CSE 396 – Computer Engineering Project

Gebze Technical University

Spring 2024

Virtual Strike Project – Project Scenario 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. Introduction

Welcome to the Scenario Document for the Virtual Strike VR game, developed by Group 4 of VS Games at Gebze Technical University. This document is designed to guide you through a series of detailed scenarios that depict typical interactions between users and the Virtual Strike game system. These scenarios will provide insights into the functionality of the game, ensuring a clear understanding of user expectations and system responses under various conditions.

Project Purpose

Virtual Strike is an innovative VR shooting game that aims to deliver a realistic and engaging experience by mimicking real-world shooting mechanics through advanced VR technology. The game integrates physical activity with virtual environments, making it not only an entertainment medium but also a potential tool for physical exercise and skill development. Using Unreal Engine, VR headsets, and a combination of Android mobile devices and custom-built controllers, the game offers a seamless blend of real and virtual worlds where precision, reaction time, and strategic thinking play crucial roles.

Audience

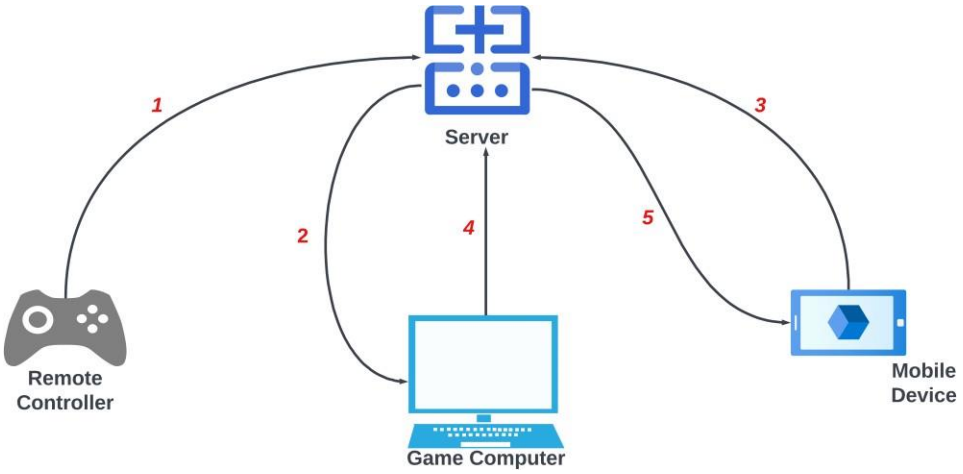
This document is intended for internal use by the project team, including developers and testers, as well as external stakeholders such as project sponsors, academic supervisors, and potential investors who may benefit from an in-depth understanding of the game's operational scenarios.

Document Overview

The following sections will outline various user scenarios, including game setup, gameplay interactions, system troubleshooting, and error handling. Each scenario is described with its preconditions, basic flow, postconditions, and exception paths to cover the full range of functionalities and potential issues. This structured approach ensures that every aspect of the game experience is meticulously planned and ready for implementation and testing.
By the end of this document, readers will have a comprehensive view of how Virtual Strike operates from both a user and a technical perspective, facilitating better decision-making and project outcomes.

3. 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.

