon, the ultimate antagonist. Successfully defeat him to unlock the spaceship and return home. An exciting blend of strategy and action awaits – embark on Bro's journey now! Press any key to commence the adventure

3) Game Character Description:

1) Player:

Meet Bro, an intrepid space adventurer stranded amidst alien adversaries. Equipped with courage and an arsenal of space weaponry, Bro relies on your strategic guidance. Utilize arrow keys to navigate the cosmic terrain and the Space key to unleash devastating firepower upon extraterrestrial foes. Bro's mission is clear: annihilate enemies, avoid treacherous mines, and collect power-ups to enhance his chances of survival. Beware of close encounters with enemies, as they pose a threat to Bro's health.

2) Enemies:

a) Alpha:

Move in coordinated waves, changing direction abruptly. Their collective gravitational pull makes them challenging to predict.

b) Beta:

Will move fastly and try to take his health away. It moves horizontally.

c) Gamma:

This enemy also moves horizontaly and this enemy is not very hard to beat.

d) Dalta:

This is a special type of enemy which is very hard to aim and kill its movement is Diagonal.

e) General Zorgon:

This is the last Enemy that you have to kill in order to go back home but it is very hard to kill General because its movement is vertical.

5)Rules and Regulations:

1. Player:

Bro can move in any direction he wants and can shoots fires horizontally. If bro touch the score pill then score will increase and it touches the bomb its health decreases bro has 3 totals lives and each life has 100 health.

2. Enemies:

There are total 5 enemies and in which Alpha, beta, gamma are the enemies with horizontal movement and Dalta can move diagonally and the General can move vertically

When even enemy touches with the player Health Decreases.

Every enemy will destroy after taking different bullets.

6)Objective of the Game:

The objective of the game is to kill and the enemy so that bro can go to earth his home.

7)Wire Frames:

