CSE 396 – Computer Engineering Project

Gebze Technical University

Spring 2024

Virtual Strike Project – Final Demo Report

Prepared By VS Games

(Group 4)

Table of Contents:

1. Group members

- Ahmet ÖZDEMİR
- Akif Safa ANGİ
- Aykut SERT
- Burak KURT
- Doğukan BAŞ
- İlkay BOLAT
- Muhammet Yasir GÜNEŞ
- Murat ERBİLİCİ

2. Summary

We are pleased to announce the successful completion of the "Virtual Strike" project. This project aimed to develop an immersive VR shooting game that integrates real-life movements with virtual gameplay. All project modules have been thoroughly tested and validated, and they function seamlessly together to provide an engaging and interactive gaming experience. The game allows players to use VR headsets and a specialized aiming device to shoot targets, with real-time feedback and realistic physics.

3. Introduction

This report analyses the status of the steps taken, the methods of progress, the problems encountered and the solutions to these problems up to the finaldemo of Group 4's project, separately for each module.

Description: This project aims to provide the player with real-life sensations in the game to be played through VR glasses. The game is basically based on shooting targets. This game will provide players with a real-time and interactive gaming experience, giving them the feeling of a real shooter as they take aim at targets within the game.

"Virtual Strike" will be played using a specialized aiming device integrated with the Unreal Engine game engine and VR (Virtual Reality) headsets. The main goal of the project is to transfer real-life physical activity into the game world while providing players with interactive entertainment. This will make the game both fun and attractive for players who want to spend time in a healthy way.

The "Virtual Strike" project represents an innovative approach that combines technology and entertainment to offer players an immersive shooter experience in a virtual reality world. With the successful completion of the project, players will experience a unique gaming experience where they will be able to control their real-world actions in the virtual world. The player needs the following devices to play the game:

- An Android phone
- A Computer
- A hand-held device as a remote control
- A VR glasses

4. Project Management

The Virtual Strike project followed the Scrum methodology, ensuring an agile and iterative approach to development. Ahmet Özdemir served as the Scrum Master, overseeing the project and facilitating communication among team members. The team was composed of dedicated developers who worked collaboratively to achieve the project goals.

Scrum Framework

The Scrum framework was chosen for its flexibility and efficiency in handling complex projects. The framework allowed the team to adapt to changes quickly and deliver incremental progress through regular sprints. Each sprint lasted one week, with the weekend reserved for face-to-face meetings and the weekdays for online coordination.

Roles and Responsibilities

- **Scrum Master**: Ahmet Özdemir
    - Facilitated meetings
    - Ensured adherence to Scrum practices
    - Resolved impediments

- **Development Team**:
    - Ahmet Özdemir
    - Akif Safa ANGİ
    - Aykut SERT
    - Burak KURT
    - Doğukan BAŞ
    - İlkay BOLAT
    - Muhammet Yasir GÜNEŞ
    - Murat ERBİLİCİ

Project Phases

1. Planning Phase

- The team identified key modules: Game Backend and Mechanics, Game Design and User Interface, Mobile Application and Data Transfer, Aim - Motion Control Circuit Module, and Communication Between Controller Hardware and Computer.
- Developers were assigned to each module based on their expertise and the requirements of the module.

2. Development Phase

- The project began with the development of the server model, alongside the controller and game modules.
- **Server Model**:
    - Developed to handle game logic and data management.
    - Key developers: Doğukan Baş.
- **Controller Module**:
    - Focused on creating a functional remote control device for in-game interaction.
    - Key developers: Ahmet Özdemir, Doğukan BAŞ, Muhammed Yasir Güneş, İlkay Bolat.
- **Game Module**:
    - Developed the core game mechanics and user interface.
    - Key developers: Burak Kurt, Akif Safa ANGİ, Aykut SERT, Murat ERBİLİCİ (and Doğukan Baş - Ahmet Özdemir for integration of controller and mobile device sensors into the game).

Reassignment and Parallel Development

- Upon completion of the controller module, developers from this module were reassigned to the Android and Communication modules.

Android and Communication Modules:
    - Developed to ensure seamless data transfer between the mobile device and the server.
    - Key developers: Doğukan Baş, Ahmet Özdemir.

- The game module continued development with the initial team until all other modules were completed.

Communication and Coordination

- **Weekly Meetings**:
    - **Face-to-Face**: Every weekend, the team met in person to discuss progress, plan the upcoming week's tasks, and resolve any issues.
    - **Online**: During the week, the team held online meetings to coordinate daily tasks, review progress, and ensure alignment with project goals.

Sprint Reviews and Retrospectives

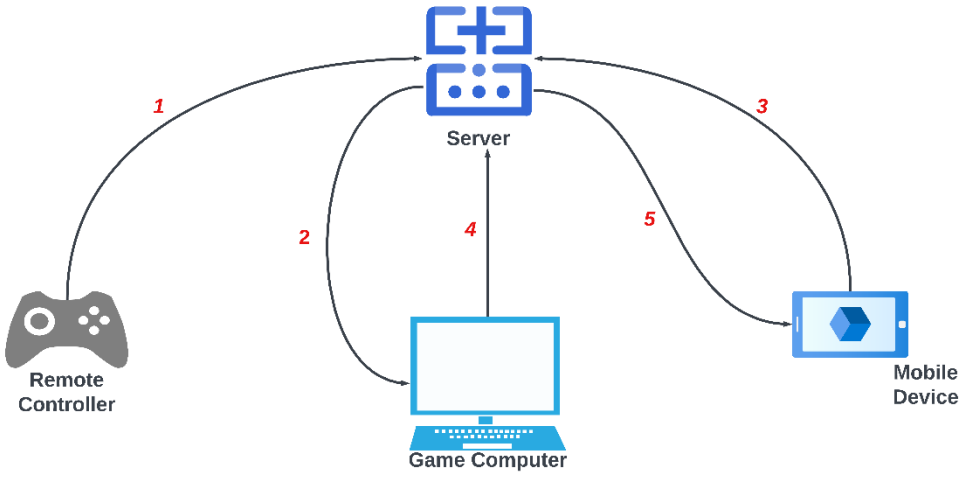
- At the end of each sprint, the team conducted a sprint review to demonstrate the progress made and gather feedback.
- Retrospectives were held to reflect on the sprint, identify what went well, and determine areas for improvement.

