AR Agility Trainer Project Report

Submission by Sanchay Baranwal (220101122)

1. Introduction

This project presents the development of an Augmented Reality (AR) Agility Trainer designed to improve rapid movement, reaction speed, and perception-action coordination across a wide range of users. Built using Unity and AR Foundation, the system places virtual cones into the user's real environment, turning any safe open area into a personalized training space that supports both physical and cognitive engagement.

To enable progress tracking and competitive interaction, the trainer integrates Firebase for user data storage, performance history, and an asynchronous multiplayer mode. The platform is further extended through a Python-based data analytics pipeline capable of analysing sports footage to generate realistic movement patterns, offering an optional advanced mode based on real-world motion, which would be exceedingly useful for people training in professional sports.

Objective: To provide a highly adaptive, measurable, and accessible training experience that can support motor skill development, cognitive-motor training, and general physical activity beyond the scope of traditional drills.

2. Game Overview and Supported Modes

The AR Agility Trainer is designed as a modular, extensible AR-based movement training system built on Unity + AR Foundation. The trainer supports **three core modes**, each addressing a different user need:

2.1 Personal Training Mode

A single-player, adaptive session where:

- The player taps virtual cones spawned in a 4-meter radius arena.
- Difficulty automatically adapts based on recent performance (coneTimeout adjustment).
- Players may optionally enable **Sports Pattern Mode**, where real-world movement sequences—extracted from sports footage via the Python ML pipeline—define the cone placements.
- Scores and difficulty metrics are recorded to Firebase for long-term tracking.

2.2 Challenge Mode (Asynchronous Multiplayer)

A competitive mode where:

- Player 1 **creates a challenge** → receives a unique 5-digit code.
- Player 2 **joins the same challenge using the code**.
- Firebase synchronizes:
  - A **shared game seed**
  - Challenge state (**waiting –> active –> completed**)
- Both players receive **identical cone sequences**, difficulty settings, and arena parameters.
- Each player plays independently; scores are uploaded upon completion.

2.3 User Stats Mode

A historical dashboard that:

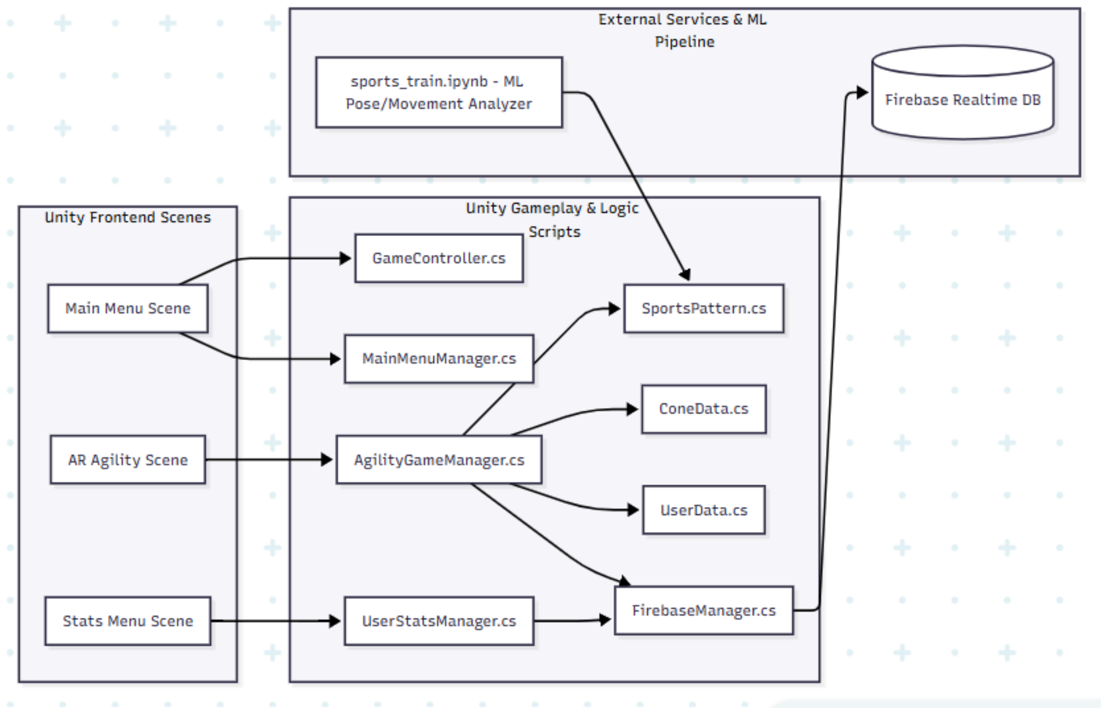
- Fetches the **last 20 training sessions** from Firebase (score, difficulty, timestamp).
- Displays the player's progression and trend over time.
- Enables data-driven improvement and self-assessment.

3. System Architecture

The following component diagram visualizes the **Unity scene structure**, **C# managers**, **Firebase backend**, and the **external ML pipeline** that form a crucial part of the AR Agility Trainer application..

