Report On

Bottle Shooter Game using VR

Submitted in partial fulfillment of the requirements of the Course project in Semester VII of Fourth Year Computer Engineering

By

Vipul Bhoir(Roll No.07)

Mrudul Chaudhari(Roll No. 12)

Abhinav Desai(Roll No. 14)

Supervisor

Mr. Vikrant Agaskar

University of Mumbai

Vidyavardhini's College of Engineering & Technology

Department of Computer Engineering

(2024-25)

Vidyavardhini's College of Engineering & Technology
Department of Computer Engineering

CERTIFICATE

This is to certify that the project entitled **"Bottle Shooter Game using VR"** is a bonafide work of **"Vipul Bhoir(Roll No.07), Mrudul Chaudhari(Roll No. 12), Abhinav Desai(Roll No. 14)"** submitted to the University of Mumbai in partial fulfillment of the requirement for the Course project in semester VII of Fourth Year Computer Engineering.

Supervisor
Mr. Vikrant Agaskar

Dr. Megha Trivedi
Head of Department

Abstract

The "Bottle Shooter Game Using VR" is an immersive virtual reality experience that blends engaging gameplay with the innovative capabilities of VR technology. This project aims to develop a highly interactive and visually captivating game where players can test their shooting accuracy and reflexes by aiming and shooting at bottles in a virtual environment. Leveraging motion tracking, 3D environments, and realistic physics, the game offers an authentic shooting experience. The project demonstrates the potential of virtual reality for creating engaging entertainment while pushing the boundaries of immersive gaming. By integrating VR technology, the "Bottle Shooter Game" offers a unique experience that blends fun with skill development in a virtual space.

1 Introduction

1.1 Introduction

In this game, players step into a virtual world where they aim to shoot bottles using a simulated weapon, testing their precision, timing, and reflexes. Virtual reality technology has revolutionized the gaming industry by providing a sense of presence and immersion, allowing users to interact with a 3D environment as if they were physically there.
This game takes advantage of VR's unique features, such as motion tracking and 360-degreeenvironments, to create an engaging, lifelike experience. Players use VR controllers to simulate the handling of a gun, targeting bottles at varying distances and angles, all while being surrounded by realistic visuals and sound effects. The game's objective is to challengethe player's accuracy and improve their hand-eye coordination in an enjoyable and stimulating way.

1.2 Problem Statement

Traditional 2D shooting games, while entertaining, lack the immersive experience that fully engages players in the gaming environment. As gaming technology evolves, players are seeking more interactive and realistic experiences that offer a sense of presence and active participation. The challenge is to create a gaming experience where players can physically interact with the environment, enhancing their precision, reflexes, and engagement in real- time.

1.3 Project Scope

Develop a highly engaging and immersive virtual reality (VR) environment where players canshoot bottles in a realistic 3D setting. The game will simulate real-world physics, providing players with a lifelike shooting experience. The project will ensure smooth performance on VR platforms with efficient rendering, reducing motion sickness, and providing a comfortable experience for players over long periods.

Requirement Analysis

2.1 Software Requirements:

The system will require the following software :

- Unity 3D or Unreal engine
- C#
- VS Code

2.2 Hardware Requirements

The system will require the following hardware:

A computer with at least 4GB of RAM and 100GB of free disk space
An internet connection

Recommended:

- 16 GB RAM

Minimum:

- 8 GB RAM

2.3 Functional Requirements

- The system must be able to perform the following functions:
- The system should allow users to log in and create profiles to save their progress,high scores, and game settings.
- The system must allow users to aim at and shoot bottles in a 3D environmentusing the VR controllers.
- Once hit, bottles should break with realistic physics and sound effects.
- The game should simulate real-world physics for projectile shooting and bottlebreaking

2.4 Nonfunctional Requirements

Performance: The game should maintain a stable frame rate of at least 90 FPS toensure smooth VR experiences and avoid motion sickness.The game should load quickly, with minimal lag during gameplay, even as complexity increases (more objects or higher difficulty levels).

Scalability: The system should be scalable to support additional levels, features, orVR devices without significant redevelopment.

Maintainability: The codebase should be modular, well-documented, and easy tomaintain for future updates or feature additions.

Reliability: The system should be stable and handle crashes or glitches effectively.It should not lose player progress or settings in case of an error.

Compatibility: The game should be compatible with various VR devices (Oculus Rift, HTC Vive, etc.) and controllers.It should run smoothly across different hardware configurations, ensuring the minimum system requirements are met.

3. System Design

3.1 System Design:

The system design for a VR-based "Bottle Shooter Game" can be broken down into several key components and subsystems. These components work together to ensure that the game is immersive, responsive, and engaging for the user.

The following are the main modules of the system:

Game Engine: Handles real-time physics for bottle movement, gunfire trajectories, and the breaking of bottles when hit. This ensures realistic interactions with the virtual environment.

