

Virtual Reality Fitness Game: Unleashing Fun and Exercise Fusion

Virtual Reality Fitness Game: Unleashing Fun and Exercise Fusion

Nebil Gokdemir

**CALIFORNIA STATE UNIVERSITY
FULLERTON**

Table of Contents

Introduction

1.1 Background

1.2 Objectives

1.3 Scope and Limitations

Abstract

2.1 Overview

2.2 Problem Definition

2.3 Proposed Solution

Problem Statement

3.1 Common Exercise Challenges

3.2 Need for an Engaging Workout Solution

User-Friendly Project Overview

4.1 Ease of Use

4.2 Benefits

Significance

5.1 Addressing Workout Boredom

5.2 Motivation and Sustained Engagement

Future Enhancements

6.1 Expansion of VR Fitness

6.2 Integration with Wearable Tech

Development Environment

7.1 Unity Platform

7.2 Version Control

7.3 System Architecture

Implementation

8.1 Oculus Integration

8.2 Scene Design

8.3 First and Second Level

Challenges and Solutions

Learning

10.1 C# Programming

10.2 Unity Game Development

Introduction

Virtual reality (VR) technology has gained popularity in recent years for its ability to create an immersive and interactive environment that mimics real-life situations. This technology has been widely used in the entertainment and gaming industries. Additionally, VR has the potential to provide users with health benefits by incorporating physical activity into the gameplay. In this project, the goal is to help users combine entertainment with physical activity to create an engaging and motivating workout experience. The game will allow users to use VR controllers resembling swords to punch and destroy objects that come towards them. This incorporation of physical movements into gameplay provides an opportunity for users to improve their fitness and muscle strength while having fun. The game has three scenes. The first one is a menu with options to play, adjust music volume, and quit. The second scene is the first level, and the third scene is the second level. Each level features different objects (enemies) and varying difficulty levels to destroy them. This game is developed with Unity, and the majority of assets were obtained from the Unity Store. The VR project aims to provide significant health benefits to users and make exercise more convenient and enjoyable everywhere.

Abstract

This project aims to provide an immersive and interactive environment for users. The user can use VR controllers resembling sword to punch and destroy objects that come towards them. The physical movements required in gameplay will help improve the user's fitness and muscle strength. To develop this game, I used Unity and 3D assets from the unity asset store. The game starts with a menu, and the menu has play, option (volume), and quit buttons. In first and second level, there are score and max score label. If score pass max score, score switch with max score, and save it. When player destroy 50

object, they enter second level which is different object and faster. Furthermore, while player destroy object, it play sword music in the game. The goal of this project is to create a for engaging and motivating workout experience for users. The convenience of VR technology allows users to exercise anywhere and fit exercise into their busy schedules. This project has the potential to provide significant health benefits to users while making exercise more enjoyable.

Problem Statement

People often find it hard to stick to regular exercise because it can be boring. Not everyone gets excited about lifting weights or running on a treadmill. Plus, with busy schedules, it's tough to make time for workouts, so they can void do workout for this reason. CNN journalist Melanie Radzicki mention in 6 creative way to transform your workouts article that “Maintaining motivation in long-term workout routines is a common challenge, as highlighted in a study by the National Institutes of Health, which found boredom to be a widespread emotion among athletes. Dr. Dan O'Neill, a sports psychologist and orthopedic surgeon, advises that continuous engagement of the body and mind involves incorporating variety—new challenges, ideas, and activities. The key to sustained motivation lies in embracing novelty and steering clear of monotony in workout routines.” Therefore, this project is perfect to help people make their work out variety. Due to is more fun, it will be more motivation.

Problem Definition

There's a need for a fun and engaging way to exercise that fits into people's daily lives. The usual workouts just don't click with everyone, and finding something enjoyable and easy to stick to is a challenge, especially with our busy routines such as playing game while do exercise.

Problem Rationale

Enter virtual reality (VR). I think using VR can make exercising way more interesting. Imagine playing a game where you punch and destroy stuff with VR controllers—it's like a workout, but it feels like you're in a video game. The goal is to make exercise something you look forward to, not something you force yourself to do. The idea is to use technology to turn exercise into a game, making it not only good for your health but also something you genuinely enjoy. If we can make working out fun and accessible, we might just change how people feel about exercise.

Proposed Solution

To make exercise more fun, this project suggests using virtual reality (VR) technology. We're creating a VR game with Unity where you can use VR controllers like swords to punch and destroy things in a virtual world.

Key Ideas

Fun Gameplay: The game will immerse you in a virtual world where you can punch and destroy things.

It's like a video game, but you're also working out!

Different Levels: There are three parts to the game—the menu, the first level, and the second level.

Each level has different challenges to keep things interesting.

Scores and Music: You'll earn points in the game, and when you reach a certain score, you move to the next level. Plus, there's cool sword music playing while you're destroying things.

Reasons for the Idea

We know that regular exercise can get boring. This VR game is all about making exercise exciting and something you look forward to. By turning it into a fun game, we hope it'll be more appealing and easier to stick to, even if you're super busy. The goal is to use technology to make exercising enjoyable and accessible to everyone.

User-Friendly Project Overview

Your project stands out for its user-friendly design, making it accessible and convenient for everyone.

Ease of Use: The project is designed with simplicity in mind, ensuring that users can navigate and interact with it easily. Whether it's setting up or using various features, the user experience is straightforward.

Offline Functionality: A notable feature is its ability to work without the need for an internet connection. This offline functionality ensures users can seamlessly use the project, providing convenience and accessibility even in areas with limited connectivity.

