Experiment No.6
Develop a simple UI(User interface) menu with images, canvas, sprites and button.Write a C# program to interact with UI menu through VR trigger button such that on each successful trigger interaction displays a score on scene.
Date of Performance: 05/09/23
Date of Submission: 19/10/23

AIM: -

Develop a simple UI(User interface) menu with images, canvas, sprites and buttons. Write a C# program to interact with UI menu through VR trigger button such that on each successful trigger interaction displays a score on scene .

OBJECTIVES: -

This Unity project aimed to create an interactive VR user interface (UI) featuring images, buttons, and a scoring system. It involved setting up a Canvas to host UI elements, scripting in C# to enable interactions via a VR trigger button, and dynamically displaying scores on the scene upon successful triggers. The project emphasized the integration of VR technology with Unity's UI capabilities, enhancing user engagement. Rigorous testing ensured seamless VR interactions, while code optimization prioritized performance. Comprehensive documentation was provided to aid future reference and collaboration. In summary, this project equipped developers with valuable skills in VR UI design and scripting, enriching their capabilities in VR application development.

THEORY: -

The theory behind developing a UI menu in Unity with images, canvas, sprites, buttons, and integrating it with a VR trigger button to display a score upon interaction involves several key concepts:

1. Unity UI Components:
    a. Canvas: A Canvas is a fundamental UI component in Unity. It acts as a container for all other UI elements and provides a plane on which UI elements are rendered.
    b. Images: Images are UI elements used to display graphics or visual elements on the Canvas. They can represent background images, icons, or any visual content.
    c. Sprites: Sprites are 2D images or graphics used in Unity to represent objects or elements within the game or UI.
    d. Buttons: Buttons are interactive UI elements that users can click or, in the case of VR, trigger interactions with.
2. C# Scripting:
    a. C# Programming: Unity uses C# scripting to create interactive behaviors. In this context, C# scripts are written to handle the interactions between the VR trigger button and the UI elements.
    b. VR Integration: Integration with VR hardware, such as Oculus or HTC Vive, involves capturing input events like trigger presses.
3. Interactivity:
    a. VR Trigger Interaction: The VR trigger button serves as the input mechanism for interacting with the UI menu. When the trigger is pressed or activated, it initiates

an interaction event.

b. Score Display: Upon a successful interaction event, the C# script updates and displays a score on the UI Canvas within the scene.

4. Testing and Optimization:
   a. Testing: Rigorous testing is essential to ensure that the VR trigger interactions function correctly, and the score display updates accurately.
   b. Performance Optimization: Optimization of code and UI elements ensures smooth and responsive interactions within the VR environment.
5. Documentation:
   a. Code Documentation: Comprehensive documentation of the C# scripts and UI setup is critical for understanding and maintaining the project and for future reference and collaboration.

In summary, this project combines Unity's UI components, C# scripting, and VR integration to create an interactive UI menu. It demonstrates the potential of VR technology to enhance user engagement by allowing users to interact with UI elements using a VR trigger button and dynamically display scores as a result of successful interactions. This project enriches developers' skills in VR application development and UI design for immersive experiences.

CODE: -

- *Background.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Background : MonoBehaviour {
   public float speed;
   public float Xend;
   public float Xstart;
   private void Update()
   {
      transform.Translate(Vector2.left * speed * Time.deltaTime);
      if (transform.position.x < Xend) {
         Vector2 pos = new Vector2(Xstart, transform.position.y);
         transform.position = pos;
      }
   }
}
```

- *Destroyer.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Destroyer : MonoBehaviour {
   public float lifetime;
   private void Start()
   {
      Destroy(gameObject, lifetime);
   }
}
```

- *Obstacle.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Obstacle : MonoBehaviour {
   public float speed;
   public GameObject effect;
         void Update () {
      transform.Translate(Vector2.left * speed * Time.deltaTime);
         }
   private void OnTriggerEnter2D(Collider2D other)
   {
      if (other.CompareTag("Player")) {
         other.GetComponent<Player>().health--;
         other.GetComponent<Player>().camAnim.SetTrigger("shake");
         Instantiate(effect, transform.position, Quaternion.identity);
         Destroy(gameObject);
      }
   }
}
```

- *Obstacle Spawn.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class ObstacleSpawn : MonoBehaviour {
   public GameObject obstacle;
   private void Start()
   {
      Instantiate(obstacle, transform.position, Quaternion.identity);
   }
```

```
}
```