This diagram highlights:

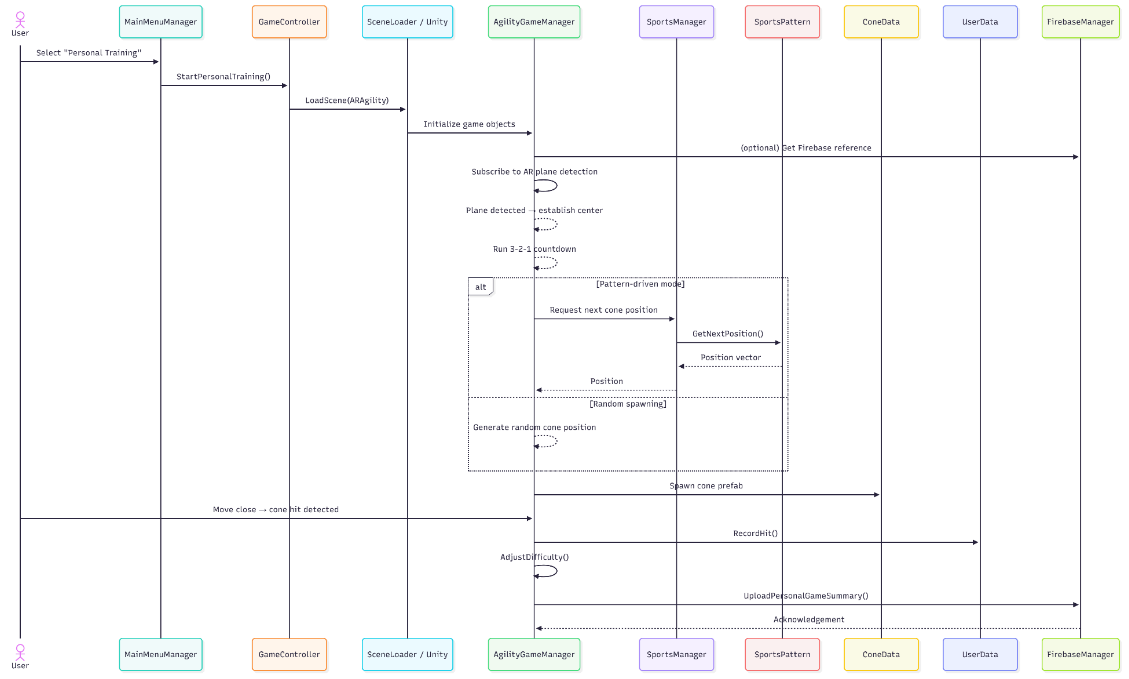
- The relationship between **MainMenu, GameController** and **Gameplay Managers.**
- How **SportsPattern** plugs into the adjust difficulty logic when patterns analysed by the sports training scripts are used
- How **SportsTrainer.cs and Python YOLO pipeline** feed real-world rally data into the Unity system
- How **FirebaseManager** handles authentication, game history, and challenge-mode synchronisation.



4. Logic Flow

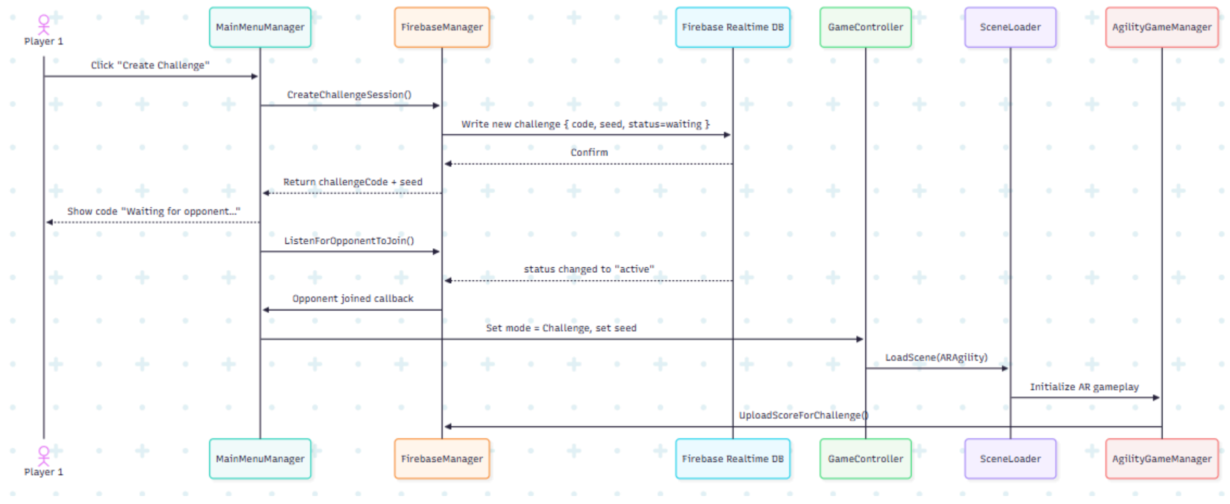
This section illustrates the runtime behavior of the system through sequence diagrams that capture **personal training**, **multiplayer challenge**, and **stats** workflows.