Unobtrusive Operation: The project operates unobtrusively, meaning it doesn't interfere with other tasks or activities on the user's device. This enhances the overall user experience, allowing individuals to integrate the project seamlessly into their routines.

Independence from Internet: The fact that the project doesn't require an internet connection is a significant advantage. Users can engage with it anytime, anywhere, without being dependent on online access. This feature caters to users in various settings, including those with limited connectivity options.

Intuitive Functionality: The intuitive nature of the project contributes to its user-friendliness. Users can easily understand how to interact with the project, minimizing the learning curve and ensuring a positive and efficient experience.

Benefits of this Project

Engaging Workout Experience: The VR game provides users with a unique and engaging workout experience by combining entertainment with physical activity. The immersive environment and interactive gameplay make exercise more enjoyable.

Improved Fitness and Muscle Strength: Incorporating physical movements into the gameplay, such as punching and destroying objects, offers users a chance to enhance their fitness and muscle strength while having fun.

Variety in Workouts: Addressing the common issue of workout boredom, the VR game introduces variety through different levels and challenges. This variety helps in maintaining motivation and avoiding monotony in exercise routines.

Convenience and Accessibility: With VR technology, users can exercise anywhere, eliminating the constraints of time and location. The project aims to make exercise more convenient, fitting into busy schedules without the need for a specific gym or location.

Significance: Addressing Workout Boredom: The project responds to the challenge of workout boredom, a widespread issue highlighted by studies. By incorporating novelty and variety through virtual reality, it offers a solution to the emotional aspect of boredom during exercise.

Motivation and Sustained Engagement: The game's goal is to provide a motivating workout experience, encouraging sustained engagement in physical activity. The incorporation of VR technology aims to make exercise something users genuinely look forward to.

Technology-Enhanced Fitness: By utilizing VR technology, the project aligns with the trend of integrating technology into fitness routines. This approach not only makes workouts more enjoyable but also reflects advancements in leveraging technology for health benefits.

Accessible Fitness for Everyone: The user-friendly design and offline functionality ensure accessibility for a wide range of users. The project aims to make exercise accessible to everyone, regardless of their location or internet connectivity.

Future

Expansion of VR Fitness: The success of this VR fitness game could pave the way for the development and expansion of similar projects. As technology evolves, we may see more innovative ways of integrating virtual reality into fitness routines.

Health and Wellness Applications: The concept of combining entertainment with physical activity could extend beyond gaming. Future applications might explore using VR for various health and wellness activities, catering to different preferences and fitness goals.

Integration with Wearable Tech: The project sets the stage for potential integration with wearable technologies. Future developments could explore syncing the VR game with fitness trackers or other wearables to provide users with comprehensive health data and insights.

Community and Social Interaction: Future iterations may focus on incorporating social features, allowing users to engage in virtual fitness communities or compete with friends. This social aspect could enhance motivation and create a sense of shared fitness goals.

Development Environment

Unity Development Platform:

Version Control: Employ the Git version control system to meticulously track changes and foster smooth collaboration during the development process.

Unity Hub: Efficiently manage diverse Unity versions and projects through Unity Hub, ensuring an organized and streamlined workflow for the development team.

Programming Language: C#: Harness the robust capabilities of C# as the primary programming language within the Unity development environment, empowering the creation of intricate and dynamic scripts.

Operating System: Edition Windows 11 Home, and 22H2 version.

Visual Studio: Seamlessly integrate Visual Studio with Unity, providing an enriched development experience with advanced features, including comprehensive code completion and robust debugging capabilities.

Graphics and Assets: Unity Asset Store: Access a diverse array of high-quality 3D models, textures, and assets from the Unity Asset Store, elevating the visual aesthetics and overall quality of the game.

Music: Used online source such as pixabay.com

Operational Environment

Operational Environment 8.0: Hardware Prowess: In crafting the virtual dimensions, a robust operational setting becomes paramount. Our hardware configuration, the backbone of this digital realm, embodies a harmonious synergy between performance and efficiency.

Central Processing Nexus - AMD Ryzen 5 5600H: At the core of this computational ballet lies the AMD Ryzen 5 5600H processor, a pinnacle of processing prowess. With its adept handling of computational choreography, it orchestrates the intricate dance of algorithms, rendering graphics with precision and finesse.

Memory Nexus - 64.0 GB RAM: The memory, akin to an expansive archive, stands tall with 64.0 GB RAM. Within this capacious vault, the nuances of our game find sanctuary. Data flows seamlessly, ensuring swift accessibility and retrieval, contributing to an immersive and uninterrupted gaming experience.

System Architecture

1. Unity Framework:

Role: Unity serves as the foundational framework overseeing the graphical rendering, user interactions, and overall game logic. It functions as the orchestrator ensuring a coherent and immersive virtual reality (VR) experience.

2. Core Modules:

User Interface (UI) Module: Manages the design and implementation of the UI, providing a seamless and intuitive navigation experience. Integrates UI elements for menu options, volume adjustment, first level and second level.

Interactions: Collaborates with other modules to trigger level transitions, score updates, and audio adjustments.

Gameplay Module: Implements fundamental gameplay mechanics, translating VR controller inputs into virtual sword movements. Collaborates with the level design module for dynamic object interactions and challenges.

Interactions: Communicates with the scoring module to update scores based on successful interactions with virtual objects. Triggers audio events through the audio module for immersive gameplay.

Level Design Module:

Defines the structure, challenges, and progression of each level within the game.