Figure 1Main Menu

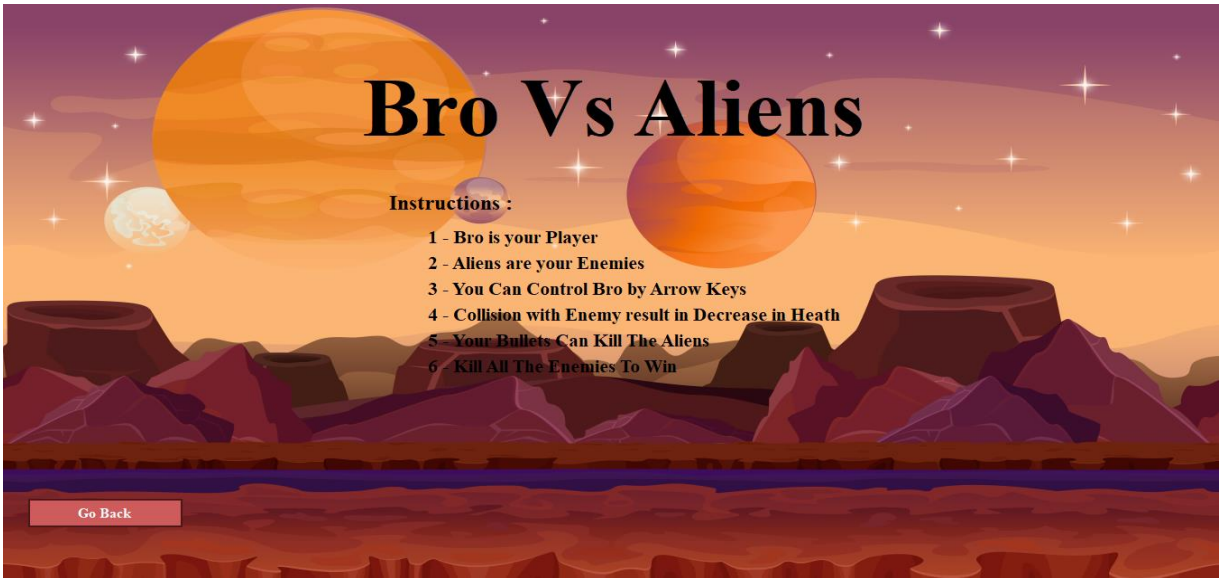


Figure 2Instructions