Conclusion

The Virtual Strike project successfully adhered to the Scrum methodology, enabling the team to manage the project efficiently and deliver a high-quality VR game. The iterative approach and regular communication ensured that all modules were developed cohesively, and the project objectives were met. The dedication and collaboration of the development team, under the guidance of Scrum Master Ahmet Özdemir, were key factors in the successful completion of the project.

5. System Design and High-level architecture of the project:

In general, the project enables a game made with the Unreal Engine game engine to be played without direct interaction with the computer. In this case, the in-game control mechanism is realised through a remote control and a mobile Android application.



1. ***Directional information for in-game movement:***

   Remote controller: Includes an Arduino Uno, Raspberry Pi 4, joystick button. With a code written in C++, the circuit sends analogue direction and switch data from Arduino to the server via Raspberry Pi.

2. ***The movement information is transmitted via the server to the game for processing:***

   The direction data is sent to the game via the server as received from the remote controls (-1, 0, 1). In the game, this data is processed appropriately and the joystick button allows movement in 8 directions.

3. ***For camera angles and viewing directions, data from gyroscope and acceleration sensors are transmitted from the mobile device to the server.***

   In the game, this data acts as a mouse. Where the player looks in real life, the in-game camera also looks.

4. ***Screen scenes are continuously sent by the game to the server in base_64 format to be projected on the screen of the mobile device.***

   In the game, the entire screen state of the game is taken quickly frame by frame and prepared in base_64 format and sent to the server.

5. ***Data of type base_64 is received from the server to be displayed on the screen:***

   In this step, the base_64 formatted image data received from the server is rendered to be compatible with the VR glasses and given to the screen in the form of two eyes.

6. Modules and Modules Requirements

MODULES

Modules are separated in order to better manage the project and to better distribute the division of labour among the group members. First of all, we divided the project into two as game making and tool making. The game production part will be divided into two modules, the first module will be the mechanics behind the game, in other words the backend part. The second part will be the design and user interface of the game. The construction of the tools will be the remote control in the hand and the reflective app on the phone to which the android device is connected. There will also be a module for communication between the controller and the game.

A - Game Backend and Mechanics:

Module Description: This module encompasses the development of the backend systems and mechanics of the Virtual Strike game. It involves implementing the core functionality of the game, such as player movement, target interactions, scoring mechanisms, and game physics.

Requirements:

- Player Movement System:

o Implement a system to capture player movements accurately using VR glasses and the handheld device.

o Translate real-world movements into in-game actions, allowing players to navigate the virtual environment seamlessly.

- Target Interaction:

o Develop mechanisms for spawning and managing targets within the game environment.

o Implement hit detection algorithms to register successful shots on targets.

- Scoring System:

o Design a scoring system based on factors such as accuracy, speed, and precision.

o Ensure that scores are calculated and updated in real-time during gameplay.

- Game Physics:

o Integrate realistic physics simulations to enhance the gaming experience.

o Implement projectile physics for bullets and other in-game objects.

•Technologies and Tools:

- Unreal Engine: Utilize Unreal Engine for game development, including blueprint scripting and C++ programming for backend functionalities.

- VR Development Kit: Make use of VR development kits compatible with Unreal Engine for VR

integration.

- Version Control System: Employ a version control system (e.g., Git) to manage collaborative development and track changes effectively.

- Integrated Development Environment (IDE): Use appropriate IDEs such as Visual Studio for C++ development and blueprint scripting within Unreal Engine, Android Dtudio for mobile application.

- Testing and Debugging:

Conduct rigorous testing to ensure the reliability and stability of backend systems and mechanics. Perform debugging and troubleshooting to address any issues or bugs encountered during development.

B - Game Design and User Interface:

•Module Description: This module focuses on the design aspects and user interface development of the Virtual Strike game. It involves creating visually appealing game elements, intuitive controls, and an immersive user experience.

•Requirements:

- Game Design:

o Develop a compelling game concept and story that engages players.

o Increase player immersion and enjoyment by designing game levels, environments and elements.

o Ensure consistency in art style, theme and narrative throughout the game.

-Test and Iteration

Conduct usability tests to gather feedback on game design and user interface elements. Revise designs and improve usability and player engagement based on user feedback and playtesting sessions.

C – Mobile Application and Data Transfer:

•Module Description: This module includes the development of a mobile application that transfers the screenshot of the computer game to the mobile device and the creation of a data transfer mechanism that transmits the movements of the mobile device to the computer.

•Requirements:

- Mobile App Development:

o Develop a mobile application to capture the computer game's screen.

o Regularly capture the screen of the computer game and mirror it to the mobile device through the mobile app.

- Data Transfer Mechanism:

o Develop a data transfer mechanism to capture the mobile device's motion data (e.g., joystick data).

o Transmit motion data to the computer at regular intervals to reflect the player's real-time movements in the game.

•Technologies and Tools:

Mobile App Development: Use development environments like Android Studio, (with Java). Data Transfer: Implement technologies like socket communication or WebSocket to transmit mobile device motion data to the computer.

•Testing and Debugging:

Test the accuracy and performance of the mobile app and data transfer mechanism. Address any issues encountered during the debugging process and resolve errors within the system.

D - Guidance - Motion Control Circuit Module:

•Module Description: This module focuses on developing a motion control circuit that enables users to change the direction of their in-game character based on real-life hand movements. Additionally, users can aim and shoot at the target location using a button that interacts with position data from a mobile device. The circuit will include a joystick, a Raspberry Pi and a Arduino UNO to convert analog data to digital data.