Collaborates with the gameplay module to orchestrate object spawning and difficulty adjustments.

Communicates with the gameplay module to synchronize virtual objects and challenges with user interactions. Triggers level transitions based on user performance and scores.

Audio Module: Manages the integration of audio elements, including sword music, to enhance the overall immersive experience. Allows users to adjust volume preferences through the UI module.

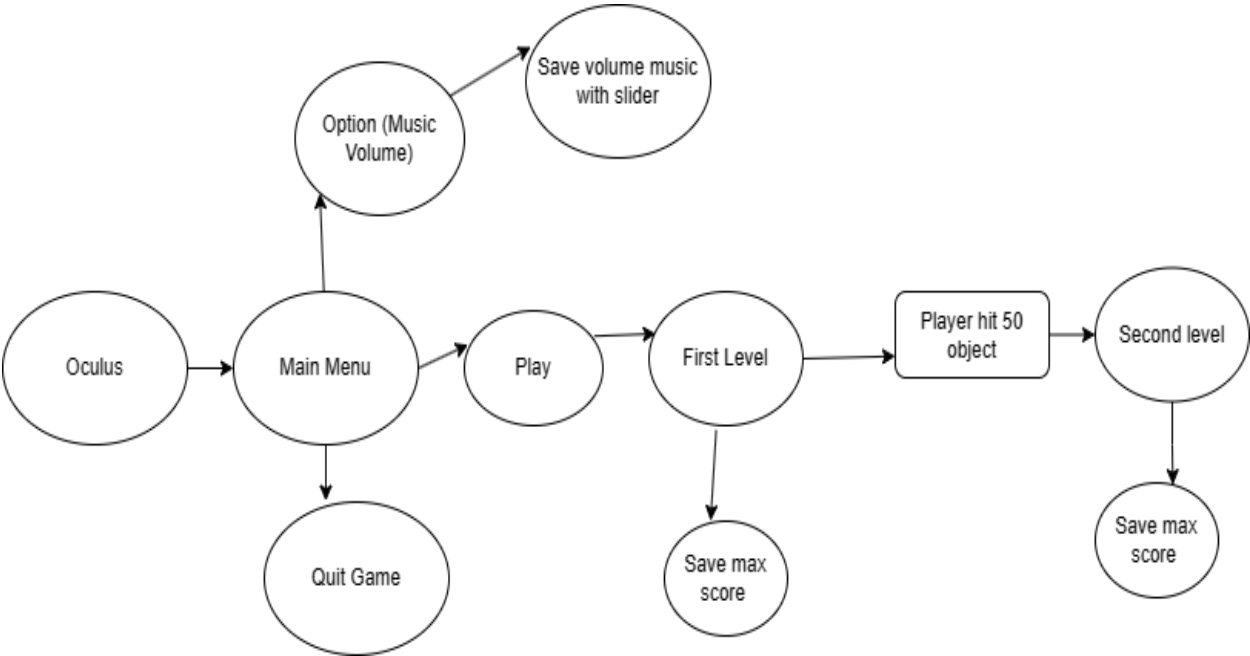
Scoring and Progression Module: Tracks user scores and determines level progression based on how many objects are hit. If more than 30 objects are hit, the difficulty increases, and if more than 50, it moves to the second level.

Manages the display of current and maximum achievable scores if the player's score gets more than the maximum score.

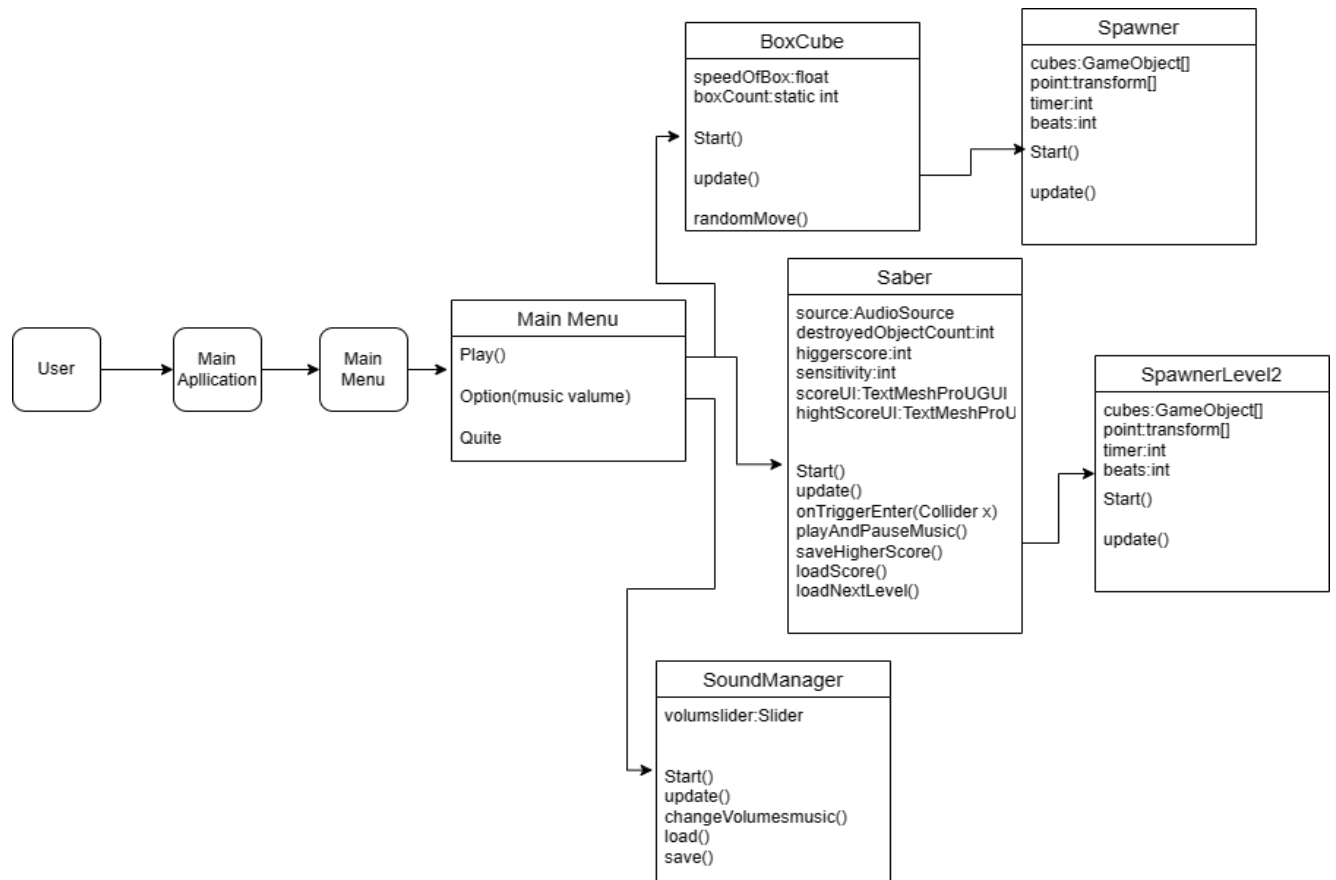
Persistence Module: Ensures the persistence of user progress by saving and loading relevant data.

