ACKNOWLEDGEMENT

We would like to express my greatest appreciation to all the individuals who have helped and supported me throughout this project. I am thankful to whole Centre of AI department for their ongoing support during the experiments, from initial advice and provision of contact in the first stages through ongoing advice and encouragement, which led to the final report of this project.

A special acknowledgement goes to my colleagues who helped me in completing the file by exchanging interesting ideas to deal with problems and sharing their experience.

We wish to thank our professor Dr.Vibha Tiwari as well for her undivided support and interests which inspired me and encouraged me to go my own way without whom I would be unable to complete my project.

At the end, we want to thank my friends who displayed appreciation for my work and motivated me to continue my work.

Ankit Jain 0901AI221011

Mayank Verma 0901AI221038

CERTIFICATE

This is certified that Ankit Jain (0901AI221011) and Mayank Verma (0901AI221038) have submitted the project report titled AR/VR Project Report under the guidance of Dr.Vibha Tiwari in partial fulfillment of the requirement for the award of the degree of B.Tech in Information Technology (AI & Robotics) from Madhav Institute of Technology and Science, Gwalior.

Dr. Vibha Tiwari

Assistant Professor

Centre For AI

Dr. Rajani Rajan Makhanvana

Professor & Head

Centre For AI

AR Project Title : AR Flashcards

1. Objectives and Project Description

Project Description :

The “AR Flashcards for Language Learning” project is an Augmented Reality application developed to make vocabulary learning more engaging and visually interactive. Built using Unity 3D and AR frameworks, the project allows users to scan printed flashcards with their smartphone camera and view related 3D images that appear in real space.

Instead of traditional flashcards that rely only on text, this AR-based approach provides a more immersive way of connecting words with visuals. When a user scans a particular flashcard, a corresponding image (such as an animal or object) appears on the screen, helping in better retention and understanding of the word’s meaning.

The project demonstrates how Augmented Reality can be effectively used in education to make learning simple, fun, and interactive—especially for beginners and young learners.

Key Objectives :

The project was driven by a set of key goals focused on creating an engaging and educational Augmented Reality learning experience.

- **Educational Visualization:** To enhance the process of language learning by connecting words with 3D images that appear through AR when a flashcard is scanned.
- **Simple AR Interaction:** To implement an easy-to-use AR experience that detects and displays 3D objects on scanning flashcards.
- **User Engagement:** To make learning more enjoyable and memorable through visual cues rather than plain text.
- **Practical Implementation:** To demonstrate the use of Unity and AR technology in creating an interactive educational tool.

- **Foundation for Future Enhancements:** To design a base that can later include features like audio pronunciation, animations, or text overlays for a richer learning experience.

2. Development Tools Used

The development of the AR Flashcards for Language Learning project utilized a set of modern and efficient tools designed specifically for mobile Augmented Reality applications:

- **Game Engine: Unity 3D (Version 6.2)**
  Unity served as the main development platform for creating the project. It was used to design scenes, manage 3D models, implement scripts, and handle the AR camera setup. Its user-friendly interface and powerful cross-platform support made it ideal for developing AR applications targeting mobile devices.
- **Augmented Reality SDKs: ARCore (for Android) and ARKit (for iOS)**
  These SDKs provided the essential AR functionality, including surface detection, device motion tracking, and accurate placement of 3D images in the real world. They helped achieve a smooth and realistic AR experience by ensuring stable image anchoring and responsive object placement.
- **Primary Scripting Language: C# (C-Sharp)**
  All the interaction logic—such as detecting flashcards, spawning corresponding 3D objects, and controlling their appearance—was programmed in C#. The scripting language provided flexibility and precise control over how the application responds to user actions and camera input.
- **3D Assets and Models: Unity Asset Store / Online Resources**
  The 3D images used in this project were sourced from online repositories and Unity’s Asset Store to save development time. However, for those seeking full creative control or wishing to design custom visuals, tools like Blender or Maya can be used to create original 3D models.