User VR interaction: This interaction is processed and interpreted by the VR integration layer.

VR integration Layer: This layer connects the hardware (with the game engine, ensuring hand movements are correctly interpreted in the virtual environment.

Testing and performance optimization: Use optimizations in game code and rendering to ensure that user actions (e.g., shooting) are reflected in real-time with minimal delay.

3.2 Module Description:

User interface module: This module is responsible for designing and rendering the game's interface, including the menus, HUD (heads-up display), scoreboards, and interaction buttons..

VR environment and control module: This module manages the virtual world, creating an immersive experience for the player. It defines the player's interactions within the VR world, including shooting mechanics, bottle physics, and object interactions.

Game physics and mechanic module: This module defines the physics engine of the game, including gravity, collision detection, and bottle dynamics when hit by the virtual bullets.

Audio and Visual Effects module: This module handles all sound and visual effects in the game, including background music, bottle shattering sounds, bullet firing effects, and visual feedback when a bottle is hit.

4. Implementation

4.1 Methodology

1. Requirement analysis

Gather all requirements for the game, including hardware specifications (VR headsets,controllers), software frameworks, and user expectations.

- Identify key gameplay mechanics: shooting bottles, level progression, scoring.
- Define software tools: Unity3D, Unreal Engine.

2. Game Development

Implement the core functionality of the game, including VR controls, bottle shooting mechanics, and scoring system. Set up the VR environment and integratethe controls for tracking user movements and shooting actions. Integrate audio cuesfor bottle shattering, shooting, and background music.

3. Testing and debugging

Ensure the game runs smoothly across different VR devices, with minimal bugs and an immersive experience. Conduct gameplay testing with different users to ensure the game isengaging, intuitive, and bug-free.

4.2 Sample Modules

1. User Interface Module

This module handles the creation of the virtual environment and user interface. It includes menus for starting the game, selecting levels, and displaying scores. The module ensures seamless interaction between the user and the VR interface.

2. VR control module

Handles input from VR devices like hand controllers or motion trackers. It captures user gestures and movements, translating them into in-game actions like aiming, shooting, and interacting with objects.

3. Scoring and Feedback Module

This module tracks the player's performance, including hit accuracy, speed, and level completion. It provides real-time feedback to the player and manages the scoring system, offering rewards or progressing the player to higher difficulty levels.

4. Sound and Visual effects module

Responsible for integrating sound effects like gunfire, bottle breaking, and background music. This module also handles visual effects such as explosions or slow-motion effects when the player hits a bottle, adding to the overall immersive experience.

4.3 Code

```
using System.Collections;
using System.Collections.Generic;using UnityEngine;

public class BottleSpawner : MonoBehaviour
{

public GameObject bottlePrefab;bool bottleOnTheWay;

void Start()
{
SpawnNewBottle();
}

public void InitializeNewBottle()
{
//Check if spawn is already initializedif (bottleOnTheWay)
return;

//Spawn bottle in 4 seconds
```

```
bottleOnTheWay = true; Invoke("SpawnNewBottle", 4);
}

private void SpawnNewBottle()
{
        GameObject myBottle = Instantiate(bottlePrefab, transform.position,
Quaternion.identity);
bottleOnTheWay = false;

myBottle.GetComponent<Bottle>().mySpawner = this;
}

}

using System.Collections;
using System.Collections.Generic;using UnityEngine;
using UnityEngine.UI;
using UnityEngine.SceneManagement;using Boo.Lang;

public class GameManager : MonoBehaviour {

public int score; public int highscore;
public float timer = 20f;
```

```
public bool gameStarted;private Hud hud;

private void Awake()
{
hud = FindObjectOfType<Hud>();
}

private void Start()
{
this.UpdateHighscore();
}

private void Update()
{
if (gameStarted)
{
//Minus 1 per second
timer -= 1 * Time.deltaTime;
//Without decimal hud.SetTimerText(timer.ToString("F0"));

if(timer <= 0)
{
//Game ended
```

```
Bottle[] bottles = FindObjectsOfType<Bottle>();foreach(Bottle bottle in
bottles)
{
Destroy(bottle.gameObject);
}

BottleSpawner[] spawners = FindObjectsOfType<BottleSpawner>();foreach
(BottleSpawner spawner in spawners)
{
spawner.CancelInvoke();
}
gameStarted = false;

Invoke("RestartGame", 8);
}
}
}

public void IncreaseScore()
{
score++; hud.SetScoreText(score);

if (score > highscore){ PlayerPrefs.SetInt("Highscore", score);
    this.UpdateHighscore();
}
```

```
}

private void RestartGame()
{
SceneManager.LoadScene(0);
}

private void UpdateHighscore()
{
highscore = PlayerPrefs.GetInt("Highscore");hud.SetHighscoreText(highscore);
}
}

using System.Collections;
using System.Collections.Generic;using UnityEngine;
using UnityEngine.UI;

public class Hud : MonoBehaviour
{
public Text timerText; public Text scoreText; public Text highscoreText;

public void SetScoreText(int value)
{
```

```
scoreText.text = "Treffer: " + value;
}

public void SetHighscoreText(int value)
{
highscoreText.text = "Highscore: " + value;
}

public void SetTimerText(string value)
{
timerText.text = value + " Sekunden";
}
}

using System.Collections;
using System.Collections.Generic;using UnityEngine;

public class Revolver : MonoBehaviour {public Transform spawnPoint;

private SteamVR_TrackedObject trackedObj;private
SteamVR_Controller.Device device;

public GameObject flashEffectPrefab;
```