Manages the storage and retrieval of user-specific information. When the score gets more than the maximum, it switches with the score and saves it.

Case Diagram



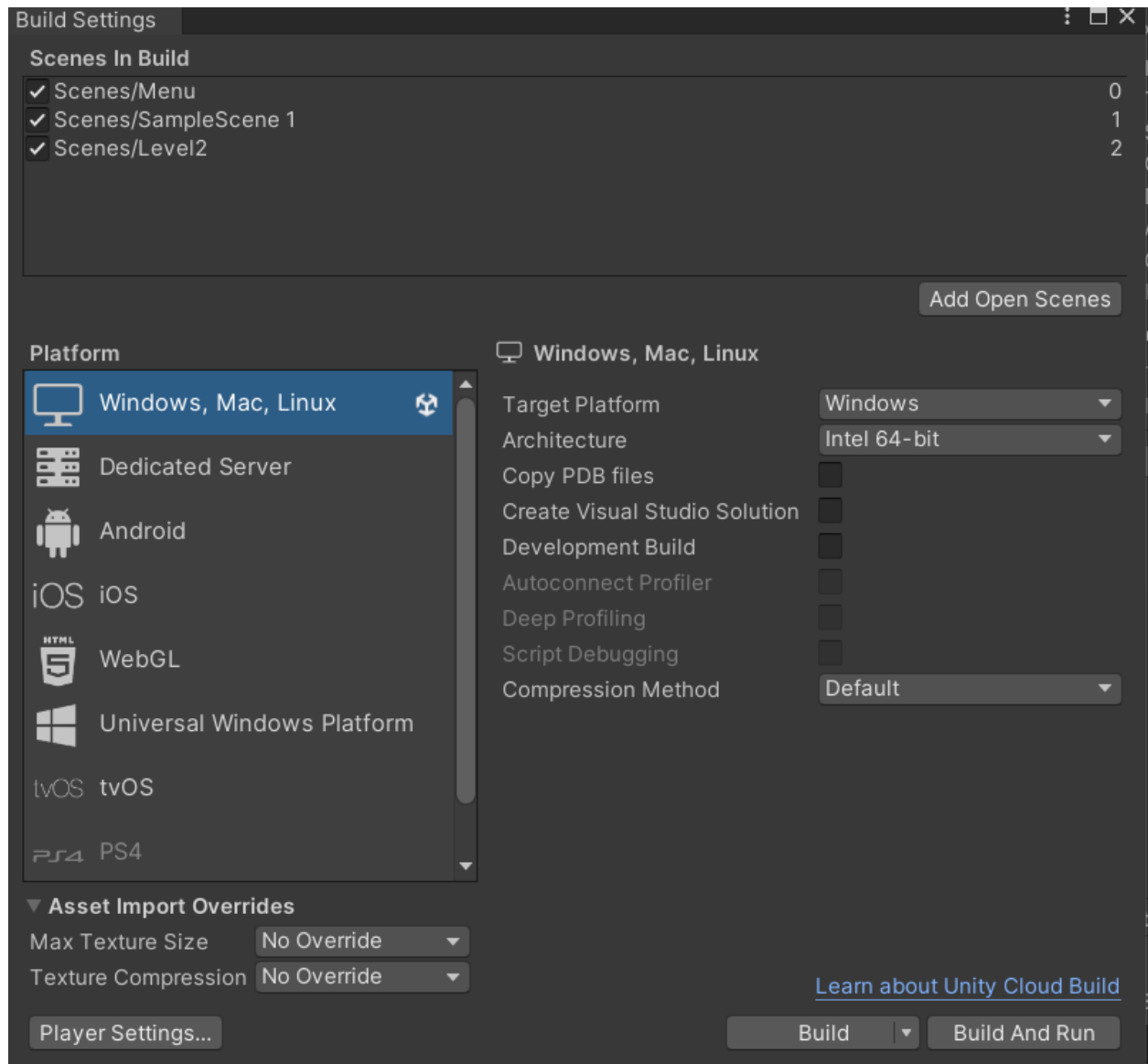
Class Diagram



Implementation

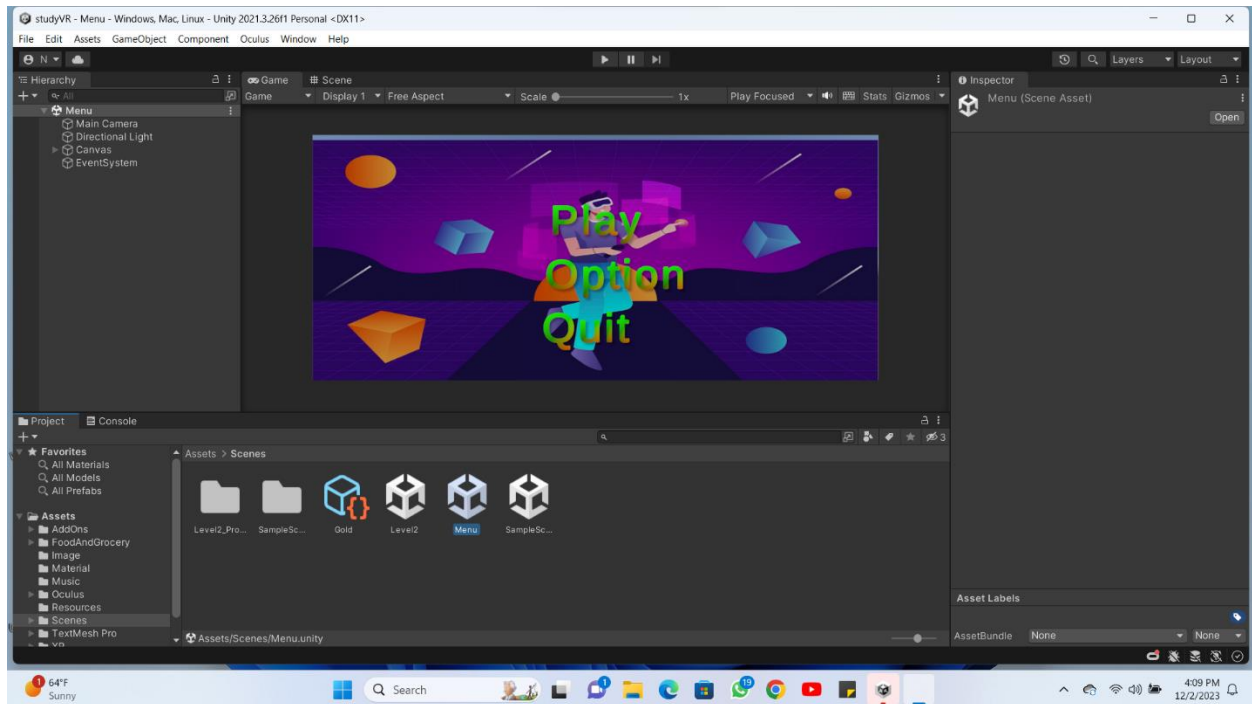
I have three scene in the queue. One is Menu and second one is first level, and last one is second level.

When game start, it start with menu. After 30 object spawned, object go up and down which get more difficult, and when player destroy 50 object, it take second level. As you can see in picture there are 3 scene in queue.



Menu Scene

The menu includes buttons designed with TextMeshPro, and it has three buttons. All button is functional now. When we click, Play button, it take us first level of game, and when we click option, it change volume of music and save it. You can see main menu picture in down.



MainMenu scene script class

This class is just for Play and Quit button, but option button has different script. Play button is point to playGame function when we click play button invoke playGame function and take us first level of game. Quit button connect to quitGame function, and when we click it, it shut down game. You can see script in down picture.

```
You, 3 minutes ago | 1 author (You)
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4 using UnityEngine.SceneManagement;
5
0 references | You, 3 minutes ago | 1 author (You)
6 public class MainMenu : MonoBehaviour
7 {
8
0 references
9 public void playGame() {
10
11     LoadNextLevel();
12
13     // SceneManager.LoadScene(SceneManager.GetActiveScene().buildIndex + 1);
14
15
16 }
17
18
19
1 reference
20 public void LoadNextLevel()
21 {
22     int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
23     int nextSceneIndex = currentSceneIndex + 1;
24
25     // Check if there's another scene in the build
26     if (nextSceneIndex < SceneManager.sceneCountInBuildSettings)
27     {
28         SceneManager.LoadScene(nextSceneIndex);
29     }
30     else
31     {
32         Debug.Log("No more levels in build!");
33     }
34 }
35
36
37
0 references
38 public void quitGame() {
39
40
41     Application.Quit();
42 }
43
44
45 }
46
```

Change Volume of Music

When player click Option button in the menu, it take another scene to change volume of music. Scene has slider which player can change volume with it, and there are button called Back when player click it, it take back to Main Menu. You can see picture in down.