4.1 Start Training



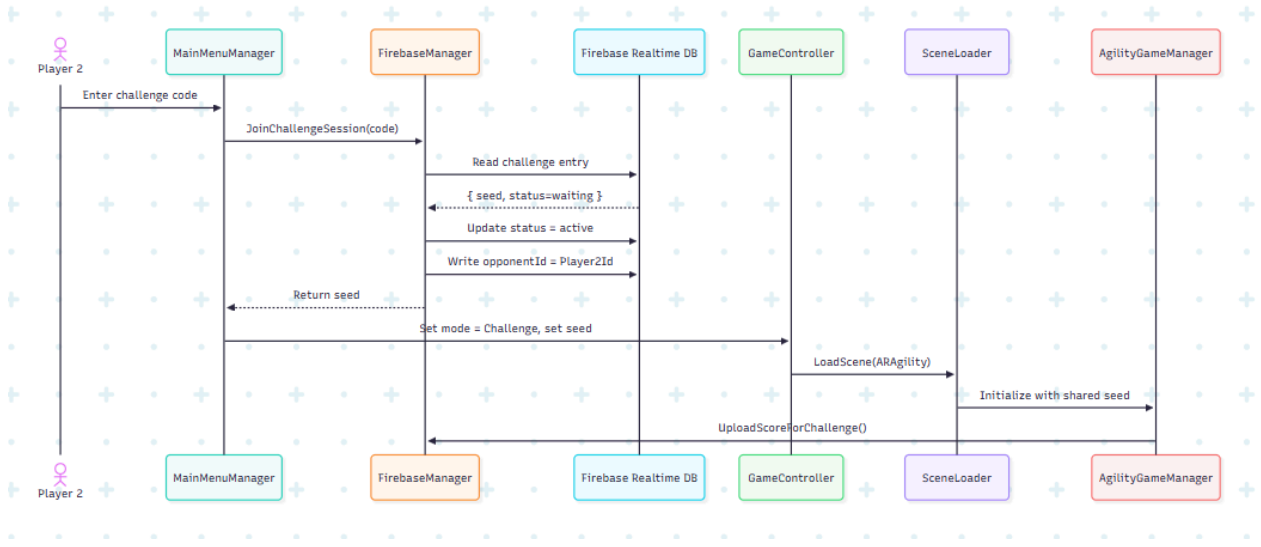
The user starts Personal Training from the main menu. The game loads the AR scene, detects a plane, sets a center point and begins a countdown. Cones are then spawned either randomly or using a sports pattern based on the SportsPattern logic. Each hit updates UserData, which AgilityGameManager uses to adjust difficulty. When the session ends, results are uploaded to Firebase via the FirebaseManager.

4.2 Create Challenge



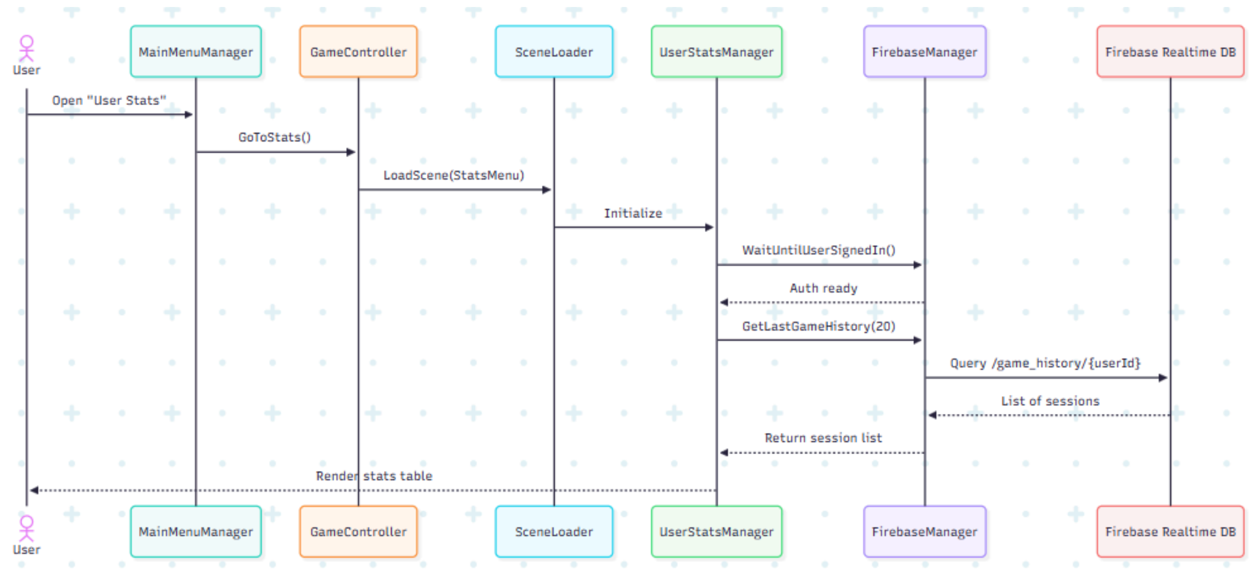
Player 1 creates a challenge. FirebaseManager writes a challenge entry in the database and returns a code. The menu listens for an opponent joining, the game does not start until the opponent joins while the player has the option of cancelling the challenge; once the Firebase status updates to **active**, the game starts using the shared seed so both players get identical cone sequences. After the game, the score for Player 1 gets uploaded.