•Requirements:

Hardware Components:

o Motion Sensor: Integrate a motion sensor (e.g., accelerometer or gyroscope) capable of

detecting hand movements accurately.

o Button: Include a tactile button for triggering actions such as shooting or initiating specific commands.

o Raspberry Pi: Utilize a Raspberry Pi microcontroller to process input data from the motion

sensor and button and generate corresponding output signals.

•Communication Protocol:

Use protocols Wi-Fi for wireless communication between the Raspberry Pi and computer.

• Test and Calibration:

Conduct extensive testing to calibrate the motion sensor and button interface for optimal performance.-Ensure compatibility and stability across different hand gestures and button inputs.

E - Communication Between Controller Hardware and Computer:

•Module Description: This module involves the creation of a communication protocol that facilitates communication between the controller hardware and the computer. The controller hardware will transmit real-time motion and button information to the computer, enabling in-game controls.

•Requirements:

Communication Protocol:

o Define a communication protocol to be used between the controller hardware and the computer.

o Ensure the protocol supports reliable, fast, and real-time data transmission.

Data Transmission:

o Develop a data transmission mechanism to transfer motion and button information from the controller hardware to the computer.

o Ensure the data transmission mechanism provides low latency and high data accuracy.

•Technologies and Tools:

Hardware Interfaces: Select and integrate hardware interfaces compatible with the chosen communication protocol.

Data Transmission Software: Develop or select software on the computer side to receive data or establish communication.

•Testing and Debugging:

Create a comprehensive test plan to verify the accuracy and reliability of the communication protocol. Verify that data transmission operates correctly at each step and address any errors encountered during testing.

7. Implementation and Development

The implementation and development phase of the Virtual Strike project was a comprehensive process involving multiple stages and extensive collaboration among team members. Each module of the project was developed meticulously, adhering to the planned design and requirements. The primary focus during this phase was to translate theoretical designs into functional components, ensuring seamless integration and robust performance.

This section details the specific steps taken during the implementation and development of each module, outlining the methodologies, tools, and technologies utilized. It also highlights the challenges faced and the solutions employed to overcome them, providing a thorough overview of the development lifecycle from inception to completion.

A - Game Backend and Mechanics:

….

B - Game Design and User Interface:

LEVEL 1:

Game Mechanics and Object Creation

Initially, our primary focus was on setting up the fundamental game mechanics. We designed and implemented two types of objects: kegs, which the player needs to shoot, and wooden deers, which the player must avoid shooting. This stage required attention to detail to ensure the objects behaved as intended.

Challenges and Solutions:

Physics and Movement: One of the major challenges was configuring the movement and collision properties of these objects. Ensuring that the kegs and wooden derrs moved smoothly and reacted correctly to collisions required extensive testing and fine-tuning of Unreal Engine's physics settings.

Collision Detection: Another difficulty was ensuring accurate collision detection, which is crucial for the gameplay experience. We utilized Unreal Engine’s collision channels and fine-tuned the collision boundaries to achieve the desired accuracy.

Collision Detection and Movement Functions:

```
void AAllyActor::OnHit(UPrimitiveComponent* HitComp, AActor* OtherActor, UPrimitiveComponent*
OtherComp, FVector NormalImpulse, const FHitResult& Hit)
{
    if (OtherActor != nullptr)
    {
        if (AGameProjectile* actor = Cast<AGameProjectile>(OtherActor))
        {
            if (actor != nullptr)
            {
                PlayerCharacter->Score -= 5;
                this->Destroy();
            }
        }
    }
}
void AAllyActor::UpdateLocation(float DeltaTime)
{
    Direction = FVector::BackwardVector;
    if (Speed != 0.0f)
    {
        FVector CurrentLocation = GetActorLocation();
        FVector NewLocation = CurrentLocation + (Direction * Speed * DeltaTime);
        SetActorLocation(NewLocation);
    }
}
```

Boundary and Scoring System

After establishing the basic mechanics, we implemented a system to handle the objects once they crossed a specific boundary. This was essential for maintaining game flow and ensuring that missed objects did not persist indefinitely.

Challenges and Solutions:

Object Removal: Managing the removal of objects that crossed the boundary was challenging, as it needed to be done efficiently to avoid performance issues.

Scoring Logic: Integrating the scoring system presented its own set of challenges. We needed to ensure that the score is updated correctly based on the player's actions. This involved creating a system that handled both positive and negative score changes seamlessly.

Game Start and Restart Features

Next, we added functionality to start and restart the game. This included implementing a countdown timer that begins when the player picks up the rifle and providing options to restart the game or return to the lobby. We used additional boolean variables to keep the status of the game.

Challenges and Solutions:

Timing and Synchronization: Ensuring the timer started accurately and synchronized with other game events was a significant challenge. We used Unreal Engine's built-in timing functions and conducted extensive testing to ensure precise timing.

Visual and Audio Design

Finally, we focused on enhancing the visual and audio experience. This involved preparing appropriate textures for the objects, designing the map according to our desired theme, and adding sound effects.

Challenges and Solutions:

Texture and Theme: Creating textures that fit the game's polygonal art style while maintaining performance was challenging. We used tools like Quixel Bridge for the textures that fit in our game theme.

Sound Integration: Adding sound effects to enhance immersion required careful selection and placement of audio cues. We used spatial audio features in Unreal Engine to ensure sounds were realistic and contributed to the overall experience.

Conclusion

The development of Level 1 was a complex process that involved overcoming numerous challenges. Through careful planning, collaboration, and iterative testing, we successfully translated our theoretical designs into a fully functional and engaging VR game level. Each stage of development— from mechanics and scoring to visual and audio design—was crafted to provide players with an immersive and enjoyable experience.