SoundManager script class

There are play, quite, and option in menu. This class is connect to option button. Option change level of volume with slider as you can picture above. Furthermore, it save volume. I used PlayerPrefs to save volume level. You can see script class down below.

```

1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEditor.UI;
4  using UnityEngine;
5  using UnityEngine.UI;
6
7  0 references
8  public class SoundManager : MonoBehaviour
9  {
10
11     3 references
12     [SerializeField] Slider VolumeSlider;
13     // Start is called before the first frame update
14     0 references
15     void Start()
16     {
17         if (!PlayerPrefs.HasKey("musicVolume"))
18         {
19             PlayerPrefs.SetFloat("musicVolume",1);
20             load();
21         }
22         else {
23             load();
24         }
25     }
26
27     // Update is called once per frame
28
29     0 references
30     public void changeVolumemusic()
31     {
32         AudioListener.volume = VolumeSlider.value;
33         save();
34     }
35
36     2 references
37     private void load() {
38         VolumeSlider.value = PlayerPrefs.GetFloat("musicVolume");
39     }
40
41     1 reference
42     private void save() {
43         PlayerPrefs.SetFloat("musicVolume", VolumeSlider.value);
44     }
45 }
46
47

```

First Level Scene

In the scene, there are two entities: Object Spawned and the Player's sword. The Object Spawned continually generates unlimited objects that move towards the Player. When the Player hits these objects with their sword, the objects get destroyed. As the Player destroys objects, a point label on the top left side of the scene increments by one for each destroyed object, updating the score. If the score

surpasses the maximum score displayed on the top right side of the scene, the maximum score is updated. The Player's sword is equipped with a script. Additionally, the objects have two different scripts: one for creating objects and another for generating an unlimited number of objects in the scene. After the generation of 50 objects, the game becomes more challenging. Initially, objects move straightforwardly toward the player, but as the game progresses, they become faster and move in an up-and-down pattern, increasing the difficulty level. This game not only provides an enjoyable experience for the player but also promotes exercise, as the player needs to move the Oculus controller up and down to engage with the game.



Object Class

This script is for each object in scene. It has two function update and RandomlyMove. Static int boxCubeCount variable keep track how many object generated if it pass 20, it invoke RandomlyMove function which every second object go up and down to make game more difficult for user to destroy.

```
You, 3 seconds ago | 2 authors (You and others)

using System.Collections;
using UnityEngine;
using UnityEngine.SceneManagement;

0 references | You, 3 seconds ago | 2 authors (You and others)
public class BoxCube : MonoBehaviour
{
    3 references
    float speedOfBox = 4f;
    3 references
    static int boxCubeCount = 0;

    // Start is called before the first frame update
    0 references
    void Start()
    {
        // Increment boxCubeCount when a new instance is created
        boxCubeCount++;

        // Check if 50 instances have been created, and if so, start the movement
        if (boxCubeCount > 20)
        {
            InvokeRepeating("RandomlyMove", 2f, 2f);

            speedOfBox += 4;
        }

        Debug.Log(boxCubeCount);
    }

    // Function to start randomly moving up or down
    0 references
    void RandomlyMove()
    {
        // Generate a random adjustment factor between -0.3 and 0.3
        float adjustmentFactor = Random.Range(-0.3f, 0.3f);

        // Apply the adjustment to the position
        transform.position += new Vector3(0f, adjustmentFactor, 0f) * speedOfBox;
    }

    // Update is called once per frame
    0 references
    void Update()
    {
        // Move forward continuously
        transform.position += Time.deltaTime * transform.forward * speedOfBox;
        nebil, 2 months ago • first save
    }
}
```