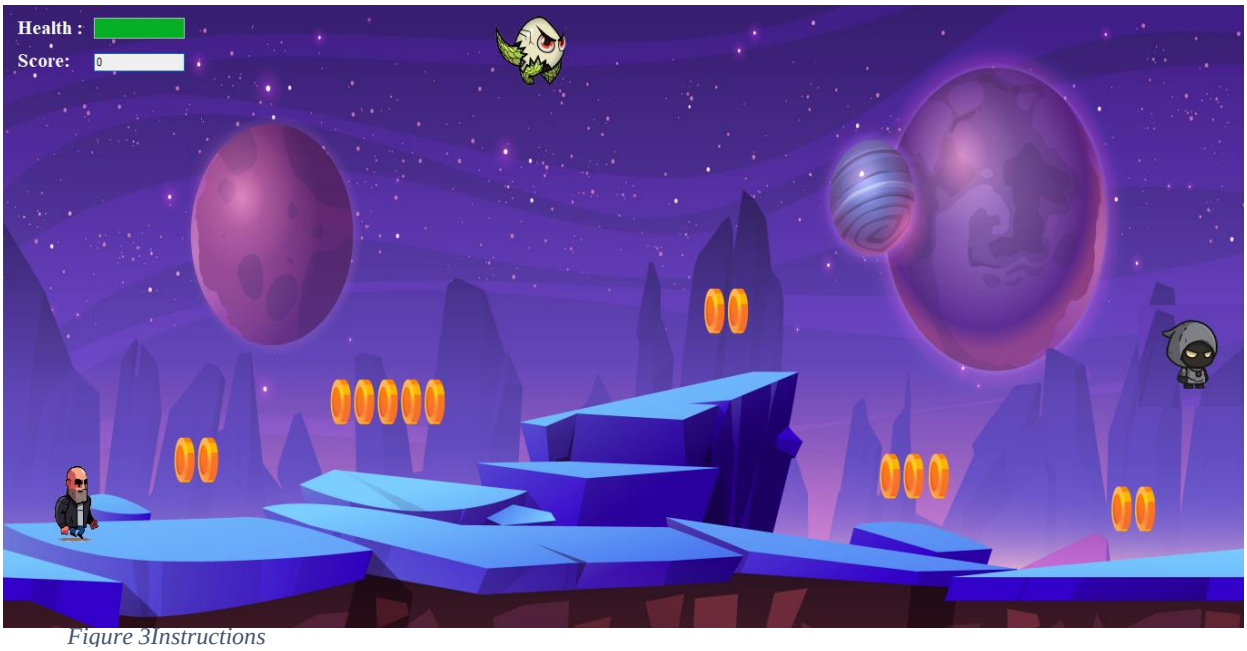


Figure 3Instructions

Figure 4Level 1

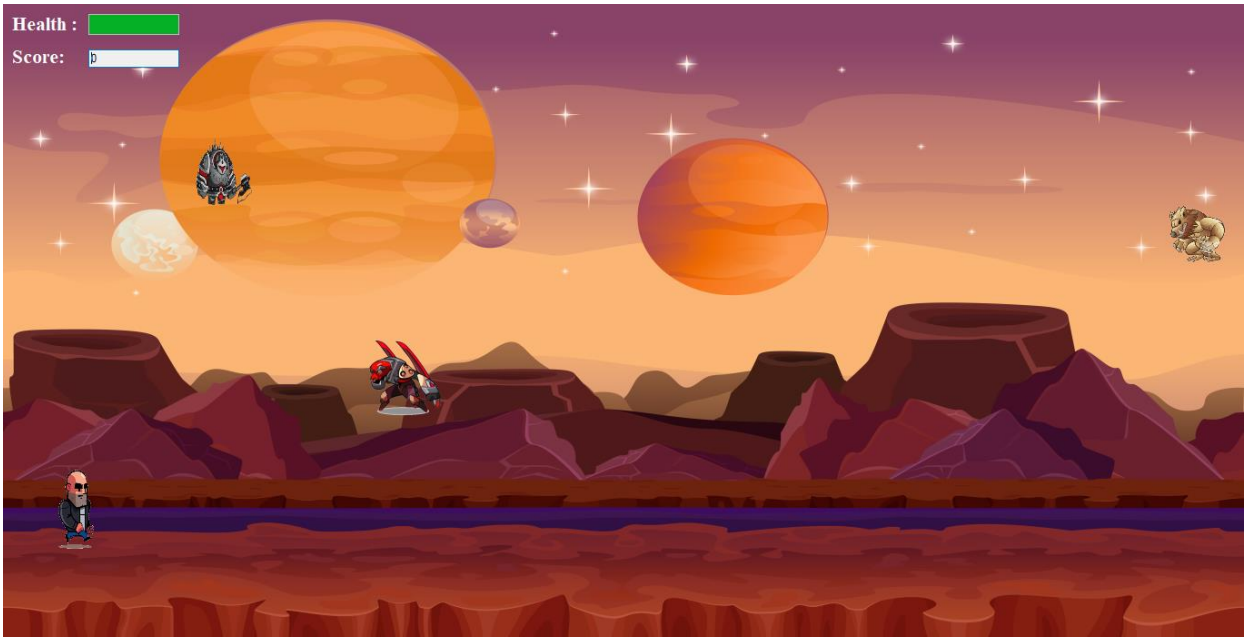