4.Scenario Desciptions

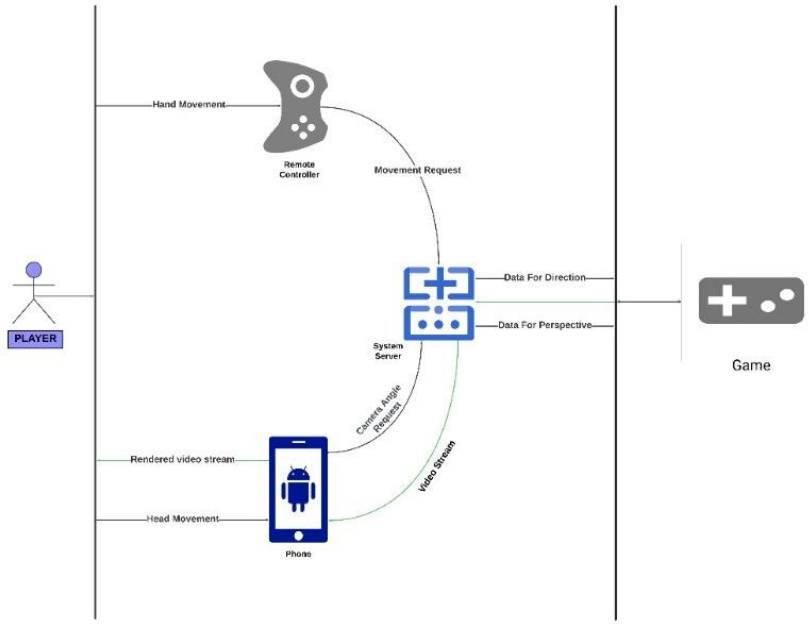


**Diagram to summarize the general data flow that make the game playable*

Scenario 1: Game Initialization and Setup

- **Title:** Starting the Virtual Strike Game
- **Description:** This scenario describes the comprehensive process that players undergo to start a new game session, from system setup to the beginning of gameplay. This includes ensuring all hardware is correctly configured and selecting game settings.
- **Actors:** Player
- **Preconditions:**
    - The player has the game installed on their system.
    - The mobile phone is plugged to the VR headset and the remote controller is functional, connected each other and properly configured.
    - The player's gaming environment is set up with adequate space and minimal obstructions.
- **Basic Flow:**

1. **System Check:**
    - The player checks that the game screen successfully connected to mobile phone and remote controller are connected to the computer.
    - Both remote controller and mobile phone connected to the internet and mobile phone sends sensor data successfully.
    - The player verifies that the game software and hardware drivers are up to date.
2. **Game Start:**
    - The player starts the game from the computer and wears the VR headset and the game starts.

3. **Lobby Entry:**
    - The game loads the lobby environment, a virtual space where players can choose from various game modes. There are 3 doors to go in. Which teleports the user to these modes:
      1) Polygon.
      2) Game Mode 1.
      3) Game Mode 2.
4. **Mode Selection:**
    - The player selects the game mode by entering the door named with that mod.
5. **Game Initialization:**
    - Once the mode are confirmed, the game loads and the player finds themselves in the game, the game placing the player in the starting position, player can pick up the gun and shoot to the start game button and start the game.
6. **During Game:**
    - When the player started the game a countdown starts according to the game mode. And the game flows. Player earns or loses points according to their gameplay. Player can see how many points they earned or lost. When the countdown finishes the final point is shown to the player. And the player can play again or exit the game mode.
7. **Exit Game:**
    - Player can exit the mode or exit the game. If the player is in the lobby there is a button or a door in the lobby for player to exit from the game. But if the user has entered a mod they can go to the exit mode door in the mod platform and go back to the lobby.

- **Postconditions:**
    - The player is fully immersed in the game environment, ready to engage with the gameplay as per the chosen settings.
- **Exception Paths:**
    - **Game Load Failure:**
        - If the game fails to load (due to hardware issues, software crashes, etc.), an error message is displayed.
        - The player is prompted to check their hardware connections or restart the game.
        - If issues persist, the player is advised to reinstall the game or update their hardware drivers.
    - **Hardware Malfunction:**
        - If the VR headset or controller malfunctions, the game pauses, and troubleshooting tips are provided.
        - The player is guided through basic hardware checks or directed to customer support.
- **Assumptions:**
    - The player has a basic understanding how to connect the necessary equipments each other.
    - All game and hardware settings are optimized for performance to prevent lag or motion sickness.

Scenario 2: Interaction During Gameplay

- **Title:** Engaging with Game Targets
- **Description:** This scenario outlines how players interact with dynamically spawning targets within the game environment, utilizing the VR system's shooting mechanics.
- **Actors:** Player
- **Preconditions:**
    - The game has been started and is in active gameplay mode.
    - The player is equipped with the VR headset and controller, both functioning correctly.
- **Basic Flow:**

1. **Target and Ally Spawning:**
    - Targets appear at random locations within the virtual environment, simulating a real-world shooting range experience.
    - But there is not only targets in the game. There are allies too. Which we shouldn't shoot them, if we do we lose points.
    - Allies spawn close to the targets so that the player should be careful to shoot to target.
2. **Aiming and Shooting:**
    - The player uses the the VR headset to aim by looking around to aim at the targets. And uses the controller to move around and press the button on the controller to shoot.
    - The player aligns the sight of the virtual gun with a target.
3. **Trigger Mechanism:**
    - The player presses the trigger button on the controller to fire at the target.
    - The game detects the trigger press and releases the ammo which is cute balls.
4. **Hit Detection and Scoring:**
    - The game uses specific algorithms to determine if a shot hits a target.
    - Successful hits on enemy targets increase the player's score, while hits on ally or civilian targets result in penalties.
5. **Feedback to Player:**
    - Immediate audio and visual feedback is provided. For a hit, the player hears a satisfying sound effect and sees the target react (e.g., shattering or falling over).
    - For a miss, a different sound cue and visual effect like the bullet hitting a wall or the ground are shown.
6. **Score Update and Continuation:**
    - There is a countdown mechanism in the game. When game starts the countdown starts too. When the countdown finishes the game is over.
    - The player's score is updated in real-time and displayed on the game screen constantly.
    - New targets continue to spawn, maintaining the gameplay flow until the countdown finishes.
    - The player can play again and see what is the highest score in the game in the game screen.