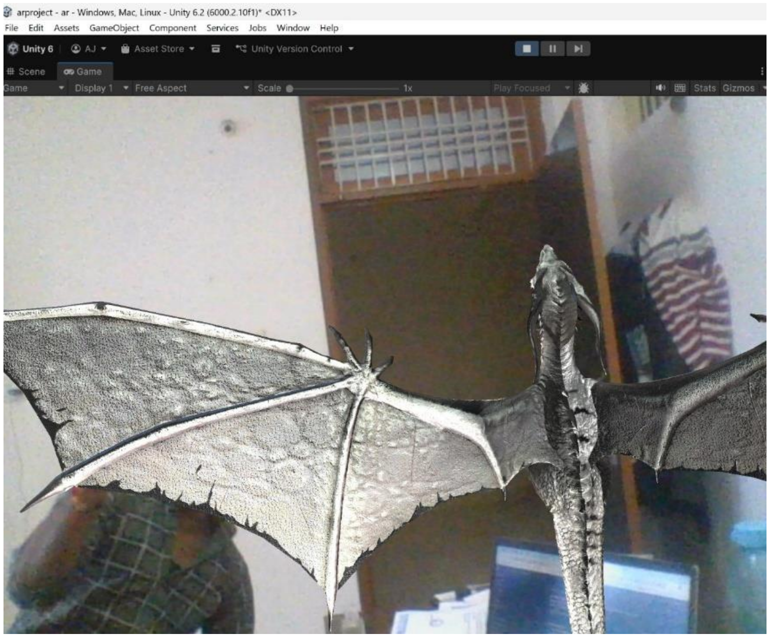
3. Screenshots or Visual Output

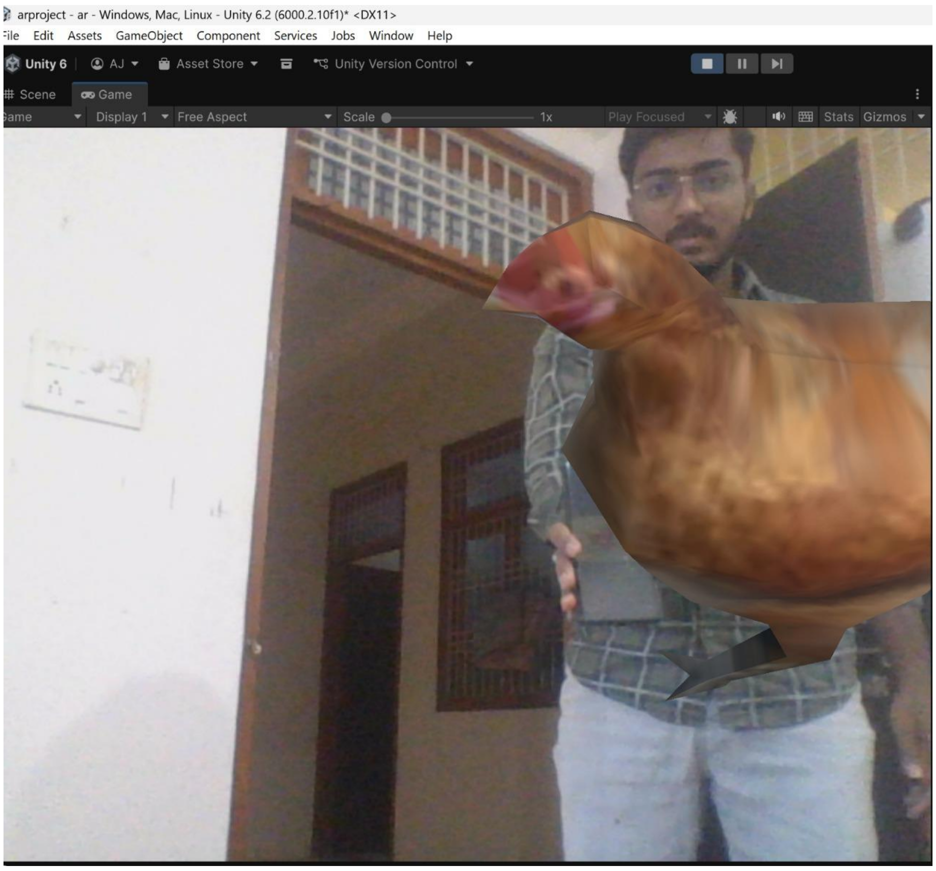
The visual output of the AR Flashcards for Language Learning project focuses on displaying 3D objects that appear when specific flashcards are detected through the Unity Game window. Rather than a mobile APK, the testing and demonstration were carried out directly within the Unity 6.2 editor.