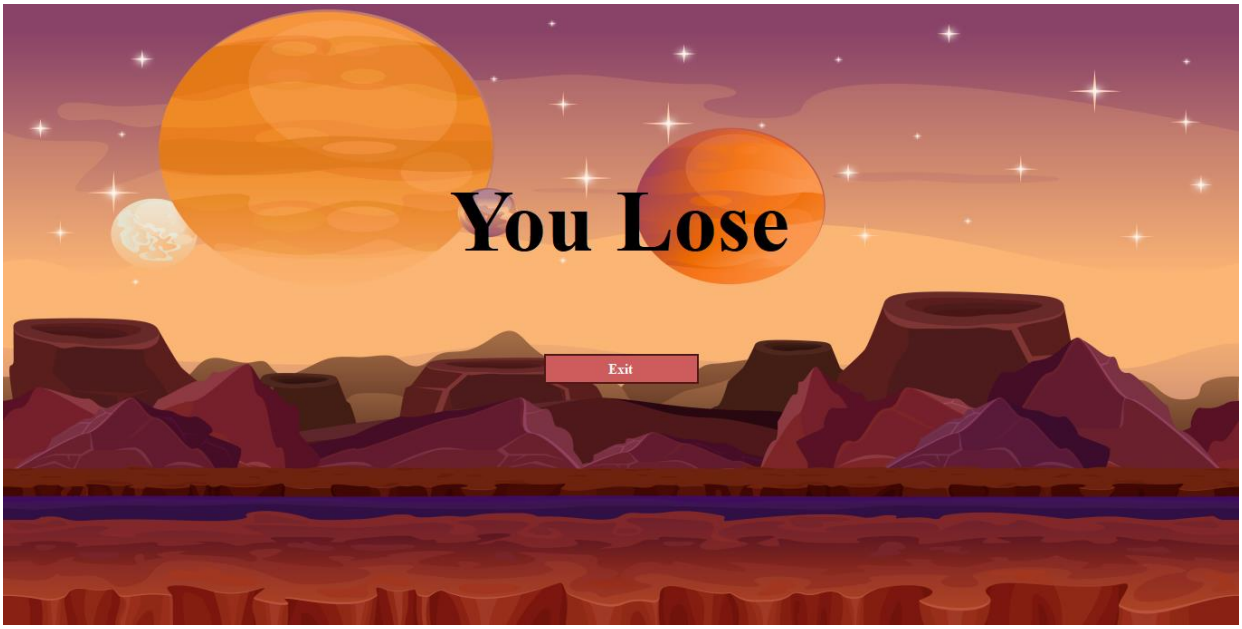
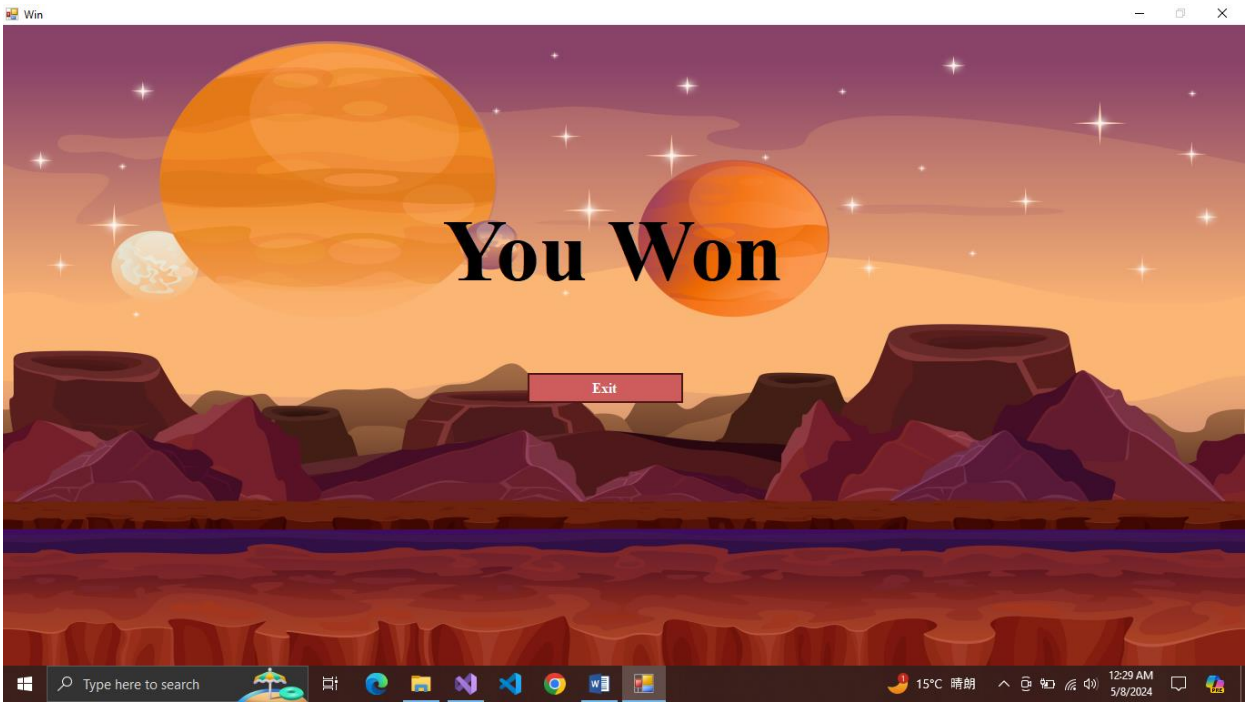