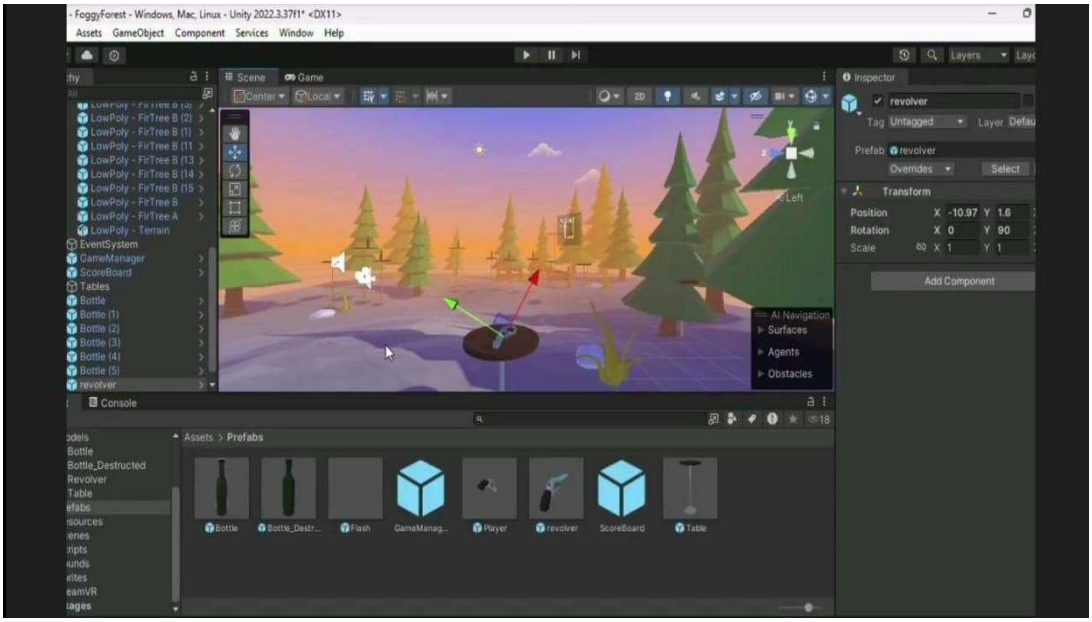
```
AudioSource audioSource;


private void Start()
{
trackedObj = GetComponentInParent<SteamVR_TrackedObject>();
audioSource = GetComponent<AudioSource>();
}

void Update () {

device = SteamVR_Controller.Input((int)trackedObj.index);

//Visible Raycast for Debug
        Debug.DrawRay(spawnPoint.transform.position,
spawnPoint.transform.forward,Color.red);


//Raycast to check what we hitRaycastHit hit;

        if
(Physics.Raycast(spawnPoint.transform.position,spawnPoint.transform.forward,
out hit, Mathf.Infinity))
{
if(hit.transform.tag == "Target")
{
       if (device.GetPressDown(SteamVR_Controller.ButtonMask.Trigger))
```

```
{
hit.transform.GetComponent<Bottle>().GotShot(hit.point);
}
}
}

DoOnShoot();


}

private void DoOnShoot()
{
if (device.GetPressDown(SteamVR_Controller.ButtonMask.Trigger))
{
device.TriggerHapticPulse(60000);
audioSource.PlayOneShot(audioSource.clip); GameObject effect =
Instantiate(flashEffectPrefab,
spawnPoint.transform.position, Quaternion.identity); effect.transform.parent =
        spawnPoint.transform;Destroy(effect, .2f);
}
}
}

using System.Collections;
using System.Collections.Generic;using UnityEngine;
```

```
public class Teleport : MonoBehaviour
{
private SteamVR_TrackedObject trackedObj;private
SteamVR_Controller.Device device;

private LineRenderer lineRenderer;

private void Awake()
{
lineRenderer = GetComponent<LineRenderer>(); trackedObj =
GetComponent<SteamVR_TrackedObject>();
}

private void Update()
{
device = SteamVR_Controller.Input((int)trackedObj.index);
lineRenderer.SetPosition(0, transform.position);

RaycastHit hit;
if(Physics.Raycast(transform.position, transform.forward, out hit,
Mathf.Infinity))
{
if(hit.transform.tag == "Teleport")
{
lineRenderer.SetPosition(1, hit.point);
if (device.GetPressDown(SteamVR_Controller.ButtonMask.Trigger))
{
```

```
// Move player to pointed position transform.parent.position = hit.point;
}
}
} else
{
lineRenderer.SetPosition(1, transform.position);
}
}



}
```