- **AR Object Placement:**
  When a flashcard image (for example, a dragon or a chicken) is recognized by the AR camera, the corresponding 3D model appears anchored in the user's real-world environment as seen through the webcam feed.
- **Real-Time Scene Visualization:**
  The 3D objects blend naturally with the live camera background, giving the illusion that they exist in the user's physical surroundings. For instance, the dragon model appears as if it is standing on a desk or flying above the workspace, visible from different viewing angles.
- **Model Quality and Detailing:**
  The models used in the project (such as the dragon and chicken) are taken from online 3D sources and are rich in surface detail, textures, and structure. These assets enhance the realism and visual appeal of the AR output.
- **Unity Game View Integration:**
  Since the project was executed within Unity, the Game window displays the live output, combining the webcam feed with overlaid 3D objects. This provides a clear preview of how the AR experience would appear in an actual mobile deployment.
- **Testing and Interaction:**
  Users can view multiple objects in the same scene, move the camera, or adjust angles to explore the AR visuals interactively. This demonstrates the basic functionality of marker-based AR in an educational context.

The examples below demonstrate how the AR system responds to different flashcard inputs by displaying the corresponding 3D objects.

JORDI GONZALEZ


arproject - ar - Windows, Mac, Linux - Unity 6.2 (6000.2.10f1)* <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
AJ
Asset Store
Unity Version Control
Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos


arproject - ar - Windows, Mac, Linux - Unity 6.2 (6000.2.10f1)* <DX11>
File Edit Assets GameObject Component Services Jobs Window Help
Unity 6
AJ
Asset Store
Unity Version Control
Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Stats
Gizmos

4. Conclusion

The “AR Flashcards for Language Learning” project successfully demonstrates how Augmented Reality can be used to make the learning process more interactive and engaging. The project met its core objective by enabling 3D image visualization upon scanning specific flashcards, thus combining physical learning tools with digital content in an innovative way.

By utilizing Unity 3D (version 6.2) along with ARCore/ARKit, the project effectively anchored 3D models in real-world environments, providing a seamless AR experience directly within the Unity Game view. The integration of C# scripting allowed precise control over object behaviour and camera interaction, ensuring smooth performance during testing.

Overall, this project serves as a practical demonstration of AR’s potential in education—especially for beginners and language learners. In the future, this concept can be extended by incorporating additional features such as audio pronunciations, animated 3D models, or text overlays to create a richer and more comprehensive language learning experience.

VR Project Title : VR Shooting Game

1. Objectives and Project Description

Project Description:

The “VR Shooting Game” project is an attempt to create an immersive Virtual Reality-based shooting experience using the Unity 3D game engine. The main idea behind the project is to provide users with a first-person VR environment where they can aim, shoot, and interact with virtual objects using motion-based controls.

The project focuses on exploring the fundamentals of VR interaction such as hand tracking, object manipulation, and spatial awareness rather than creating a full-scale game. Players are expected to use VR controllers to shoot at targets or cubes placed in the 3D environment, helping them understand the core concepts of physics and interactivity within a virtual space.

The project serves as a foundation for understanding how virtual environments are designed, how user actions are mapped in 3D space, and how Unity integrates VR components using the XR framework.

Key Objectives

The main objectives of the VR Shooting Game are:

- **Immersive VR Gameplay:**
  To provide an interactive virtual environment where players can aim and shoot at 3D targets using VR motion controls.
- **Core VR Interaction Setup:**
  To understand and implement VR essentials like headset tracking, hand or controller input, and interaction with 3D objects in Unity.
- **Physics-Based Shooting Mechanics:**
  To simulate realistic shooting behaviour, where bullets or rays interact with virtual targets and respond accordingly (e.g., destruction or disappearance).
- **Basic Environment Design:**
  To create a simple but engaging 3D arena that allows the player to freely move, aim, and interact within the VR space.

- **Learning Virtual Reality Frameworks:**
  To gain hands-on experience with Unity's XR Interaction Toolkit and OpenXR, which are key technologies for VR application development.