Figure 5Level 2





re 6Won form

Figure 7Lose Form

8)Complete Code:

```
            using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GameDLL.Core
{
   public class CollisionDetection
   {

      private GameObjectType objectType1;
      private GameObjectType objectType2;
      private GameAction action;

      public CollisionDetection(GameObjectType objectType1, GameObjectType objectType2,
GameAction action)
      {
         this.objectType1 = objectType1;
         this.objectType2 = objectType2;
         this.action = action;
      }

      public string CheckCollision(GameObject object1, GameObject object2)
      {
         if (object1.type1 == objectType1 && object2.type1 == objectType2)
         {
            if (object1.GetPictureBox().Bounds.IntersectsWith(object2.GetPictureBox().Bounds))
            {
               switch (action)
               {
                  case GameAction.IncreasePoints:
                     return "Increase points logic";
                  case GameAction.DecreasePoints:
                     return "Decrease points logic";
                  default:
                     return "Default action";
               }
```

```
            }
        }
        return "";
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GameDLL.Core
{
  public enum Direction
  {
    Left,
    Right,
    Up,
    Down
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GameDLL.Core
{
  public enum GameAction
  {
    IncreasePoints,
    DecreasePoints
  }
}
using GameDLL.Interface;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Resources;
```

```
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using EZInput;
using System.Net.Configuration;

namespace GameDLL.Core
{
   public class GameBL
   {
      List<GameObject> gameObjects;
      Form container;
      List<CollisionDetection> collisionDetections;
      static GameBL Instance;

      public static GameBL GetInstance(Form container)
      {
         if(Instance == null)
         {
            Instance = new GameBL(container);
         }
         return Instance;
      }
      private GameBL(Form container)
      {
         this.container = container;
         gameObjects = new List<GameObject>();
         collisionDetections = new List<CollisionDetection>();
      }


      /*public GameBL(Form container)
      {
         gameObjects = new List<GameObject>();
         this.container = container;
         collisionDetections = new List<CollisionDetection>();
      }*/

      public void AddGameobject(Image img,int left,int top,IMovement controller,
GameObjectType type)
      {
         GameObject obj = new GameObject(img,left,top,controller,type);
```

```
            gameObjects.Add(obj);
            container.Controls.Add(obj.GetPictureBox());
        }
        public void CreateBullet(Image image)
        {
            if (EZInput.Keyboard.IsKeyPressed(Key.Space))
            {
                int top = 0, left = 0;
                foreach (GameObject objj in gameObjects)
                {
                    if (objj.type1 == GameObjectType.Player)
                    {
                        top = objj.GetPictureBox().Top;
                        left = objj.GetPictureBox().Left;
                        break;
                    }
                }
                IbulletMovement bullet = new UpwardBullet(25);
                GameObject obj = new GameObject(image, left+50, top, bullet,
GameObjectType.Bullet1);
                gameObjects.Add(obj);
                container.Controls.Add(obj.GetPictureBox());
            }
        }
        public void AddCollision(CollisionDetection collisionDetection)
        {
            collisionDetections.Add(collisionDetection);
        }
        public string CheckCollisions()
        {
            string msg = "";

            for (int i = 0; i < gameObjects.Count; i++)
            {
                GameObject object1 = gameObjects[i];

                for (int j = i + 1; j < gameObjects.Count; j++)
                {
                    GameObject object2 = gameObjects[j];

                    foreach (CollisionDetection collision in collisionDetections)
                    {
```

```
                if
(object1.GetPictureBox().Bounds.IntersectsWith(object2.GetPictureBox().Bounds))
                {
                    if (object1.type1 == GameObjectType.Enemy1 &&
                        object2.type1 == GameObjectType.Enemy2)
                    {
                        // Swap controllers between Enemy1 and Enemy2
                        IMovement tempController = object1.Controller;
                        object1.Controller = object2.Controller;
                        object2.Controller = tempController;
                        gameObjects[i].Controller = object1.Controller;
                        gameObjects[j].Controller = tempController;
                    }
                    else if (object1.type1 == GameObjectType.Enemy1 &&
                        object2.type1 == GameObjectType.Enemy3)
                    {
                        // Swap controllers between Enemy1 and Enemy2
                        IMovement tempController = object1.Controller;
                        object1.Controller = object2.Controller;
                        object2.Controller = tempController;
                        gameObjects[i].Controller = object1.Controller;
                        gameObjects[j].Controller = tempController;
                    }
                }

                msg = collision.CheckCollision(object1, object2);
                if (!string.IsNullOrEmpty(msg))
                {
                    if (msg == "Increase points logic")
                    {