4.3 Join Challenge



Player 2 enters the challenge code shared to them by Player 1. FirebaseManager verifies the challenge and also avoids more than one player entering the session using the challenge code, marks it **active**, writes Player2 as the opponent, and returns the shared seed. Player 2 loads the AR scene and plays the identical cone sequence. At the end of the session, their score is uploaded.

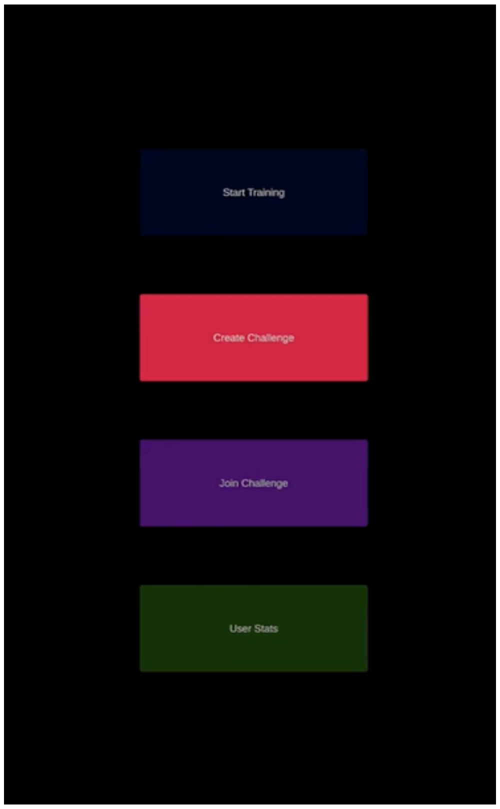
4.4 User Stats



When the user opens the Stats menu, UserStatsManager ensures Firebase authentication is ready. It then asks FirebaseManager for the last **20 game entries**, receives them, and renders a simple history table in the UI.