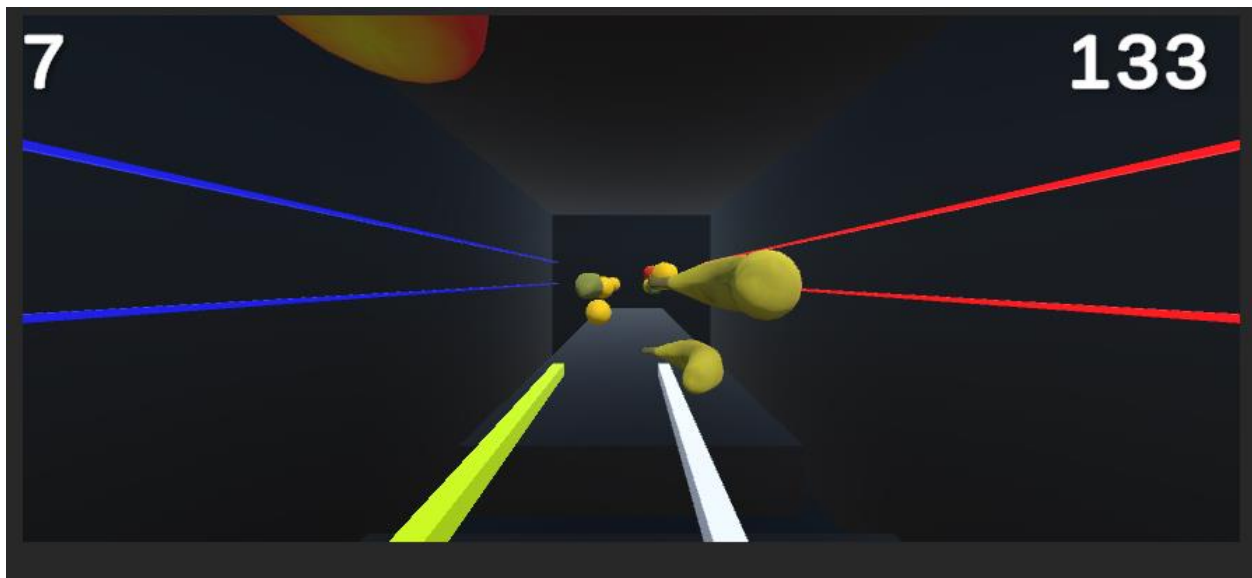
Spawned Class

Spawned script for generated unlimited object in scene. GameObject type array cubes take two different object, and points' type Transform take 4 different location. When game start, unlimited these two object generated and start from random location of points of array. To able to able to generated unlimited object in Unity Game Engine, we use instantiate. In update function I called instantiate and pass object and location parameter, so it generated unlimited object in scene. As you can see in picture down below.

```
1 using System.Collections;
2 using System.Collections.Generic;
3 using UnityEngine;
4
5
6 0 references | nebil, 2 months ago | 1 author (nebil)
7 public class Spawner : MonoBehaviour
8 {
9     1 reference
10     public GameObject[] cubes;
11
12     1 reference
13     public Transform[] points;
14
15     2 references
16     public float beats = (120 / 130) * 2;
17
18     6 references
19     private float timer;
20
21     // Start is called before the first frame update
22
23     0 references
24     void Start()
25     {
26         nebil, 2 months ago + first save
27         // Add this line to ensure there's a Rigidbody component
28         Rigidbody rb = gameObject.AddComponent<Rigidbody>();
29
30         rb.isKinematic = true; // Set it to kinematic if you don't want physics interactions
31
32         // Initialize the timer
33         timer = 0f;
34
35         // Update is called once per frame
36
37     0 references
38     void Update()
39     {
40         // Uncomment the timer logic if you want to spawn cubes at intervals
41         timer += Time.deltaTime * 10;
42
43         // Check if the timer has exceeded the beats interval
44         if (timer > beats)
45         {
46             // Reset the timer
47             timer = 0f;
48
49             // Instantiate the cube at the chosen position
50             GameObject cube = Instantiate(cubes[Random.Range(0, 2)], points[Random.Range(0, 4)]);
51
52             cube.transform.localPosition = Vector3.zero;
53             //cube.transform.Rotate(transform.forward, 90 * Random.Range(0, 4));
54
55             cube.tag = "CubeTag";
56
57             timer -= beats;
58         }
59
60         timer += Time.deltaTime;
61     }
62 }
```

Second Level Scene

In the second scene, things work pretty much the same as the first one – we're keeping it simple and fun for the user. The script that powers the game is almost identical, making the transition between scenes smooth. The major change here is the objects you get to destroy. Instead of the original objects, we've spiced things up with 3D fruits. They're free assets from the Unity Store! So, now, users get to smash and bash their way through a fruity obstacle course. The introduction of these colorful fruits not only adds a visual pop to the game but also gives users a whole new set of things to knock around. It's a bit like turning your workout into a fruit ninja session – healthy and entertaining. The second scene cranks up the challenge a notch by making those fruits move faster. It's like a speed boost for the game. We did this on purpose to keep users on their toes and get their heart rates pumping a bit more. After all, we want this VR workout to be both effective and enjoyable. To sum it up, while the technical side of things stays familiar from the first scene, the introduction of fruity assets and the speedier gameplay in the second scene adds an extra layer of fun and difficulty. It's all about keeping the workout fresh and exciting for the users



Player Class

This script is connected to the Oculus controller, and it turns the sword in the game. When the user moves the controller, the sword moves in the same direction because the Oculus package is integrated into this project. The variable with the type AudioSource is for music, so when the player hits an object, StartCoroutine(PlayAndPauseSound) is invoked. This function plays the music of hitting the object in the scene. The variable scoreUI is the label on the top-left of the scene for the score that the player gets by destroying objects in the game, and the hightScoreUI variable is on the top-right of the scene to show the highest score from all games. When the player's score surpasses the highest score, we save this score with PlayerPrefs.SetInt("score", higherScore) in the saveHigherScore function. When the game starts, the highest score is retrieved from the LoadScore function because this function is called in the start function. The OnTriggerEnter(Collider other) function is the most important function in this game. When the sword collides with an object, this function is invoked, and if the object's tag is CubeTag, the if condition is executed; otherwise, the else condition is executed. In the if condition, we play music, increment the score, and save it. Also, if the player's score surpasses 50, it triggers the second level of the game.

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

0 references | You, 3 weeks ago | 2 authors (You and others)
public class Saber : MonoBehaviour
{
    2 references
    public AudioSource source;
    //public AudioClip clip;
    6 references
    private int destroyedObjectCount = 0;
    5 references
    private int higherScore = 0;

    2 references
    public float sensitivity = 2.0f;
    1 reference
    public TextMeshProUGUI scoreUI;
    1 reference
    public TextMeshProUGUI hightScoreUI;

    0 references
    private void Start() {
        if (!PlayerPrefs.HasKey("score"))
        {
            PlayerPrefs.SetFloat("score", 0);
            loadScore();
        }
        else {
            loadScore();
        }
    }
}
```

```

void Update()
{
    if (Input.GetKeyDown(KeyCode.Space))
    {
        // Call the method to load the next level
        LoadNextLevel();
    }

    this.controllerSaberWithMouse();
}

1 reference
void controllerSaberWithMouse()
{
    float mouseX = Input.GetAxis("Mouse X");
    float mouseY = Input.GetAxis("Mouse Y");

    // Adjust the hand anchor position based on mouse movement
    Vector3 newPosition = transform.position + new Vector3(mouseX * sensitivity, mouseY * sensitivity, 0f);
    transform.position = newPosition;
}

nebil, 2 months ago • first save
References
private void OnTriggerEnter(Collider other)
{
    if (other.gameObject.tag == "CubeTag")
    {
        //source.Play();
        StartCoroutine(PlayAndPauseSound());

        destroyedObjectCount++;

        if(destroyedObjectCount > higherScore) {
            higherScore = destroyedObjectCount;

            saveHigherScore();
            loadScore();