                        object2.GetPictureBox().Visible = false;
                        object1.GetPictureBox().Visible = false;
                        gameObjects.Remove(object1);
                        gameObjects.Remove(object2);
                    }
                    return msg;
                }
            }
        }
    }

    return msg;
```

```
        }

        public void Update()
        {
            foreach(GameObject obj in gameObjects)
            {
                if(obj.type1 == GameObjectType.Bullet1)
                {
                    obj.Updatee();
                    continue;
                }
                obj.Update();
            }
        }
    }
}
using EZInput;
using GameDLL.Interface;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace GameDLL.Core
{
    public class GameObject
    {
        public PictureBox pictureBox;
        private IMovement controller;
        private IbulletMovement controllerr;
        private GameObjectType Type;
        public GameObjectType type1 { get => Type; set => Type = value; }

        public GameObject(Image img,int left,int top,IMovement controller, GameObjectType
type)
        {
            this.Type = type;
            pictureBox = new PictureBox();
            pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
```

```
            pictureBox.Image = img;
            pictureBox.Left = left;
            pictureBox.Top = top;
            pictureBox.Width = 120;
            pictureBox.Height = 112;
            pictureBox.BackColor = Color.Transparent;
            this.controller = controller;
        }
        public GameObject(Image img, int left, int top, IbulletMovement controller,
GameObjectType type)
        {
            this.Type = type;
            pictureBox = new PictureBox();
            pictureBox.SizeMode = PictureBoxSizeMode.StretchImage;
            pictureBox.Image = img;
            pictureBox.Left = left;
            pictureBox.Top = top;
            pictureBox.Width = 70;
            pictureBox.Height = 65;
            pictureBox.BackColor = Color.Transparent;
            this.controllerr = controller;
        }
        public void Update()
        {
            this.pictureBox.Location = controller.move(this.pictureBox.Location);
        }
        public void Updatee()
        {
            this.pictureBox.Location = controllerr.move(this.pictureBox.Location);
        }
        public PictureBox GetPictureBox()
        {
            return pictureBox;
        }

        public IMovement Controller { get => controller; set => controller = value; }
    }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Threading.Tasks;

namespace GameDLL.Core
{
   public enum GameObjectType
   {
      Player,
      Enemy1,
      Enemy2,
      Enemy3,
      Bullet1
   }
}
using GameDLL.Interface;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GameDLL.Core
{
   public class HorizontalPatrol : IMovement
   {
      private int speed;
      private int offset = 200;
      private Point boundry;
      private string direction;

      public HorizontalPatrol(int speed, Point boundry, string direction)
      {
         this.speed = speed;
         this.boundry = boundry;
         this.direction = direction;
      }
      public Point move(Point Location)
      {

         if ((Location.X + speed) >= boundry.X - offset)
         {
            direction = "Left";
```

```
            }
            else if ((Location.X - speed) <= 300)
            {
                direction = "Right";
            }

            if (direction == "Left")
            {
                Location.X -= speed;
            }
            else
            {
                Location.X += speed;
            }

            return Location;

        }
    }
}
using EZInput;
using GameDLL.Interface;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Point = System.Drawing.Point;

namespace GameDLL.Core
{
    public class KeyboardMovement : IMovement
    {
        private int speed;
        private Point boundry;
        private int offset;

        public KeyboardMovement(int speed, Point boundry)
        {
            this.speed = speed;
            this.boundry = boundry;
```

```
        this.offset = 50;
    }

    public Point move(Point Location)
    {
        if(EZInput.Keyboard.IsKeyPressed(Key.UpArrow))
        {
            if(Location.Y + speed > 50)
            {
                Location.Y -= speed;
            }
        }
        if (EZInput.Keyboard.IsKeyPressed(Key.DownArrow))
        {
            if (Location.Y + speed < boundry.Y - 100)
            {
                Location.Y += speed;
            }
        }
        if (EZInput.Keyboard.IsKeyPressed(Key.LeftArrow))
        {
            if (Location.X + speed > 50 )
            {
                Location.X -= speed;
            }
        }
        if (EZInput.Keyboard.IsKeyPressed(Key.RightArrow))
        {
            if (Location.X + speed < boundry.X - 50)
            {
                Location.X += speed;
            }
        }
        return Location;
    }
  }
}
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
```

```
using GameDLL.Interface;
using Point = System.Drawing.Point;
using EZInput;


namespace GameDLL.Core
{
   public class UpwardBullet : IbulletMovement
   {
      private int speed;

      public UpwardBullet(int speed)
      {
         this.speed = speed;
      }

      public Point move(Point Location)
      {
         Location.Y -= speed;
         return Location;
      }
   }
}
using GameDLL.Interface;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GameDLL.Core
{
   public class VerticalPatrol : IMovement
   {
      private int speed;