5. Screens and Interface Flow

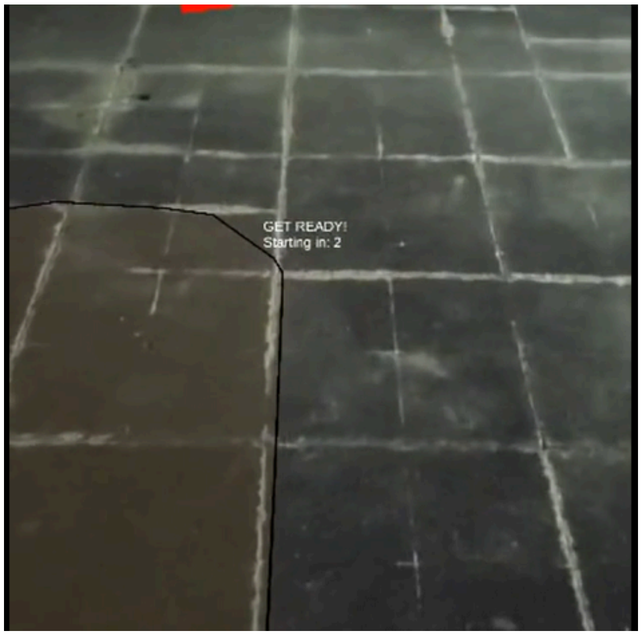
5.1 Main Menu Screen

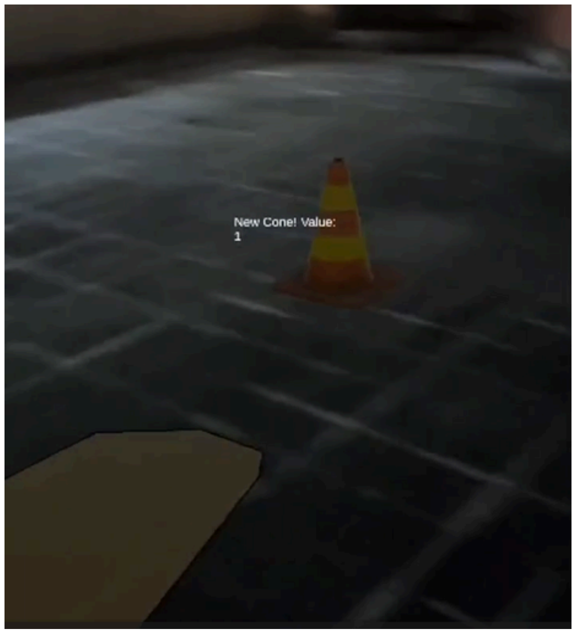


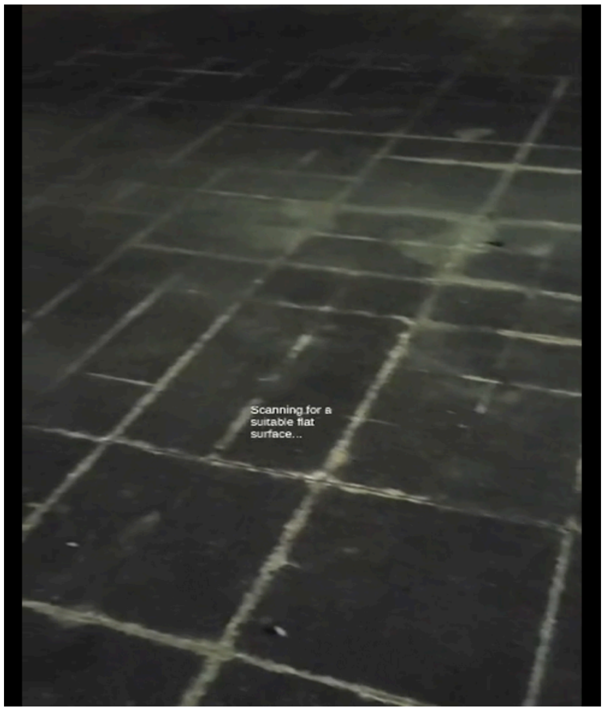
The Main Menu screen acts as a central hub with four main buttons: Personal Training, Create Challenge, Join Challenge, User Stats. It handles the modal pop-ups upon challenge creation and joining the challenge.

5.2 AR Game Screen

It displays all the gameplay elements including the AR plane detection phase, the 3-2-1 countdown, the spawning of cones at locations determined randomly or by logic contained in the SportsPattern file. It also includes a dynamic score/timer display, and a final scorecard panel upon the completion of the game.

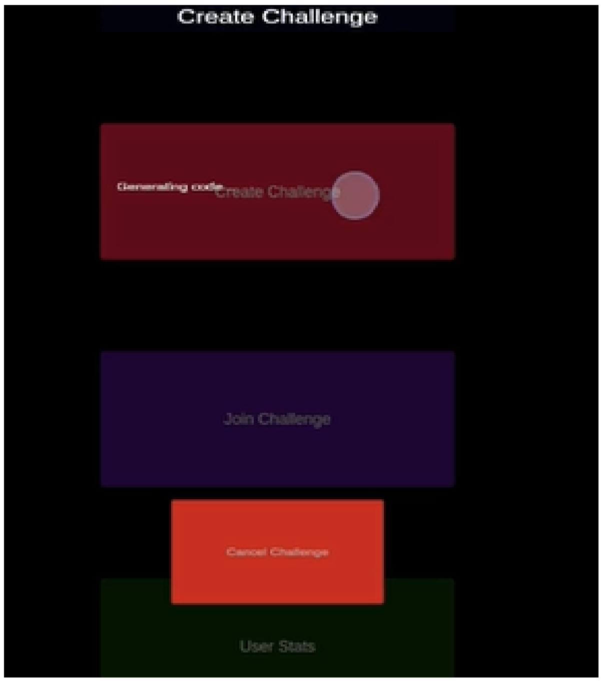

GET READY!
Starting in: 2


New Cone! Value:
1

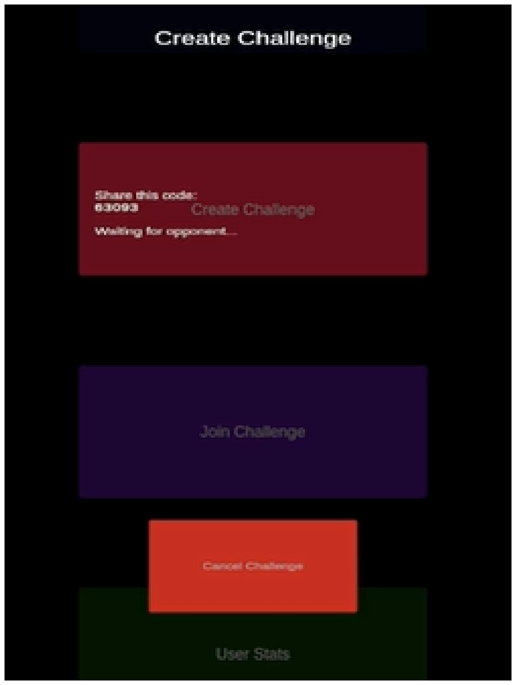

Scanning for a suitable flat surface...

5.3 Create Challenge Popup

It displays when a player (Player 1 in our challenge) chooses to create a challenge, displaying a newly generated 5-digit challenge code. This panel remains visible while the system waits for an opponent to join, indicating the “Waiting for opponent...” state. A Cancel button is provided so that the player may exit the challenge setup at any time. The interface updates in real time as Firebase listeners detect when another player successfully joins the challenge session.



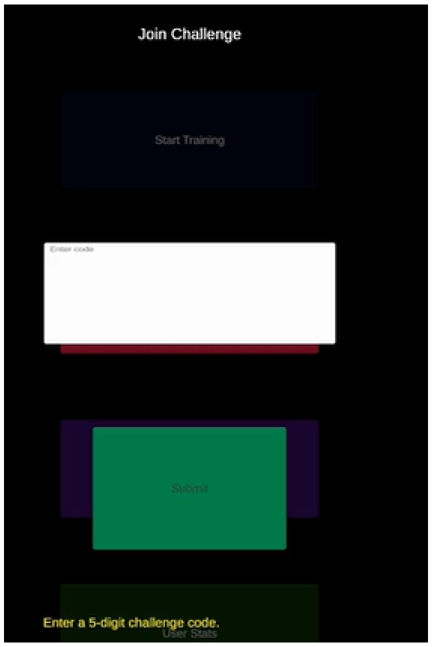
Generating code...