4.4 Output:

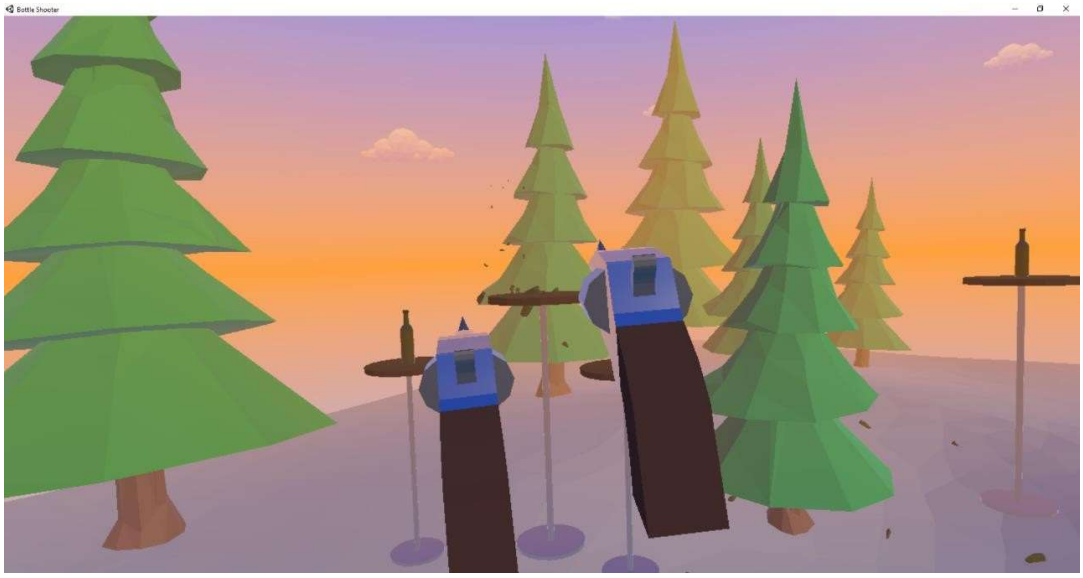


4.4.1 Interface before shooting the bottles

4.4.2 Interface of background environment with Bottles

4.4.3 Interface Showing the Bottles shotted

4.4.4 Interface with Guns dor shooting



4.4.5 Interface when bootles are been shoot

4.4.6 Final Interface

5. Results:

5.1 Results:

The **Bottle Shooter Game using VR** successfully integrates immersive virtual reality technology with engaging gameplay mechanics. The game provides a realistic environment whereusers can interact with 3D objects, aiming and shooting at bottles with high accuracy using VR controllers. Testing across multiple VR platforms showed strong compatibility and responsiveness, creating an engaging, intuitive user experience. The physics engine accurately simulated real-world bottle-breaking and projectile mechanics. Performance optimization ensured smooth gameplay, minimizing motion sickness. User feedback praised the immersive interaction, sound effects, and visual elements, with potential for further updates and enhancements.

5.2 Conclusion:

The **Bottle Shooter Game using VR** demonstrates the power of virtual realityin creating highly immersive and interactive gaming experiences. By simulating real-world physics and offering intuitive controls, the game successfully engages users in a fun and challenging environment. It highlights the potential of VR technology in enhancing user immersion through precise motion trackingand lifelike 3D environments. The game has proven to be both enjoyable and accessible, providing a solid foundation for future expansions. As VR continues to evolve, this project underscores its applicability in gaming and entertainment.

References:

[1] Corcos A. Being enjoyably challenged is the key to an enjoyable gaming experience: an experimental approach in a first-person shooter game. Socioaffect Neurosci Psychol. 2018, 15;8(1):1474668.
[2] I. Makarov, M. Tokmakov, and L. Tokmakova. Imitation of human behavior in 3D-shooter game. In 4th International Conference on Analysis of Images, Social Networks and Texts, 2015,64-77.
[3] Wang D. Tan A. H.. Creating autonomous adaptive agents in a real-time first-person shooter computer game. IEEE Transactions on Computational Intelligence and AI in Games, 2015, 7(2):.123-138.
[4] Jansz, Jeroen, Tanis, Martin. Appeal of Playing Online First Person Shooter Games. Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society, 2020, 10:133- 136.