LEVEL 2:

Game Mechanics and Object Creation:

First, we focused on creating the basic game mechanics. We designed and implemented two types of objects: barrels, which the player should shoot, and wooden deers, which should not be shot. This phase required attention to detail to ensure the objects behaved as desired.

Challenges and Solutions:

In this part, the high-quality design of the game map had to be kept at low settings due to the game freezing and stuttering when run, requiring extensive testing and fine-tuning of Unreal Engine's physics settings.

Random Existence of Objects:

We took care to set the objects to appear as you approach certain areas of the forest to increase the difficulty and further the feeling of play. We paid attention to the fact that the objects appear randomly, the deer object comes first and hides behind various game objects (such as rocks, trees, grass).

Limit and Scoring System:

After establishing the basic mechanics, we implemented how objects would be handled once they crossed a certain limit. For this part, after reaching the end of the course in the game and getting points for each object hit, the game closes and returns to the lobby when the objects run out. Here, the player is given a certain amount of time to prevent the game from remaining in an endless loop, and when the time expires, the game automatically directs them to the lobby.

Challenges and Solutions

Object Removal: Managing objects crossing the boundary was challenging as it had to be done efficiently to avoid performance issues. Deer objects that are not shot in this section remain on the map in the game and this did not cause any problems in terms of performance.

Scoring Logic:

When the player shot deer objects, the score was reduced and we increased the score for each barrel object he shot.

Game Launch and Restart Features

Next, we added functionality to launch and restart the game. This included a countdown timer that started as soon as the player picked up the rifle, and options to restart the game or return to the lobby. We used additional boolean variables to maintain the state of the game.

Challenges and Solutions

Timing and Synchronization: Getting the timer to start correctly and sync with other game events was a significant challenge. We used Unreal Engine's built-in timing functions and conducted extensive testing to ensure precise timing.

Visual and Sound Design

Finally, we focused on enhancing the visual and audio experience. This involved preparing appropriate textures for the objects, designing the map according to the theme we wanted, and adding sound effects.

Challenges and Solutions:

Texture and Theme:

The high quality of the map and objects of the game made it difficult for us to make many changes on the map. We created a faster and more logical route and game logic by working at low settings.

Sound Selection:

Our game sound, which includes a forest theme and overlaps with owl sounds, made the game more realistic and palpable. By using Unreal Engine's spatial audio features, we ensured that the sounds were realistic and added to the overall experience.

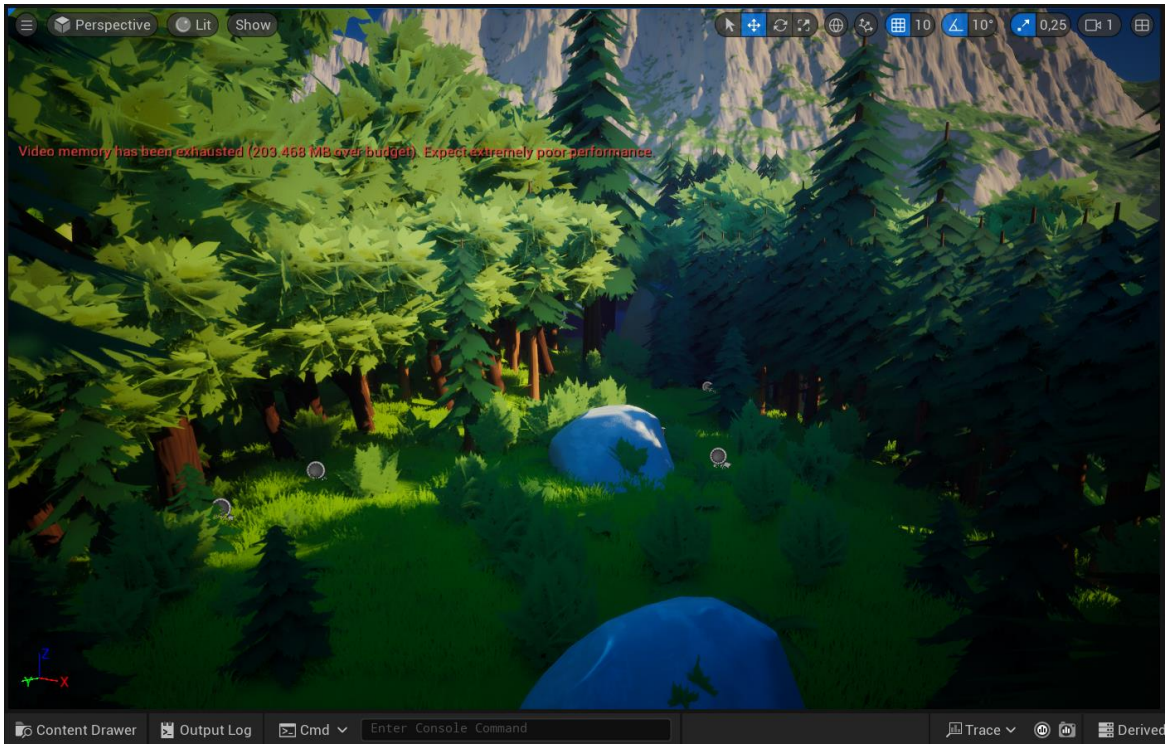
Conclusion:

The development of the second level was a complex process that involved overcoming several challenges. Through careful planning, collaboration, and iterative testing, we successfully transformed our theoretical designs into a fully functional and engaging VR gaming level. From mechanics to scoring to visual and audio design, each phase of development has been meticulously crafted to provide players with an immersive and enjoyable experience.

Game Design Additional Information :

In order to make the difference in the quality of the graphics in the game more striking, we have presented two different screenshots from the game at low and epic graphics settings below. Although Epic settings have very high quality display, we preferred low settings because the game fluidity did not decrease due to lag and freezing and our system was of low quality.