        }

        // source.Pause();

        scoreUI.text = destroyedObjectCount.ToString();
        // hightScoreUI.text = higherScore.ToString();

        Debug.Log("Destroyed objects: " + destroyedObjectCount);
        Destroy(other.gameObject);

        if (destroyedObjectCount > 50)
        {
            LoadNextLevel();
        }
    }
}

```

```

1 reference
IEnumerator PlayAndPauseSound()
{
    // source.PlayOneShot(clip);
    | source.Play();
    yield return new WaitForSeconds(0.5f); // Adjust the duration as needed
    source.Pause();
}

1 reference
private void saveHigherScore() {
    PlayerPrefs.SetInt("score", higherScore);
}

3 references
private void loadScore() {
    higherScore = PlayerPrefs.GetInt("score", 0);
    hightScoreUI.text = higherScore.ToString();
}

2 references
public void LoadNextLevel()
{
    int currentSceneIndex = SceneManager.GetActiveScene().buildIndex;
    int nextSceneIndex = currentSceneIndex + 1;

    // Check if there's another scene in the build
    if (nextSceneIndex < SceneManager.sceneCountInBuildSettings)
    {
        SceneManager.LoadScene(nextSceneIndex);
    }
    else
    {
        Debug.Log("No more levels in build!");
    }
}

```

Asset of Game

I got couple assets for this project, and rest of design, I used Unity tools. Second level object is fruit, and I got this 3D asset from unity store. For music I download music from website that called pixabay.com

Unity Asset Store

Search for assets

3D

2D

Add-Ons

Audio

AI

Decentralization

Essentials

Templates


Tools

VFX

Sale

Sell Assets

Home > 3D > Props > Food > Low Poly Fruit Pickups



Low Poly Fruit Pickups

Rem Storms

★★★★★ (21) | ❤️ (875)

FREE

👁 185 views in the past week

Add to My Assets

jrmgamedev

★★★★★ 8 months ago

Excellent Assets

Quality - 10/10

Ease of Use - 10/10

File Size - 10/10

These assets are really helpful in my little project, thank you.

Sound Effects ▾ Duration ▾ Theme ▾

⚙ Most relevant ▾

188 royalty-free sword sound effects

Download sword royalty-free sound effects to use in your next project.

Royalty-free sword sound effects. Download a sound effect to use in your next project.

metal battle swords hit impact blade knife draw medieval

Royalty-free sound effects

px

Sword Thud

Pixabay

🔊 0:08

Download ▾

px

Sword sound 2

Pixabay

🔊 0:0

Download ▾

Challenges and Solution

One of the biggest challenges I faced was integration of Oculus packages for Unity because the camera of Unity is different than Oculus. To overcome that difficulty, I watched some YouTube Tutorial and read Unity documentation for Oculus integration. Another challenge was Unity functionality. That is my first time to develop a game with Unity so that I was similar to use it. I purchased some Unity tutorial in Udemy, and then I learnt. Also, Java knowledge helped me a lot because C# and Java syntax is pretty similar. My third challenge was the Control system version Git and GitHub. When I work in any project, I use Git because when I broke code, I can checkout and go back. However, I couldn't do for Unity project, and I couldn't understand for a couple of weeks. After more research, I learnt, Git allows commit for a certain size due to the Unity project is a big size, I wasn't able to save it so that I copy past Project for every progress I have done if I broke code, I check from copy project that how I overcome that challenge.

Future Enhancements

In the future, I plan to make my capstone project even better. More game levels with different challenges and enemies to keep things interesting. I want players to feel a sense of accomplishment, so I'll create a scoring system that takes into account how well they play. Users will have the freedom to personalize their gaming experience by choosing different swords, changing background music, and even making objects look cooler. I'm thinking about adding a multiplayer option so friends can join in the fun together. Plus, I want to make the game more a workout routine, with exercises tied to the game actions.

I want the game to work on different devices like HTC Vive and PlayStation VR. Maybe even connect it to fitness trackers to keep an eye on how much exercise players are getting. And get this—how about a virtual trainer in the game to guide users through warm-ups and cool-downs? I'm considering letting users buy extra cool stuff in the game. You know, like special powers or new levels. And to make things fair, I'm thinking of adjusting the game difficulty based on how good players are getting. Lastly, I will add Adverts Google to project which help to make money from Ad.

Learning

In this project, I jumped into C# programming and got the hang of using the Unity game engine. I also figured out how to make things cool with Oculus for virtual reality. Designing for VR in Unity was a bit tricky, but I got better at it. As I learned the ropes, I found useful stuff like the Unity documentation. It was like a helpful guide that got me through tough parts and made problem-solving easier. It wasn't just about coding—I learned how to turn my ideas into awesome-looking games. Mixing design tricks with what Unity can do became my way of being creative in the virtual world. Another important part was learning how to write down what I did. I got good at explaining the details of my finished projects, making a kind of story about how they came to life. It's not just for me—it's also for others who might find it useful. Throughout this learning adventure, Unity's documentation was like a trusty friend, helping me tackle challenges and learn more. Knowing that learning never really stops and being able to share what I find makes this project's story even more interesting

Citation

Radzicki, Melanie. "6 creative ways to transform your workouts." CNN, Cable News Network, 6 Jan. 2022, www.cnn.com/2022/01/06/health/creative-workout-motivation-wellness/index.html.