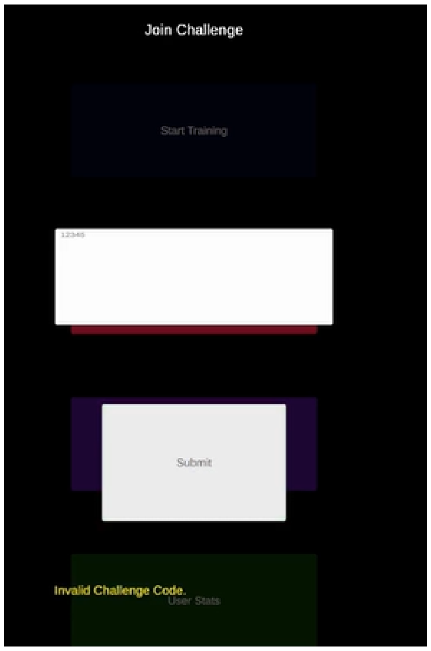
Share this code:
63093
Waiting for opponent...

5.4 Join Challenge Popup

When Player 2 selects the option to join a challenge, the popup panel prompts them to enter the 5-digit code shared by Player 1. As soon as the code is submitted, the system provides live status feedback such as “Verifying code...,” followed by either “Invalid code” if the code cannot be matched, or “Success! Loading game...” once the session is validated. After successful verification, the player is automatically transitioned into the gameplay scene using the shared game seed.



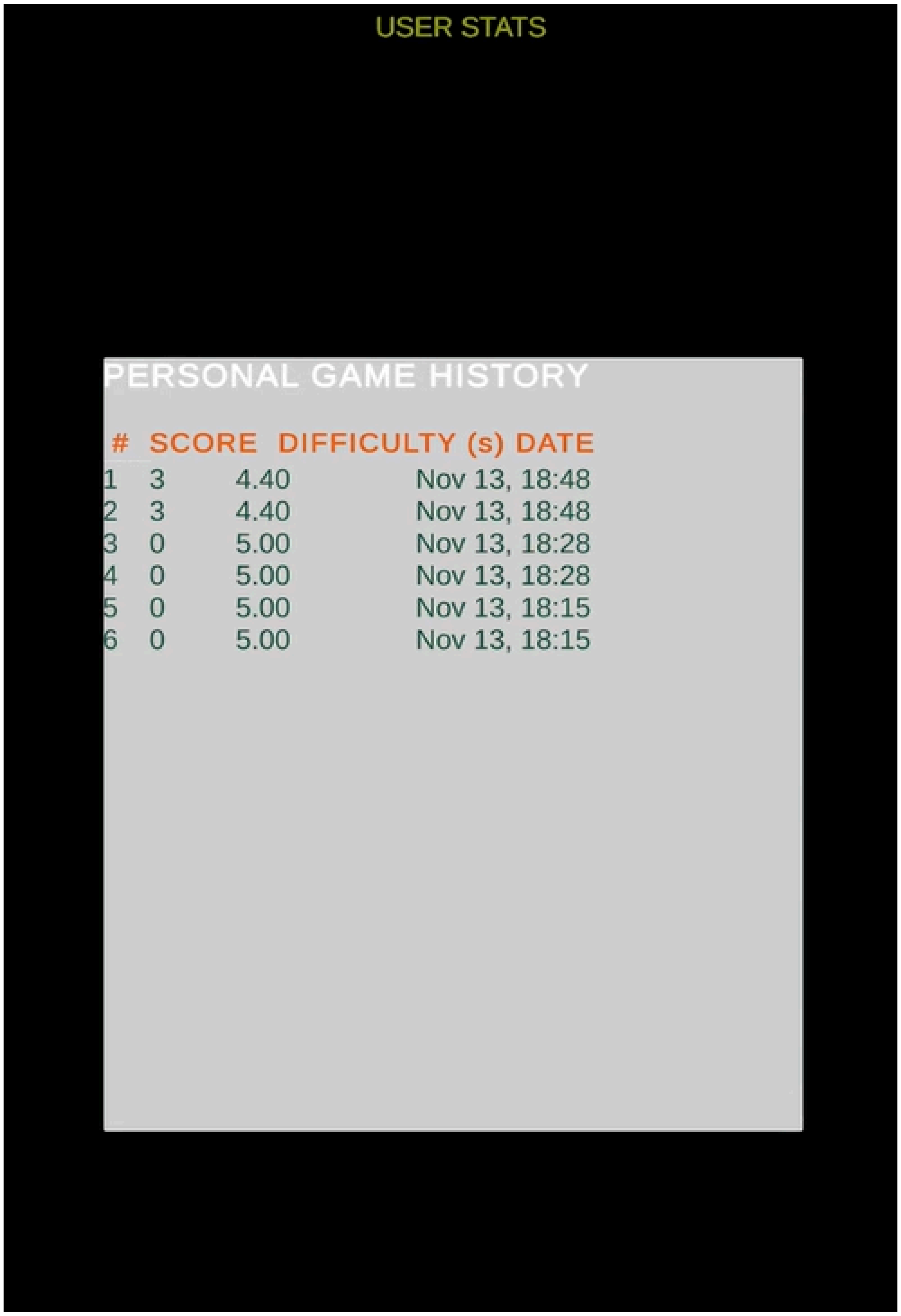
Enter a 5-digit challenge code.