- *Player.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;
public class Player : MonoBehaviour {
   public float speed;
   public float increment;
   public float maxY;
   public float minY;
   private Vector2 targetPos;
   public int health;
   public GameObject moveEffect;
   public Animator camAnim;
   public Text healthDisplay;
   public GameObject spawner;
   public GameObject restartDisplay;
   private void Update()
   {
      if (health <= 0) {
         spawner.SetActive(false);
         restartDisplay.SetActive(true);
         // SceneManager.LoadScene("MainMenu");
         Destroy(gameObject);
      }
      healthDisplay.text = health.ToString();
      transform.position = Vector2.MoveTowards(transform.position, targetPos, speed *
Time.deltaTime);
      if (Input.GetKeyDown(KeyCode.UpArrow) && transform.position.y < maxY) {
         camAnim.SetTrigger("shake");
         Instantiate(moveEffect, transform.position, Quaternion.identity);
         targetPos = new Vector2(transform.position.x, transform.position.y + increment);
      } else if (Input.GetKeyDown(KeyCode.DownArrow) && transform.position.y > minY) {
         camAnim.SetTrigger("shake");
         Instantiate(moveEffect, transform.position, Quaternion.identity);
         targetPos = new Vector2(transform.position.x, transform.position.y - increment);
      }
   }
}
```

- *Restart.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;
public class Restart : MonoBehaviour {
        void Update () {
      if (Input.GetKeyDown(KeyCode.Q)){
        SceneManager.LoadScene("MainMenu");
      }
        }
}
```

- *Score.cs*

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
public class Score : MonoBehaviour {
   public int score;
   public Text scoreDisplay;

   private void Update()
   {
      scoreDisplay.text = score.ToString();
   }
   private void OnTriggerEnter2D(Collider2D other)
   {
      score++;
      Destroy(other.gameObject);
   }
}
```

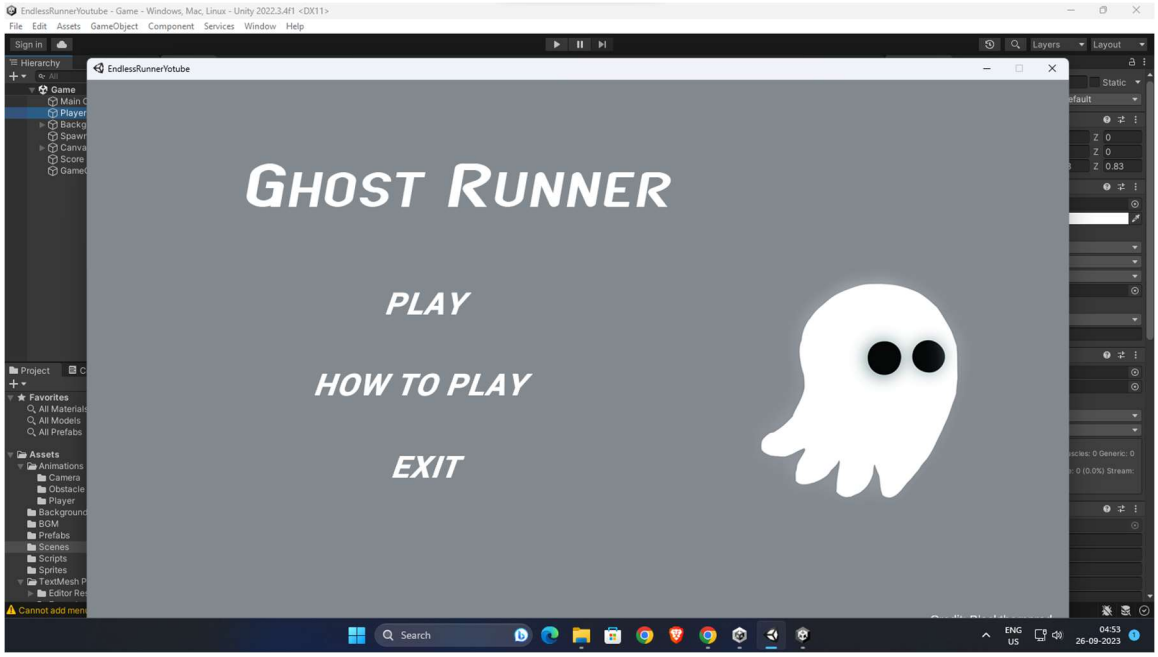
- *Spawner.cs*

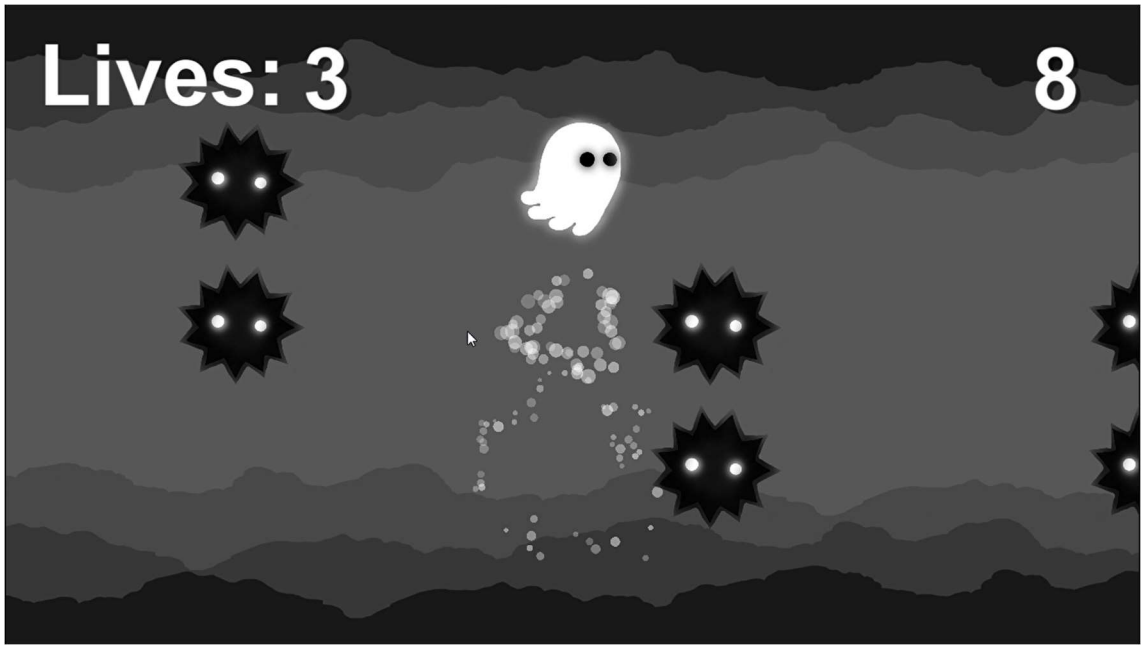
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class Spawner : MonoBehaviour {
   private float timeBtwSpawns;
   public float startTimeBtwSpawns;
   public float timeDecrease;
```

