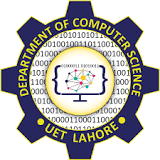
**Game Project**

****

**Session 2023-2027**

**Submitted by:**

Anza Tamveel 2023-CS-87

**Supervised to:**

Sir Laeeq Khan

**Course:**

OOP

Department of Computer Science

**University of Engineering and Technology**

**Lahore**

* **About Game:**

In your Windows Form game project, you're crafting an intense survival experience where the player faces off against three formidable enemies amidst a landscape dotted with obstacles and hurdles. The game puts the player's reflexes, strategy, and precision to the test as they strive to outmaneuver their adversaries and emerge as the ultimate survivor.

As the player navigates the game environment using keyboard controls, they must dodge obstacles while strategically positioning themselves to take down the approaching enemies. Armed with a limited supply of bullets, every shot counts, and accuracy is key to defeating the enemies and progressing to the next level.

The enemies in your game project exhibit distinct behaviors and attributes, requiring the player to adapt their tactics accordingly. Whether it's evading the swift movements of a nimble foe, breaking through the defenses of a heavily armored adversary, or outsmarting the cunning maneuvers of a strategic opponent, the player must stay on their toes to stay alive.

Throughout the game, the player will encounter various power-ups and bonuses strategically placed within the environment. These could include extra ammunition, temporary invincibility, or speed boosts, providing the player with much-needed advantages to overcome the increasingly challenging obstacles and enemies they face.

With its simple yet engaging gameplay mechanics, your game project offers a thrilling experience that will keep players immersed and entertained as they strive to survive against overwhelming odds. As the sole developer of this exciting venture, you have the opportunity to shape every aspect of the game to deliver a memorable and enjoyable gaming experience for your audience.

* **Objectives of game:**

My objective in creating this game project could be multifaceted, incorporating elements such as:

**Skill Development**: Creating the game to improve your programming skills, particularly in Windows Form development, game logic, and user interface design.

**Portfolio Enhancement**: Building a polished game project to showcase your abilities to potential employers or collaborators in the game development industry.

**Personal Challenge**: Setting a personal challenge to see how far you can push your creativity and technical skills in designing and implementing a game from scratch.

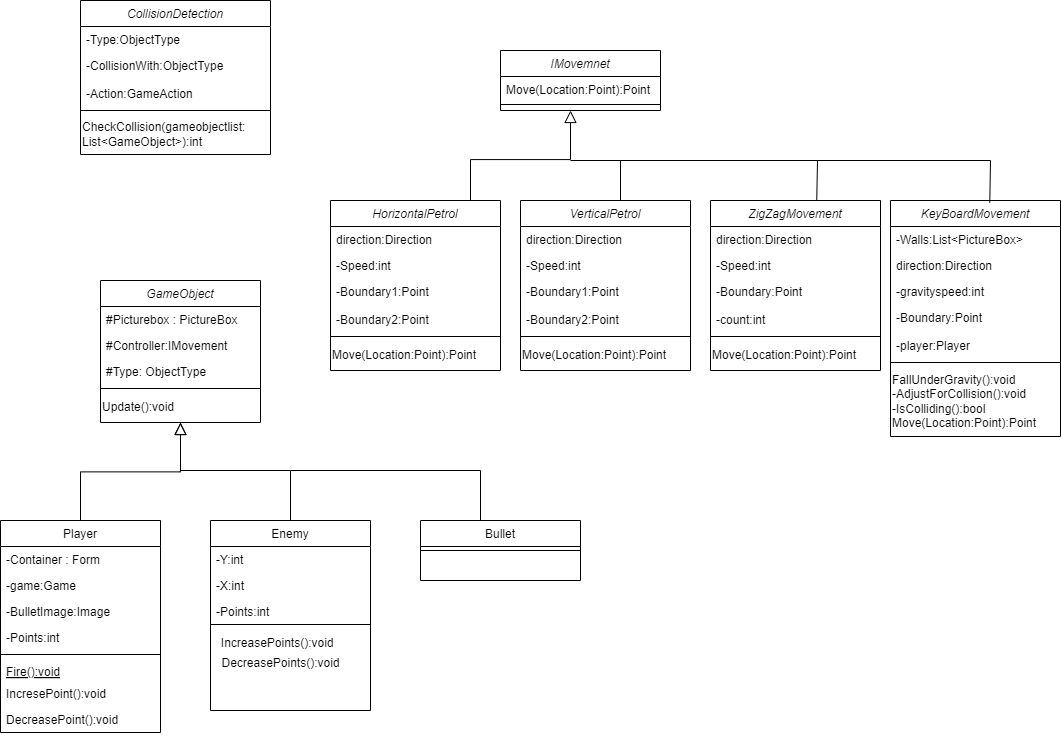
**Learning Experience**: Using the project as an opportunity to learn new techniques, tools, or programming languages that you can apply to future projects.