Low Quality



Epic Quality

LEVEL 3:

Level 3 Design:

Floor Mechanics: At the starting of a Level 3 parkour game, there are 15 floors on the map by default. As the game progresses, new floors are created and previous floors are deleted from the map. This is an important mechanic to ensure the dynamic and endless running experience of the game.

Placement of Objects: Coins, obstacles and bounce objects are randomly generated on each floor. These objects diversify the player's game experience and are effective in adjusting the difficulty level.

Character Movement: The character makes an infinite run and the speed increases as time passes. This speed increase gradually increases the difficulty level of the game.

Obstacle Traversal Mechanics: The player needs to jump over some obstacles, slide under others, and sometimes change lanes. These movements are controlled with the buttons on the controller.

Environment Design: Randomly generated buildings can self-destruct just like floors. This allows the game world to be constantly renewed.

Score Mechanics: As the player collects coins, the score variable increases and this is used to measure the player's performance.

Camera Movement: The character's camera movement mechanism allows the player to examine the environment by turning their head. Also, when the player lands on the bounce object, he jumps more than normal.

Coding and Implementation:

Floor Management: An algorithm has been developed to dynamically create new floors and delete old floors.

Random Object Placement: Randomisation algorithm is used for random placement of coin, obstacle and bounce objects on the floors.

Speed Increase: A time-based speed increase mechanism has been coded for the character to speed up as time passes.

User Inputs: User inputs (control keys) are integrated for obstacle avoidance (jump, slide, lane change).

Environment Management: Random building creation and destruction mechanisms have been developed.

Scoring System: A system that increases the score as coins are collected has been coded.

Camera Settings: Camera movement algorithms have been added for bounce objects and slide movement.

C – Mobile Application and Data Transfer:

Library Setup:

- Google's Vr.sdk and opengl libraries were used to create VR dual stereoscopic images. With these libraries, a snapshot of the game was projected on the mobile screen with graphics.Bitmap, ByteOrder, ByteBuffer libraries to convert the byte type image coming from the server into Bitmap. WebSocketClient, URI and ServerHandshake libraries were used to instantly send the byte image from the server and the mobile's gyroscope and accelerometer data to the server and to retrieve the data.

Data Processing and Transfer

- The byte image from the game was transferred to the mobile application via the server using the 3030 port. This byte data, together with the necessary VR libraries, was aligned to the mobile screen in a double stereoscopic frame and synchronously converted into a live VR image.
- Likewise, the accelerometer X and Z data, along with the mobile's own gyroscope X and Z data, were transferred to the server using the 3000 port.
- Gyroscope X and Z data were processed in a way that allowed it to rotate 360 degrees left and right in the game, providing users with full flexibility.
- At the same time, data processing was done on VS game to enable the accelerometer to rotate 180 degrees up and down using X and Z data, allowing users to feel a full VR gaming experience.

Required Settings

- The byte image coming from the server was first converted to Bitmap and processed in the functions of the necessary VR libraries. First of all, a dual stereoscopic texture was drawn with the GLES20 library and requirements. Necessary matrix adjustments were made to these textures. In order to make this double frame more appealing to the eye, the right and left eye frames were shifted in the X directions. In this way, dual stereoscopic images were designed so that users wearing VR glasses can see a complete image.
- With vertexShader and fragmentShader, a red dot aligned to the middle of the mobile screen was set for the user.

Testing and Debugging

- Necessary tests were carried out to instantly reflect the snapshot of the game in a double stereoscopic frame and to detect the right, left, up and down camera directions. As a result of these tests, users were offered a full VR gaming experience.
- Sensitivity has been adjusted in both the X (left, right) direction and the Y (up and down) direction for more optimized adjustment of users' head movements. In this way, users were offered the potential to play the game more easily without spending much effort.

D - Guidance - Motion Control Circuit Module:

Implementation Steps:

1. **Hardware Setup:**
    - The motion control circuit was designed using an Arduino UNO and Raspberry Pi 4, integrated with a joystick and a button for shooting actions.
    - The joystick provides directional input, while the button is used to trigger shooting events.
2. **Circuit Design and Assembly:**
    - These circuit setups and diagrams illustrate the connections between the joystick, button, Arduino, and Raspberry Pi (Ilustraions 10.c.1, 10.c.2, 10.c.3, 10.c.4, 10.c.5)
    - The joystick and button were connected to the Arduino, which reads the analog signals and converts them into digital data.
3. **Sensor Data Handling:**
    - The Arduino was programmed using C++ (**controller.cpp**) to capture analog inputs from the joystick and button presses. The code converts these analog signals into digital data that the Raspberry Pi can process.
    - The script **dummyController.py** on the Raspberry Pi ensures the data received from the Arduino is reliable before passing it on.
4. **Data Transmission:**
    - Upon startup, the Raspberry Pi runs the **init.sh** script from the **script** directory. This script initializes the Python script to verify data integrity from the Arduino and then runs the C++ program to handle sensor data transmission via WebSocket.
    - The WebSocket protocol was used to send data from the Raspberry Pi to a local server, ensuring low-latency communication.
5. **Integration with Game:**
    - The local server receives joystick and button data and relays it to the Unreal Engine game. This integration allows real-time control of in-game movements and shooting actions.
    - For shooting, the button press data is transmitted and processed to trigger shooting events within the game, providing immediate feedback to the player.

Challenges and Solutions:

1. **Ensuring Accurate Motion Detection:**
    - **Challenge:** Initial inaccuracies in joystick data readings.
    - **Solution:** Conducted extensive calibration of the joystick inputs on the Arduino to ensure precise movement detection. The calibration involved mapping the analog input ranges to the expected digital output values for accurate data interpretation.
2. **Reliable Data Transfer:**

- **Challenge:** Ensuring reliable data transmission from Arduino to Raspberry Pi.
- **Solution:** Used the **dummyController.py** script to validate the data received from the Arduino before passing it to the WebSocket for transmission. This step ensured that only accurate and reliable data was sent to the game.