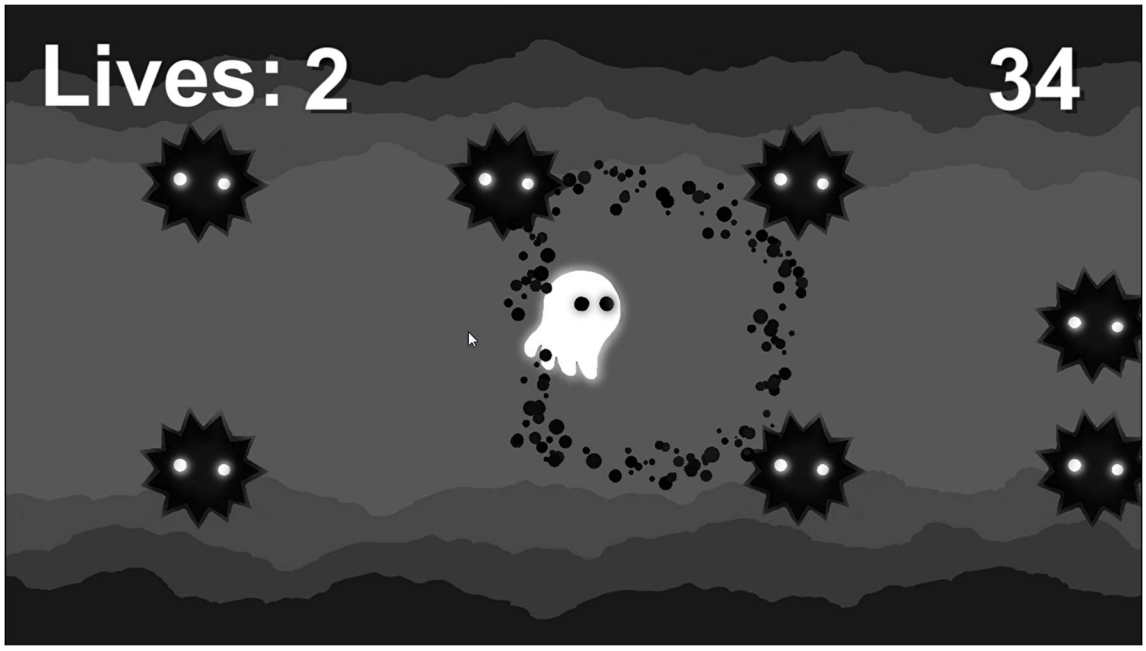
```
    public float minTime;
    public GameObject[] obstacleTemplate;
    private void Start()
    {
        timeBtwSpawns = startTimeBtwSpawns;
    }
    private void Update()
    {
        if (timeBtwSpawns <= 0)
        {
            int rand = Random.Range(0, obstacleTemplate.Length);
            Instantiate(obstacleTemplate[rand], transform.position, Quaternion.identity);
            timeBtwSpawns = startTimeBtwSpawns;
            if (startTimeBtwSpawns > minTime) {
                startTimeBtwSpawns -= timeDecrease;
            }
        }
        else {
            timeBtwSpawns -= Time.deltaTime;
        }
    }

}
```

OUTPUT: -




Lives: 3
8


Lives: 2
34

CONCLUSION: -

This Unity project successfully realized a dynamic user interface (UI) menu, incorporating images, buttons, and a score display mechanism, all seamlessly integrated with VR technology through the VR trigger button. Leveraging C# scripting, the project enabled responsive interactions, where each successful trigger interaction triggered the display of a score within the VR scene. This project highlighted the powerful synergy between Unity's UI components and VR hardware, showcasing the potential for immersive and engaging user experiences. It offered valuable insights into VR application development, user interface design, and the fusion of traditional UI elements with cutting-edge VR technology, reinforcing developers' skills in creating interactive and immersive VR applications.