      private int offset = 200;
      private Point boundry;
      private string direction;

      public VerticalPatrol(int speed, Point boundry, string direction)
      {
```

```
            this.speed = speed;
            this.boundry = boundry;
            this.direction = direction;
        }

        public Point move(Point Location)
        {

            if ((Location.Y + offset) >= boundry.Y)
            {
                direction = "Up";
            }
            else if ((Location.Y - speed) <= 0)
            {
                direction = "Down";
            }

            if (direction == "Up")
            {
                Location.Y -= speed;
            }
            else
            {
                Location.Y += speed;
            }

            return Location;
        }
    }
}
using GameDLL.Interface;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace GameDLL.Core
{
    public class ZigZagMovement : IMovement
    {
```

```
private int speed,count,offset = 150;
private Point boundry;
private string direction;

public ZigZagMovement(int speed, Point boundry, string direction)
{
   this.speed = speed;
   this.boundry = boundry;
   this.direction = direction;
   this.count = 0;
}


public Point move(Point Location)
{

   if ((Location.X + offset) > boundry.X)
   {
      direction = "Left";
   }
   else if ((Location.X + speed) <= 0)
   {
      direction = "Right";
   }
   if (count == 10)
   {
      count = 0;
   }


   if (direction == "Right")
   {
      if (count < 5)
      {
         Location.X += speed;
         Location.Y -= speed;
      }
      else if (count >= 5 && count < 10)
      {
         Location.X += speed;
         Location.Y += speed;
      }
```

```
            }
            else if (direction == "Left")
            {
                if (count < 5)
                {
                    Location.X -= speed;
                    Location.Y += speed;
                }
                else if (count >= 5 && count < 10)
                {
                    Location.X -= speed;
                    Location.Y -= speed;
                }
            }
            count++;
            return Location;
        }
    }
}
using Game.Properties;
using GameDLL.Core;
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;
using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace Game
{
    public partial class Level1 : Form
    {
        GameBL Game;
        int Count = 0;
        int Score = 0;

        public Level1()
        {
```

```
        InitializeComponent();
        Game = GameBL.GetInstance(this);

        GameLoop.Interval = 100; // Adjust the interval as needed
        GameLoop.Start();
        progressBar1.Value = 100;

    }

    private void GameLoop_Tick(object sender, EventArgs e)
    {
        Game.CreateBullet(Resources.Bullet);
        Game.Update();
        LoadScore();
        string msg = Game.CheckCollisions();
        if (!string.IsNullOrEmpty(msg) && progressBar1.Value >= 10)
        {
            if (msg == "Increase points logic")
            {
                Score += 100;
                Count++;
                if (Count == 2)
                {
                    this.Hide();
                    Win form = new Win();
                    form.Show();

                }
            }
            else if (msg == "Decrease points logic")
            {
                progressBar1.Value -= 20;
                if(progressBar1.Value <= 0)
                {
                    this.Hide();
                    Lose form = new Lose ();
                    form.Show();
                }
            }
        }
    }
```

```
        private void Level1_Load(object sender, EventArgs e)
        {
            Point p = new Point(this.ClientSize.Width, this.ClientSize.Height);

            KeyboardMovement player = new KeyboardMovement(50, p);
            HorizontalPatrol e1 = new HorizontalPatrol(30, p, Direction.Right.ToString());
            VerticalPatrol e2 = new VerticalPatrol(30, p, Direction.Down.ToString());
            //ZigZagMovement e3 = new ZigZagMovement(30, p, Direction.Right.ToString());

            Game.AddGameobject(Resources.P1, 20, 500, player, GameObjectType.Player);

            Game.AddGameobject(Resources.Enemy1, 200, 0, e1, GameObjectType.Enemy1);
            Game.AddGameobject(Resources.Enemy2, 1250, 10, e2, GameObjectType.Enemy2);
            //Game.AddGameobject(Resources.Enemy3, 0, 200, e3, GameObjectType.Enemy3);

            CollisionDetection collision = new CollisionDetection(GameObjectType.Player,
GameObjectType.Enemy1, GameAction.DecreasePoints);
            Game.AddCollision(collision);
            CollisionDetection collision1 = new CollisionDetection(GameObjectType.Player,
GameObjectType.Enemy2, GameAction.DecreasePoints);
            Game.AddCollision(collision1);
            //CollisionDetection collision2 = new CollisionDetection(GameObjectType.Player,
GameObjectType.Enemy3, GameAction.DecreasePoints);
            //Game.AddCollision(collision2);
            CollisionDetection collision3 = new CollisionDetection(GameObjectType.Enemy1,
GameObjectType.Bullet1, GameAction.IncreasePoints);
            Game.AddCollision(collision3);
            CollisionDetection collision4 = new CollisionDetection(GameObjectType.Enemy2,
GameObjectType.Bullet1, GameAction.IncreasePoints);
            Game.AddCollision(collision4);
            //CollisionDetection collision5 = new CollisionDetection(GameObjectType.Enemy3,
GameObjectType.Bullet1, GameAction.IncreasePoints);
            //Game.AddCollision(collision5);

        }
        private void LoadScore()
        {
            textBox1.Text = Score.ToString();
        }
    }
}
```