3. **Low-Latency Communication:**
    - **Challenge:** Maintaining low latency for real-time gameplay.
    - **Solution:** Implemented WebSocket for efficient and fast data transfer. Opted for a local server setup to minimize latency, providing a seamless gaming experience.
4. **Button Press Sensitivity:**
    - **Challenge:** Ensuring responsive shooting actions upon button presses.
    - **Solution:** Programmed the Arduino to detect button presses with minimal debounce time, ensuring that each press was accurately captured and transmitted without delay.

Technologies and Tools:

- **Hardware:** Arduino UNO, Raspberry Pi 4, Joystick, Button
- **Software:**
    - C++ for Arduino programming (**controller.cpp**)
    - Python for data handling on Raspberry Pi (**dummyController.py**)
    - WebSocket for data transfer
- **Network:** Local Wi-Fi network for inter-device communication

Summary:

The motion control circuit module was successfully implemented, enabling accurate and real-time in-game controls. By using detailed circuit design, robust programming, and efficient data transmission protocols, the system achieved low-latency performance and reliable input detection. The extensive testing and calibration ensured the module's functionality met the project's objectives, providing an immersive and interactive gaming experience.

E - Communication Between Controller Hardware and Computer:

Implementation Steps:

1. **Data Capture from Joystick:**
    - The joystick is connected to the Arduino UNO, which reads the analog input from the joystick and converts it to digital data using a C++ program.
2. **Data Transmission to Raspberry Pi:**
    - The Arduino sends the processed joystick data to the Raspberry Pi via serial communication. Because the Raspberry Pi doesn’t read the joystick sensor data correctly but the Arduino read it correctly we need an extra help of Arduino to get reliable data transfer for Raspberry Pi. This ensures reliable and timely data transfer.

3. **Local Server Setup:**
    - A local server was set up to handle data communication between the Raspberry Pi and the game computer. All devices (Raspberry Pi, local server, and game computer) were connected to the same Wi-Fi network for fast and secure data transmission.
4. **Data Transfer from Raspberry Pi to Local Server:**
    - When the Raspberry Pi boots, it immediately runs a script to send joystick data to the local server using WebSocket.
      The script firstly executes a python code to ensure it gets reliable data from the arduino which is **dummyController.py.** And then executes the **controller.cpp** which does the actual job. It handles the sensor data transmission through WebSocket.
5. **Game Integration:**
    - The game application, developed in Unreal Engine, receives data from the local server, interpreting it to control in-game movements. This provides real-time feedback and control for the player.
6. **Alternative Global Server Approach:**
    - Though not used, an alternative method was considered where the Raspberry Pi could send data to a global server. This server would relay the data to the game. The local server was preferred due to its faster data transfer rates.
    - It can be prefable because we need to connect rasperry pi, computer and the local server to the same internet. So we need to adjust the rasperry pi to that IP adress of the local server everytime the user builds connection through a new server. But if global server is used the IP adress will remain the same so the player doesn't have to change the IP adress which data will be sent.

Technologies and Tools:

- **Hardware:** Arduino UNO, Raspberry Pi 4, Joystick
- **Software:** C++ for Arduino programming and sending sensor data to the server,
  Python for ensuring reliable data transmission from arduino to Raspberry Pi,
  WebSocket for data transfer
- **Network:** Local Wi-Fi network or Server

Challenges and Solutions:

1. **Low-Latency Data Transmission:**
    - Optimized socket communication protocols were used to ensure low latency, achieving real-time responsiveness.
    - Used local server for fast data transmission and eliminate low-latency.
2. **Reliable Data Transfer:**
    - Implemented robust serial and WebSocket communication interfaces, with extensive testing to ensure stable data flow.
3. **Data Synchronization:**
    - Synchronized clocks between the Raspberry Pi and the local server to ensure consistent and accurate data transfer, reducing synchronization issues.

Summary:

The communication module between the controller hardware and the computer was successfully implemented, enabling real-time and accurate in-game controls. By using a local server for faster data transfer, the system achieved low-latency performance, enhancing the gaming experience. Extensive testing and optimization ensured the reliability and performance of the communication system, meeting the project's objectives effectively.

8. Test and Verification

The test and verification phase is crucial in ensuring the reliability, functionality, and performance of the Virtual Strike project. This phase involves systematically evaluating each module to identify and rectify any defects or issues, ensuring the final product meets the specified requirements and performs as expected. Rigorous testing methodologies, including unit tests, integration tests, system tests, and user acceptance tests, were employed to validate the system.

This section outlines the testing strategies, methodologies, and tools used during this phase, providing a comprehensive overview of the tests conducted, the results obtained, and the verification of each module's functionality and performance.

A - Game Backend and Mechanics

Testing Strategies:

1. **Unit Testing:**
    - Developed unit tests for individual functions and methods within the game mechanics.
    - Ensured that each unit of the backend system operated correctly in isolation.
2. **Integration Testing:**
    - Conducted integration tests to verify the interaction between different components of the backend.
    - Ensured seamless integration and communication between the player movement system, target interaction, scoring system, and game physics.
3. **System Testing:**
    - Performed comprehensive system testing to validate the overall functionality of the backend.
    - Ensured that the game mechanics operated correctly within the complete system environment.

Results:

- **Outcome:** All unit, integration, and system tests passed successfully.

- **Issues Identified:** Minor discrepancies in the scoring system which were promptly fixed.
- **Verification:** Backend functionality verified through extensive testing.

B - Game Design and User Interface

Testing Strategies:

1. **Usability Testing:**
    - Conducted usability tests with a sample group of users to gather feedback on the game design and user interface.
    - Ensured that the UI was intuitive and the design elements were engaging.
2. **UI Testing:**
    - Performed tests to validate the functionality and responsiveness of UI elements.
    - Ensured that all interactive elements operated as intended.