Invalid Challenge Code.

5.5 User Stats Screen

It displays the list of the last 20 game sessions fetched from Firebase, formats the data (Game #, Score, Difficulty, Date), and renders it to a TextMeshProUGUI component in a clear, tabular format.



PERSONAL GAME HISTORY

#	SCORE	DIFFICULTY (s)	DATE
1	3	4.40	Nov 13, 18:48
2	3	4.40	Nov 13, 18:48
3	0	5.00	Nov 13, 18:28
4	0	5.00	Nov 13, 18:28
5	0	5.00	Nov 13, 18:15
6	0	5.00	Nov 13, 18:15

6. Code Elements

6.1 Unity (C#) & AR Integration

The core application is developed in Unity using **AR Foundation**, which provides cross-platform AR functionality such as plane detection, tracking, and world-space alignment. The runtime is managed through a small set of coordinated C# scripts:

GameController.cs (Singleton)

- Maintains global, persistent game state (e.g., GameMode –> PersonalTraining or Challenge).
- Stores and carries the currentGameSeed across scene loads.
- Provides a stable entry point for starting gameplay scenes.

AgilityGameManager.cs

- Initializes the AR session and establishes the arena center once a plane is detected.
- Handles the 3-2-1 countdown logic and prepares the gameplay loop.
- Manages cone spawning (random or pattern-based), hit detection, timer updates, and scoring.
- Implements adaptive difficulty:
    - Read recent hit timings via UserData.GetAllHits().
    - Adjusts coneTimeout using a weighted average of recent performance.

- Triggers end-game logic and summary display.

SportsPattern.cs

- Loads JSON movement sequences produced by the Python ML pipeline.
- Exposes a simple API (GetNextPosition()) returning Unity Vector3 positions.
- Allows Personal Training sessions to mimic real sports rally movements.

UserData.cs

- Stores hit timestamps locally for use in difficulty adjustment.
- Acts as lightweight persistent storage for per-session performance trends.

ConeData.cs

- Attached to cone prefabs.
- Holds mutable properties such as each cone’s score value (based on distance).

UserStatsManager.cs

- Fetches the last 20 saved game summaries from Firebase.
- Formats and displays the data in the Stats scene.

MainMenuManager.cs

- Drives the UI for all main menu actions:
    - Personal Training
    - Create Challenge
    - Join Challenge
    - Viewing Stats
- Manages popup panels for challenge creation and challenge code entry.

6.2 Firebase Realtime Database Integration

All backend communication with Firebase that includes authentication, session history, and multiplayer challenge handling is handled by **FirebaseManager.cs**.

Key Responsibilities

Authentication

- Uses Firebase Anonymous Authentication.
- Provides each user a persistent unique ID without requiring sign-up.

Data Storage Structure

- **/game_history/{userId}/{autoId}**
    - Stores results for each training session
    - Includes score, difficulty level, timestamp
    - Used by both the Stats screen and the Adaptive Difficulty logic
- **/challenges/{challengeCode}**
    - Manages the Challenge mode
    - Stores:
        - 5-digit challenge code
        - Shared game seed
        - Status (waiting, active, completed)
        - Player results

Multiplayer Handshake

- **CreateChallengeSession():**
    - Generates a code, writes metadata to Firebase and sets the session to waiting.
    - Listens for opponent join.
- **JoinChallengeSession():**
    - Validates code.

    - Marks session as `active`, prevents more than two players from joining.
    - Returns the shared seed used for identical cone sequences.

- Real-time listeners ensure both players start with the same seed and synchronized settings.

6.3 Data Analytics Extension: Sports Pattern Generation

This extension enables the game to provide drills modeled on **real athlete movement**, extracted from video.

Python Analysis Script (sports_train.ipynb)

This uses computer vision and pose estimation (YOLO/Ultralytics) to analyze sports footage such as badminton or tennis rallies.

Input

- Any MP4 video file (including YouTube downloads via `yt-dlp`).
- The player and ball/shuttlecock appear in the frame.

Processing Steps

- Detects player's pose keypoints across frames.
- Tracks movement vectors and body orientation.
- Computes wrist velocity spikes to detect swings.
- Segments rally into sequences based on continuous motion.
- Extracts step direction and body orientation trends.