**Community Engagement**: Engaging with the gaming community by sharing your progress, seeking feedback, and collaborating with other developers to improve and refine your game.

**Creative Outlet:** Expressing your creativity and imagination through game design, storytelling, and visual aesthetics, providing an outlet for your passion for gaming and technology.

**Entertainment:** Ultimately, the goal could simply be to create an enjoyable and entertaining experience for players, offering them a challenging and immersive gameplay experience that they can enjoy and appreciate.

* **CRC:**

****

* **Wireframes:**

****

* **Code:**

using GameFrameWork01.Movement;

using System.Collections.Generic;

using System.Drawing;

using System.Windows.Forms;

namespace GameFrameWork01

{

public class Game

{

static Game Instance;

public static Game GetGameInctance(Form Container)

{

if (Instance == null)

{

Instance = new Game(Container);

}

return Instance;

}

public Game()

{

}

List<GameObject> gameObjectsList;

List<PictureBox> HurdlesList = new List<PictureBox>();

List<CollisionDetection> CollisionsPerformed = new List<CollisionDetection>();

Form Container;

public void AddgameObjectsList(GameObject Object)

{

gameObjectsList.Add(Object);

}

public void AddCollision(CollisionDetection Collision)

{

CollisionsPerformed.Add(Collision);

}

public List<PictureBox> GetHurdleList()

{

return HurdlesList;

}

public void AddhurdleList(List<PictureBox> Hurdles)

{

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

{

HurdlesList.Add(Hurdles[i]);

}

}

public int returnobjectcounts(ObjectType type)

{

int count = 0;

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

{

if (gameObjectsList[i].Type == type)

{

count++;

}

}

return count;

}

private Game(Form Container)

{

this.Container = Container;

this.gameObjectsList = new List<GameObject>();

}

public void AddPlayer(ProgressBar Progressbar, Image image, int Left, int Top, KeyboardMovement Controller, ObjectType Type, Image FireImage)

{

if (Type == ObjectType.Player)

{

Player player = new Player( Progressbar, image, Left, Top, Controller, Type , FireImage , this.Container);

Controller.Player = player;

gameObjectsList.Add(player);

Container.Controls.Add(player.Pb);

}

}

public void AddGameObject(Image image, int Left, int Top, IMovement Controller, ObjectType Type)

{

if (Type == ObjectType.Enemy)

{

Enemy enemy = new Enemy(Left, Top, image, Left, Top, Controller, Type);

gameObjectsList.Add(enemy);

Container.Controls.Add(enemy.Pb);

}

}

public void Update()

{

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

{

gameObjectsList[i].update();

}

foreach (CollisionDetection Col in CollisionsPerformed)

{

int index = Col.CheckCollision(gameObjectsList);

if (index >= 0)

{

Container.Controls.Remove(gameObjectsList[index].Pb);

gameObjectsList.RemoveAt(index);

}

}

}

}

}

using System.Drawing;

using System.Windows.Forms;

namespace GameFrameWork01

{

public class GameObject

{

protected PictureBox pb;

protected IMovement Controller;

protected ObjectType type;

public GameObject(Image image, int Left, int Top, IMovement Controller, ObjectType Type)

{

this.type = Type;

Pb = new PictureBox();

Pb.Image = image;

Pb.Width = image.Width;

Pb.Height = image.Height;

Pb.BackColor = Color.Transparent;

Pb.Left = Left;

Pb.Top = Top;

this.Controller = Controller;

}

public GameObject()

{

}

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

public PictureBox Pb { get => pb; set => pb = value; }

public void update()

{

this.Pb.Location = Controller.Move(this.Pb.Location);

}

}

}

using System.Collections.Generic;

using System.Windows.Forms;

namespace GameFrameWork01.Movement

{

public class KeyboardMovement : IMovement

{

private int speed;

private List<PictureBox> Walls;

private System.Drawing.Point boundary;

private int gravitySpeed = 10;

private Player player;

public Player Player { get => player; set => player = value; }

public KeyboardMovement(List<PictureBox> walls, int speed, System.Drawing.Point boundary, Player player)

{

this.speed = speed;

this.Walls = walls;

this.boundary = boundary;

this.Player = player;

}

public KeyboardMovement(List<PictureBox> walls, int speed, System.Drawing.Point boundary)