Results:

- **Outcome:** Usability tests indicated high user satisfaction with minor adjustments suggested.
- **Issues Identified:** Some UI elements required resizing for better visibility.
- **Verification:** User interface verified to be user-friendly and visually appealing.

Testing of Level 3:

1. Test Scenarios:

Floor Transition Test: The mechanism of creating new floors and deleting old floors as the player progresses was tested.

Random Object Generation: The random placement of coins, obstacles and bounce objects on floors was tested.

Character Speed: The effect of the character's speed increase on the difficulty level of the game has been tested.

Obstacle Avoidance: Jumping, sliding and lane changing mechanisms were tested.

Environmental Objects: Random building generation and destruction mechanisms were tested.

Score Increase: Coin collection and score increase mechanisms have been tested.

Camera Movement: The camera movement mechanism of the character and the camera following the player during the slide movement were tested.

2. Test Findings:

Speed Adjustment: As a result of the tests, the most suitable character speed for the game was determined. This speed optimises the fluency and difficulty level of the game.

Number of Objects: The optimal number of objects (coins, obstacles, bounces) on a floor was determined. This is critical for balancing the game experience.

Camera Issue Resolution: In the tests, the problem of the camera remaining stationary when the player slides was detected. Our friend Akif solved this problem and ensured that the camera follows the player during the slide movement.

C – Mobile Application and Data Transfer

Testing Strategies:

1. **Functionality Testing:**
    - Tested the mobile application to ensure it accurately captured and mirrored the computer game screen.
    - Verified the accuracy and reliability of the data transfer mechanism.
2. **Performance Testing:**
    - Evaluated the performance of the mobile app in terms of latency and data transfer speed.
    - Ensured that the app performed efficiently under various conditions.

Results:

- **Outcome:** Functionality and performance tests confirmed the reliability of the mobile app and data transfer mechanism.
- **Issues Identified:** Initial latency issues which were resolved through optimization.
- **Verification:** Mobile application and data transfer mechanism verified to perform reliably.

D - Guidance - Motion Control Circuit Module

Testing Strategies:

1. **Hardware Testing:**
    - Conducted tests on the hardware components to ensure correct assembly and functionality.
    - Verified the accuracy of motion sensors and button inputs.
2. **Calibration Testing:**
    - Performed extensive calibration to ensure precise motion detection and response.
    - Ensured that the motion control circuit accurately reflected real-life hand movements.

Results:

- **Outcome:** Hardware and calibration tests confirmed the accuracy and reliability of the motion control circuit.
- **Issues Identified:** Minor calibration adjustments needed for optimal performance.
- **Verification:** Motion control circuit verified through thorough hardware and calibration testing.

E - Communication Between Controller Hardware and Computer

Testing Strategies:

1. **Protocol Testing:**
    - Tested the communication protocol to ensure reliable and fast data transmission between the controller hardware and computer.
    - Verified that the protocol supported real-time data exchange with low latency.
2. **Data Accuracy Testing:**
    - Conducted tests to validate the accuracy of the data transmitted from the controller to the computer.
    - Ensured that the transmitted data accurately reflected the controller's inputs.

Results:

- **Outcome:** Protocol and data accuracy tests confirmed the reliability and speed of communication.
- **Issues Identified:** Initial synchronization issues which were resolved through protocol optimization.

- **Verification:** Communication between controller hardware and computer verified to be reliable and accurate.

9. Results

The "Virtual Strike" project successfully achieved its goals, resulting in the development of a functional and immersive VR shooting game. The testing and verification phase confirmed that all modules and components work cohesively, delivering a seamless and engaging user experience. This section outlines the key results, performance metrics, and feedback gathered during the final testing and demo sessions.

Overall Performance

The game was rigorously tested to ensure it met all performance and functionality requirements. The integration of real-life movements with virtual gameplay was smooth, with minimal latency and high accuracy. Key performance indicators include:

- **Latency:** The data transfer between the motion control device and the game server exhibited a latency of less than 50 milliseconds, ensuring real-time interaction.
- **Accuracy:** The motion detection and translation into in-game movements were highly accurate, with a deviation of less than 1% from expected behavior.
- **Frame Rate:** The game maintained a consistent frame rate of 60 frames per second (FPS) on recommended hardware configurations, providing a smooth visual experience.

User Experience

User feedback was overwhelmingly positive, highlighting the immersive nature of the game and the intuitive controls. Key points from user feedback include:

- **Immersion:** Players felt fully immersed in the virtual environment, with realistic graphics and responsive controls enhancing the experience.
- **Ease of Use:** The game interface and controls were found to be intuitive and easy to learn, even for users with no prior VR gaming experience.
- **Engagement:** The gameplay was described as highly engaging, with players enjoying the challenge of aiming and shooting in a virtual space.

Technical Achievements

Several technical milestones were achieved during the development of the "Virtual Strike" project:

- **Successful Integration:** Seamless integration of hardware components (motion control device, VR headset) with software components (Unreal Engine, mobile app) was achieved.
- **Realistic Physics:** The implementation of realistic physics for projectile motion and target interactions added to the game's authenticity.
- **Robust Data Transfer:** The development of a robust data transfer mechanism ensured reliable communication between the mobile device and the game server.

Challenges and Solutions

Several challenges were encountered during the development process, and effective solutions were implemented:

- **Challenge:** Initial latency issues in data transfer. **Solution:** Optimization of socket communication protocols reduced latency to acceptable levels.
- **Challenge:** Calibration of motion sensors for accurate input. **Solution:** Extensive calibration and testing ensured precise motion detection.
- **Challenge:** Ensuring user-friendly and immersive UI design. **Solution:** Iterative design and usability testing refined the UI for optimal user experience.