2. Development Tools Used

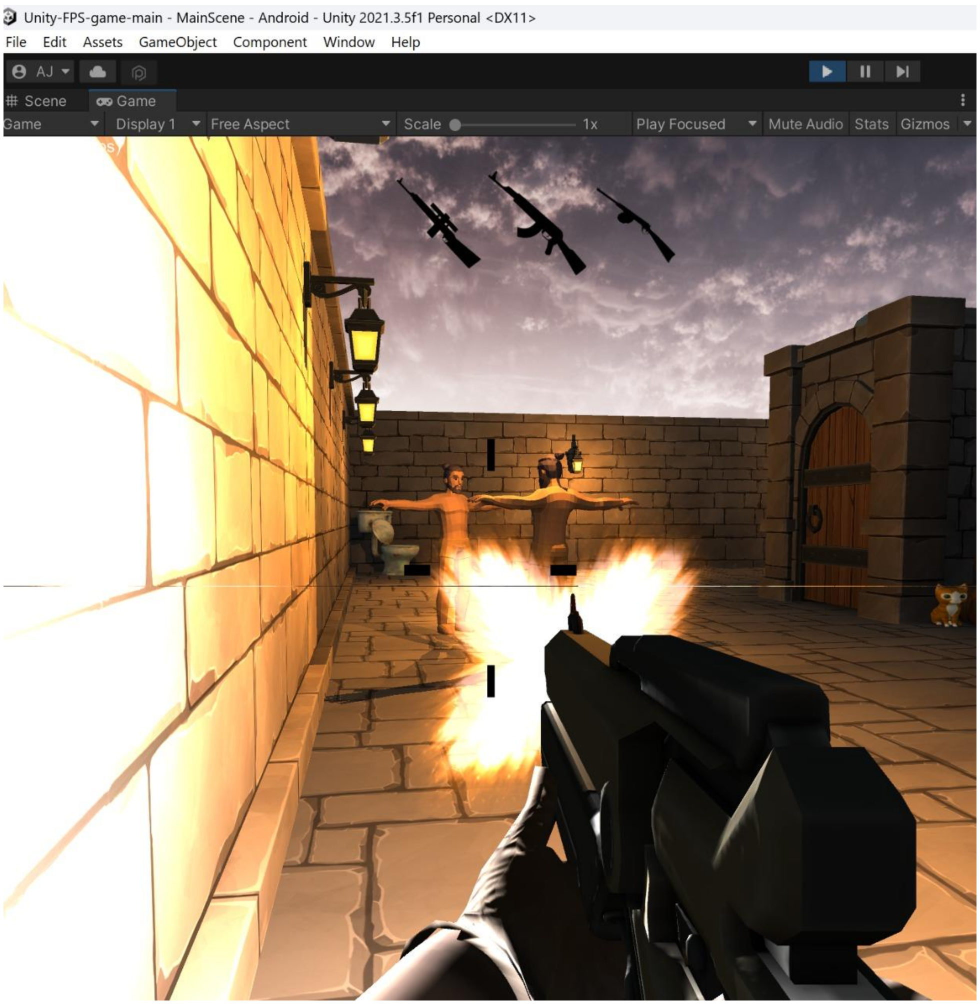
The development of the VR Shooting Game involved the use of modern, industry-standard tools to create an immersive and interactive virtual experience.