- **Postconditions:**
    - The player's score is accurately updated according to their successfull target shootings.
    - The game continues to provide new targets, sustaining engagement and challenge levels.
- **Exception Paths:**
    - **Controller Malfunction:**
        - If the controller malfunctions or loses connection, the game automatically pauses.
        - An on-screen message prompts the player to check the controller's connection.
        - The player can attempt to reconnect the controller. If reconnection is successful, the game resumes from the pause state.
        - If the problem persists, the game suggests restarting the controller or replacing its batteries.
    - **Sensor Accuracy Issues:**
        - If the player notices a consistent discrepancy between aimed shots and registered hits, they should recalibrate the controller.
- **Assumptions:**
    - The player has a clear understanding of game rules and shooting mechanics.
    - Environmental and gameplay settings are optimized to prevent any discomfort or motion sickness during play.

Scenario 3: Exiting the Game

- **Title:** Exiting the Game Session
- **Description:** This scenario outlines the steps a player takes to exit the game.
- **Actors:** Player
- **Preconditions:** The player is either in the lobby or actively playing the game.
- **Basic Flow:**
    1. Player can go through the exit game door if they are in lobby or shoot to the exit mod button if they are in a mod.
    2. Player confirms to exit by shooting to confirm button.
    3. The game confirms the player's intent to exit.
    4. Upon confirmation, the game session is safely terminated.
    5. Player removes the VR headset.
- **Postconditions:** The game is closed, and all game data is saved.
- **Exception Paths:** If the game fails to exit properly, force a shutdown and log the incident.

Scenario 4: Handling Data Synchronization Errors

- **Title:** Data Synchronization and Error Handling in Server and Raspberry Pi Interactions
- **Description:** This scenario elaborates on managing synchronization errors that occur during data transmission between the Raspberry Pi (handling joystick movements), the mobile application (handling phone direction data), and the server, which is critical for real-time gameplay integration.
- **Actors:** Game Server, Mobile Application, Raspberry Pi, Player's Computer
- **Preconditions:**
    - The game is actively being played with continuous data exchanges involving the Raspberry Pi (joystick movements), the mobile application (phone direction), and the game server.
    - The player's mobile device, Raspberry Pi, and the server initially have stable network connections.
- **Basic Flow:**

1. **Error Detection:**
    - Both the Raspberry Pi and the mobile app monitor the integrity and continuity of data sent to the server.
    - If delays or interruptions in data transmission exceed acceptable limits, an error state is triggered, automatically pausing the game.
2. **User Notification:**
    - An error message is displayed within the VR environment, indicating the nature of the problem ("Connection to server lost. Check your network settings or device connections.").
    - Players are presented with options: "Reconnect" or "Exit Game."
3. **User Decision:**
    - The player chooses either to attempt reconnection or to exit the game via the VR controller.
4. **Reconnection Attempt:**
    - If "Reconnect" is chosen, the system attempts to re-establish connections from the Raspberry Pi and the mobile app to the server.
    - The game displays a countdown during reconnection attempts, updating the player on the status ("Reconnecting in X seconds...").
5. **Outcome Handling:**
    - If reconnection is successful, the game resumes from the paused state with all gameplay data synchronized and up-to-date.
    - If reconnection attempts fail after a specified number of retries, the game offers the player the option to save the game and exit or wait longer.

- **Postconditions:**
    - If reconnected, gameplay continues seamlessly with all player progress and real-time input data intact.
    - If not, the player can choose to exit the game after saving their progress, or continue attempting to reconnect.
- **Exception Paths:**
    - **Intermittent Connectivity Issues:**
        - If intermittent connectivity issues are detected, the game reduces synchronization frequency and notifies the player of potential gameplay impacts.
    - **Device Malfunction (Raspberry Pi or Mobile):**
        - If a device-specific issue (Raspberry Pi or mobile) is suspected, the game provides troubleshooting steps specific to the device (e.g., check connections, restart the device).
    - **Server Unavailability:**
        - If the server becomes unavailable, the warning is shown to player to check the devices and the server and reconnect them each other.

    - **Data Corruption Concerns:**
        - If there is a potential data corruption during reconnection, the system checks the integrity of data received. If corruption is detected, the player is prompted to restart the game from the last saved checkpoint.
- **Assumptions:**
    - The player's devices (Raspberry Pi, mobile app) are configured to handle auto-reconnection attempts.
    - Server-side mechanisms are equipped to handle and resolve multiple, simultaneous reconnection attempts without data loss.