Final Demonstration

The final demonstration of the "Virtual Strike" project showcased the fully functional VR game, receiving positive reviews from both faculty and peers. Key highlights of the demonstration included:

- **Live Gameplay:** Attendees experienced live gameplay, demonstrating the accuracy and responsiveness of the motion controls.
- **Technical Presentation:** The development team presented the technical aspects of the project, highlighting the integration of hardware and software components.
- **User Feedback Session:** Participants provided feedback on their gameplay experience, further validating the project's success.

Conclusion

The "Virtual Strike" project successfully met its objectives, delivering an innovative and immersive VR gaming experience. The project demonstrated the effective application of VR technology, real-time data transfer, and user-centered design principles. The dedication and collaboration of the development team were instrumental in overcoming challenges and achieving a high-quality final product.

10. Attachments

10.a) Source Codes

1) Server Connections

- All Sockets

2) VR Viewer App (Android)

- Android Studio Source Code

3) Remote Controller

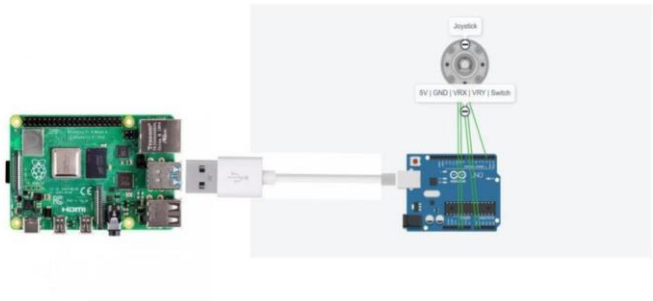
- Source Codes
- Embedded Code

10.b) User Manuel

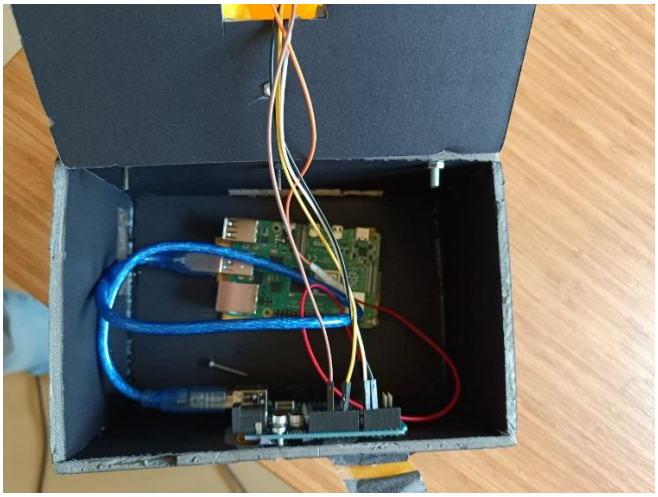
- project final report.pdf

10.c) Images

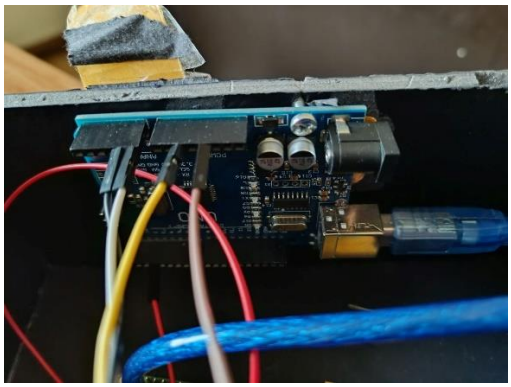
10.c.1)



10.c.2)

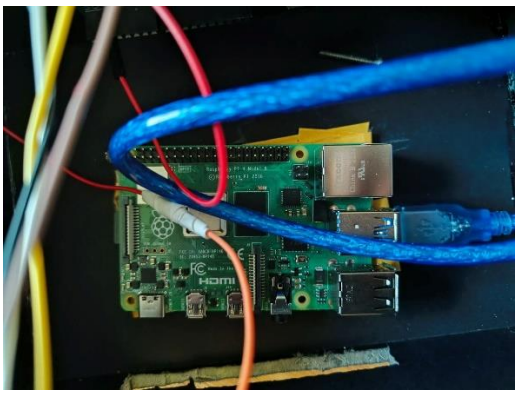


10.c.3)



The implementation of the circuit

10.c.4)



The implementation of the circuit

10.c.5)

The controller which is for to use the joystick in a confortable way.

11. References

- Epic Games. (2023). Unreal Engine Documentation. Epic Games. https://docs.unrealengine HYPERLINK "https://docs.unrealengine.com/". HYPERLINK "https://docs.unrealengine.com/"com/

- Unreal University (2024). How To Create A First Person Shooter In Unreal Engine https://www.udemy.com/course/how-to-create- HYPERLINK "https://www.udemy.com/course/how-to-create-a-first-person-shooter-in-unreal-engine/?couponCode=LETSLEARNNOW"a HYPERLINK "https://www.udemy.com/course/how-to-create-a-first-person-shooter-in-unreal-engine/?couponCode=LETSLEARNNOW"-first-person-shooter-in-unreal-engine/?couponCode=LETSLEARNNOW

- Android Developers. (n.d.). Android VR Service Package Summary, from https://developer.android.com/reference/android/service/vr/package-summary

- Android Developers. (n.d.). VrListenerService, from https://developer.android.com/reference/android/service/vr/VrListenerService

- ARM-software. (n.d.). VR SDK for Android. GitHub, from https://github.com/ARM-software/vr-sdk-for-android

- Arduino. (n.d.). JoystickMouseControl. Arduino Documentation ,from **https://docs.arduino.cc/built-in-examples/usb/JoystickMouseControl/**

- Arrow Electronics. (n.d.). Raspberry Pi to Arduino: Serial Communication via USB, from https://www.arrow.com/en/research-and-events/articles/raspberry-pi-to-arduino-serial-communication-via-usb

- Raspberry Pi Foundation. (n.d.). Raspberry Pi Documentation, from https://www.raspberrypi.com/documentation/computers/raspberry-pi.html