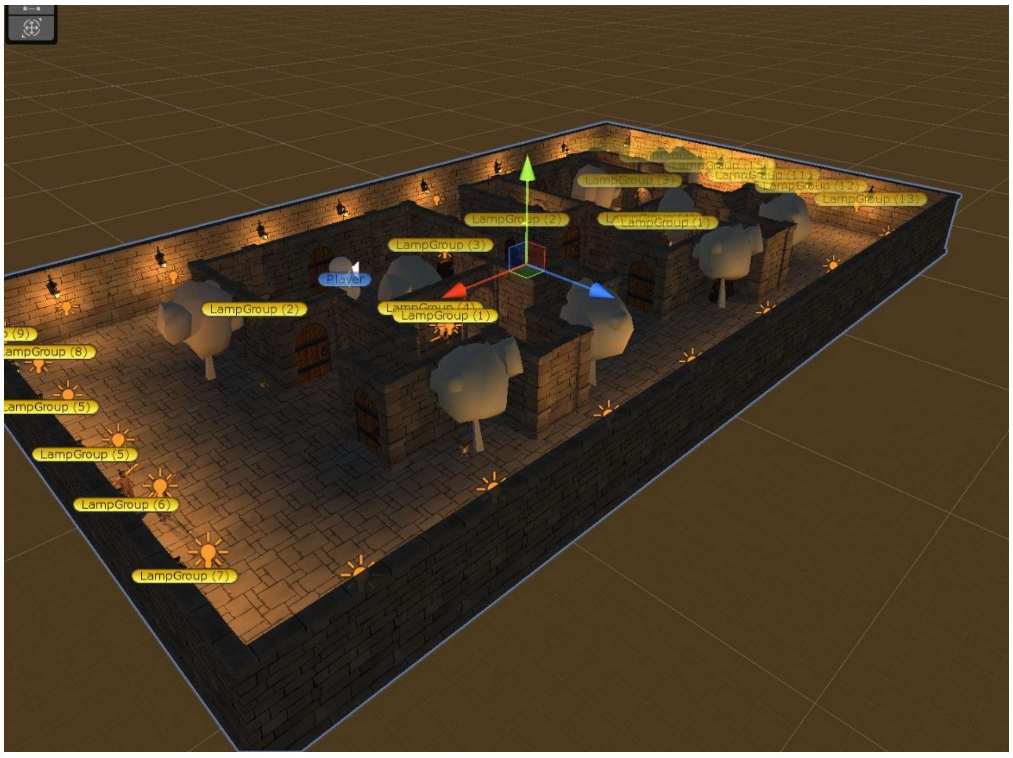
- **Game Engine: Unity 3D (Version 2021.3.5f1)**
  Unity served as the main development environment for building the VR game. It was used to design the 3D environment, manage lighting, handle physics, and integrate VR functionalities through the XR Interaction Toolkit. Unity's flexibility and compatibility with multiple VR platforms made it ideal for developing an interactive first-person experience.
- **Primary Scripting Language: C# (C-Sharp)**
  All gameplay logic — such as handling player inputs, detecting shooting actions, managing object interactions, and controlling basic VR mechanics — was implemented using C#. This allowed fine-tuned control over player behaviour and real-time responses within the virtual environment.
- **3D Models and Assets: Unity Asset Store / Online Resources**
  The 3D models and props used in the environment were primarily sourced from Unity's Asset Store and other free online repositories. This ensured efficient development without the need for custom modelling. However, tools like Blender can be used to create custom assets for those seeking complete creative control or additional realism.
- **VR Framework and SDKs: OpenXR and XR Interaction Toolkit**
  The VR experience was built using Unity's XR Interaction Toolkit integrated with the OpenXR plugin. These frameworks handled key functionalities such as headset tracking, controller input mapping, and object interaction, allowing for smooth and responsive gameplay in a VR setting.

3. Screenshots or Visual Output

The project emphasizes an immersive first-person experience that combines responsive weapon interactions with visually engaging environments. The perspective keeps the player closely tied to the action, ensuring every movement and reaction feels immediate and impactful.

- **Weapon Representation and Animation**
  The player's weapons are rendered in detailed 3D models that occupy the foreground view. Each firearm includes subtle animations — such as hand sway when moving, fluid aim transitions, and seamless reload sequences contributing to a believable sense of motion and weight.
- **Lighting and Environmental Aesthetics**
  The world is built with realistic lighting that accentuates depth and material response. Dynamic shadows, warm lamp glows, and reflective surfaces all work together to create a grounded and atmospheric setting. Each area is illuminated thoughtfully to balance gameplay visibility and mood, enhancing the realism of the scene.
- **Interface and Player Feedback**
  A minimalist HUD ensures the player's focus remains on the action. It typically displays only the most relevant information — such as ammo count, weapon details, and a functional crosshair — keeping the screen uncluttered and maintaining immersion.
- **Impact and Environmental Reactions**
  Combat visuals are enhanced through dynamic surface reactions that differ depending on the material being hit. When bullets strike wooden objects, they produce splintered fragments and dust, creating a rough, natural break effect. Impacts on metal surfaces result in bright sparks, shallow dents, and distinctive ricochet flashes that match metallic sound cues. Hitting stone or concrete surfaces generates light debris, small craters, and bursts of grey dust, emphasizing the physical texture of the environment.


Unity-FPS-game-main - MainScene - Android - Unity 2021.3.5f1 Personal <DX11>
File Edit Assets GameObject Component Window Help
AJ
Scene
Game
Game
Display 1
Free Aspect
Scale
1x
Play Focused
Mute Audio
Stats
Gizmos


LampGroup (2)
LampGroup (3)
LampGroup (1)
LampGroup (11)
LampGroup (12)
LampGroup (13)
Player
LampGroup (2)
LampGroup (4)
LampGroup (1)
LampGroup (8)
LampGroup (5)
LampGroup (5)
LampGroup (6)
LampGroup (7)

4. Code

The following are some C# code snippets that demonstrate the implementation logic discussed in this section.