{

this.speed = speed;

this.Walls = walls;

this.boundary = boundary;

}

public void FallUnderGravity(ref System.Drawing.Point Location)

{

bool Fall = true;

foreach (PictureBox wall in Walls)

{

if (IsColliding(new System.Drawing.Point(Location.X, Location.Y + gravitySpeed), wall))

{

Fall = false;

}

}

if (Fall)

{

Location.Y += gravitySpeed;

}

}

public System.Drawing.Point Move(System.Drawing.Point location)

{

System.Drawing.Point newLocation = new System.Drawing.Point(location.X, location.Y);

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

{

newLocation.Y -= 15;

}

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

{

newLocation.Y += speed;

}

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

{

newLocation.X -= speed;

}

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

{

newLocation.X += speed;

}

if (EZInput.Keyboard.IsKeyPressed(EZInput.Key.Space))

{

Player.Fire();

}

newLocation = AdjustForCollisions(newLocation);

FallUnderGravity(ref newLocation);

return newLocation;

}

private System.Drawing.Point AdjustForCollisions(System.Drawing.Point newLocation)

{

foreach (PictureBox wall in Walls)

{

if (IsColliding(newLocation, wall))

{

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

{

newLocation.Y = wall.Bottom + 1;

}

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

{

newLocation.Y = wall.Top - 65;

}

else if (EZInput.Keyboard.IsKeyPressed(EZInput.Key.LeftArrow))

{

newLocation.X = wall.Right + 1;

}

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

{

newLocation.X = wall.Left - 65;

}

}

}

if (newLocation.X < 0)

{

newLocation.X = 0;

}

else if (newLocation.X > boundary.X)

{

newLocation.X = boundary.X;

}

if (newLocation.Y < 0)

{

newLocation.Y = 0;

}

else if (newLocation.Y > boundary.Y)

{

newLocation.Y = boundary.Y;

}

return newLocation;

}

private bool IsColliding(System.Drawing.Point location, PictureBox wall)

{

return location.X < wall.Right &&

location.X + 65 > wall.Left &&

location.Y < wall.Bottom &&

location.Y + 65 > wall.Top;

}

}

}

using System.Collections.Generic;

using System.Windows.Forms;

namespace GameFrameWork01

{

public class CollisionDetection

{

ObjectType Object;

ObjectType CollisionWith;

GameAction Action;

public CollisionDetection(ObjectType Object, ObjectType CollisionWith, GameAction Action)

{

this.Object = Object;

this.CollisionWith = CollisionWith;

this.Action = Action;

}

public int CheckCollision(List<GameObject> gameObjectsList)

{

for (int x = 0; x < gameObjectsList.Count; x++)

{

if (gameObjectsList[x].Type == Object)

{

for (int y = 0; y < gameObjectsList.Count; y++)

{

if (gameObjectsList[y].Type == CollisionWith && gameObjectsList[x] != gameObjectsList[y])

{

if (gameObjectsList[x].Pb.Bounds.IntersectsWith(gameObjectsList[y].Pb.Bounds))

{

if (gameObjectsList[x] is Player)

{

Player player = (Player)gameObjectsList[x];

if (Action == GameAction.IncreasePoint)

{

player.IncreasePoint();

}

else if (Action == GameAction.DecresePoint)

{

int points = player.DecreasePoints();

if (points <= 0)

{

return y;

}

}

}

else if (gameObjectsList[x] is Enemy)

{

Enemy enemy = (Enemy)gameObjectsList[x];

if (Action == GameAction.IncreasePoint)

{

enemy.IncreasePoint();

}

else if (Action == GameAction.DecresePoint)

{

int points = enemy.DecreasePoints();

if (points <= 0)

{

// MessageBox.Show("if true");

enemy.Pb.Visible = false;

return x;

}

}

}

if (Object == ObjectType.Enemy && CollisionWith == ObjectType.Bullet)

{

if (gameObjectsList[y] is Bullet)

{

Bullet bullet = (Bullet)gameObjectsList[y];

bullet.Pb.Visible = false;

return y;

}

}

}

}

}

}

}

return -1;

}

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Drawing;

using System.Windows.Forms;

namespace GameFrameWork01

{

* **Future Directions:**

In future iterations, we plan to expand Last Stand with new levels, enemies, and power-ups, providing players with fresh challenges and experiences. Additionally, we're exploring the integration of multiplayer features to allow for competitive or cooperative play, enhancing social interaction. Furthermore, we aim to expand the game's reach by considering platform expansion, potentially bringing Last Stand to mobile devices and consoles to reach a broader audience**.**