Bullet Script:

```
using UnityEngine;


public class BulletScript : MonoBehaviour
{
    [Tooltip("Furthest distance bullet will look for target")]
    public float maxDistance = 1000000;
    RaycastHit hit;
    public GameObject decalHitWall;
    public GameObject bloodEffect;
            public GameObject woodEffect;
    public GameObject BarrelEffect;
            public GameObject stoneEffect;
            public GameObject metalEffect;
    public LayerMask ignoreLayer;


    /*
    * Upon bullet creation with this script attached,
    * the bullet creates a raycast which searches for colliders with corresponding materials.
    * If the raycast hits something with a matching material, it will create the corresponding effect.
    */
    void Update()
    {
```

```
    if (Physics.Raycast(transform.position, transform.forward, out hit, maxDistance,
~ignoreLayer))
    {
      ObjectHealth objectHealth = hit.collider.GetComponent<ObjectHealth>();

      if (objectHealth != null)
      {
        MaterialType materialType = objectHealth.materialType;

        switch (materialType)
        {
          case MaterialType.Wood:
            SpawnDecal(hit, woodEffect);
            break;
          case MaterialType.Metal:
            SpawnDecal(hit, metalEffect);
            break;
          case MaterialType.Barrel:
            SpawnDecal(hit, BarrelEffect);
            break;
          case MaterialType.Skin:
            SpawnDecal(hit, bloodEffect);
            break;
          case MaterialType.Stone:
            SpawnDecal(hit, stoneEffect);
            break;
                                  case MaterialType.Wall:
            SpawnDecal(hit, decalHitWall);
```

```
                    break;
                default:
                    break;
            }
        }

        Destroy(gameObject);
    }

    Destroy(gameObject, 0.1f);
}

void SpawnDecal(RaycastHit hit, GameObject prefab)
{
    GameObject spawnedDecal = Instantiate(prefab, hit.point + hit.normal * 0.01f, 
Quaternion.LookRotation(hit.normal));
    spawnedDecal.transform.SetParent(hit.collider.transform);
}
}
```

Object Health Script:

```
using UnityEngine;

public enum MaterialType
{
    Wood,
    Metal,
    Barrel,
    Skin,
    Stone,
    Wall
}

public class ObjectHealth : MonoBehaviour
{
    public int maxHealth = 100;
    public int currentHealth;
    public MaterialType materialType;
    public GameObject smallExplosionEffect;

    public AudioSource woodHitSound;
    public AudioSource metalHitSound;
    public AudioSource characterHitSound;
    public AudioSource destructionSound;

    private void Start()
    {
        currentHealth = maxHealth;
    }

    public void TakeDamage(int damage)
    {
        currentHealth -= damage;

        // Play appropriate hit sound based on material type
        switch (materialType)
        {
```

```
        case MaterialType.Wood:
            if (woodHitSound != null)
            {
                woodHitSound.Play();
            }
            break;
        case MaterialType.Metal:
            if (metalHitSound != null)
            {
                metalHitSound.Play();
            }
            break;
        case MaterialType.Skin:
            if (characterHitSound != null)
            {
                characterHitSound.Play();
            }
            break;
        default:
            break;
    }

    if (currentHealth <= 0)
    {
        DestroyObject();
    }
}
```

```
    private void DestroyObject()
    {
        // Play destruction sound
        if (destructionSound != null)
        {
            destructionSound.Play();
        }

        // Spawn small explosion effect
        if (smallExplosionEffect != null)
        {
            Instantiate(smallExplosionEffect, transform.position, transform.rotation);
        }

        // Handle object destruction here
        Destroy(gameObject);
    }
}
```

5. Conclusion

The VR Shooting Game project represents an engaging step toward understanding virtual reality-based interactive experiences. Through Unity 2021.3.5f1 and C# scripting, we explored the process of building a basic VR environment where the player can interact and shoot at virtual targets.

The foundational components such as player movement, weapon handling, and target interaction demonstrate the core idea of an immersive VR gaming experience. This project highlights the potential of Unity’s physics and interaction systems in creating realistic virtual environments.

In the future, the game can be expanded with features like score tracking, multiple weapon types, sound effects, and improved visual feedback to enhance realism and engagement. Overall, this work serves as a valuable introduction to VR game development and the integration of user interaction within virtual spaces.