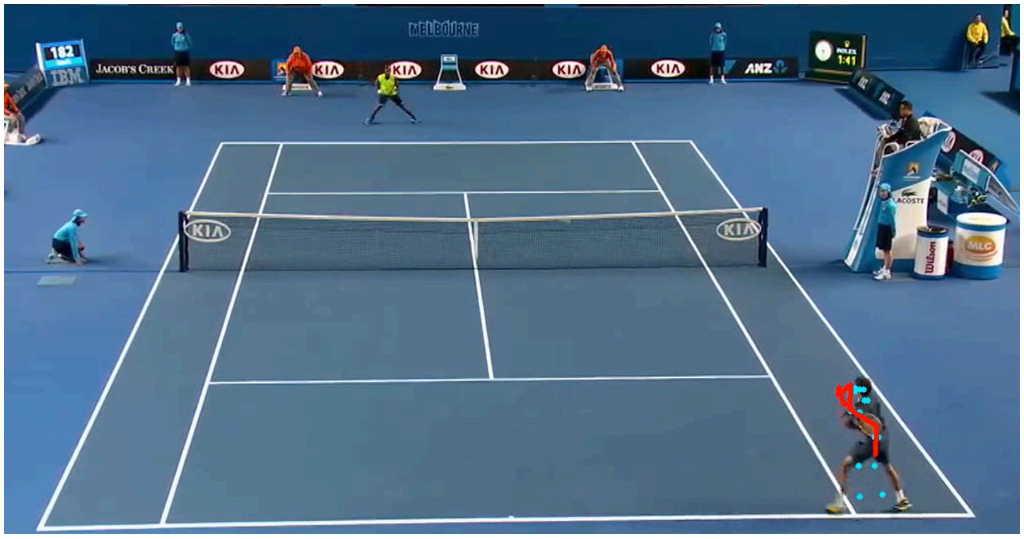
Output

- A structured JSON file containing:
    - Time-stamped movement frames
    - Extracted rally segments
    - Smoothed movement paths
    - Swing markers

This JSON represents real-world movement paths that can be converted to Unity target positions.

SportsPattern.cs (Unity Integration)

- Reads the generated JSON and converts it into ordered 3D cone targets.
- Allows AgilityGameManager to use pattern-based training modes.
- Ensures the player follows realistic rally-inspired movement sequences.



Example output from the movement pattern analyser pipeline, illustrating automated movement tracking during a tennis rally. It detects key joint coordinates (cyan markers), estimates limb orientation, and reconstructs the player's movement trajectory (red trace), which is later converted into spatial training patterns for integration into the AR Agility Trainer.

7. Future Scope and Applied Research Validation

The AR Agility Trainer is more than a technical demonstration, its core training features such as adaptive difficulty, fast perception-action responses, and physically meaningful movement match well with what research has shown to be effective in developing motor skills across different age groups. The following subsections explain how the system fits naturally within established evidence on motor learning in both children and older adults.

7.1 Lifespan Motor Skills and Cognitive-Motor Training

The adaptive architecture of the training system, particularly the adaptive difficulty logic within **AgilityGameManager.cs**, enables precise control over both cognitive load (reaction time pressure) and physical load (locomotor displacement). This dual-load design mirrors the structure of interventions known to enhance motor and cognitive development across the lifespan.

Motor Skill Development in Children

Large-scale reviews have shown that **exergames and AR-based motor activities significantly improve locomotor competence and fundamental motor skills (FMS) in children** [1][2]. These benefits extend beyond physical performance, improving **cognitive flexibility, attentional control, and motivation**, which are key components of perceptual-motor integration [1].

The AR Agility Trainer emphasizes rapid directional changes, acceleration, and visually cued action selection which are all validated features of high-impact FMS intervention, thereby it extends naturally to evidence-based pediatric motor training paradigms.

Cognitive-Motor Maintenance and Fall-Prevention in Older Adults

There is strong clinical evidence to support that AR and mixed-reality exercises lead to improvements in **balance, gait stability, muscle coordination**, and reductions in fall risk among adults over 65 [3][4]. Research meta-analyses of AR and mixed-reality exercise programs show clear improvements on the Berg Balance Scale (BBS), with effect sizes of about 0.56, which indicate significant clinical benefits [3].

The Agility Trainer's requirement for controlled, multi-directional movement in a monitored 4m radius forms a safe analogue to validated dynamic balance training tasks, making it suitable for future exploration in geriatric rehabilitation contexts.

7.2 Mapping Agility, Perception-Action Coupling, and Trainability

The AR agility training system stores granular performance data (reaction times, adaptive difficulty scores, consistency across sessions), thus providing an avenue for non-invasive, longitudinal research into trainability of agility and neuromuscular adaptation.

Agility Across Demographics

Motor performance follows predictable developmental arcs across childhood, adulthood, and older age, with reaction time, cognitive processing speed, and movement variability all showing characteristic age-related trends [5][6]. The system's ability to continuously track difficulty and performance provides a means to **map these trajectories at scale**, offering population-level insights into agility development.

Perception-Action Analysis

Agility is not purely physical, rather it is also strongly affected by **thinking and decision-making skills**, especially stimulus recognition and rapid action selection. Evidence from controlled athletic studies shows that **perception-action training devices significantly outperform traditional drills**, yielding improvements of up to **14–15% in reaction speed** among competitive athletes [7].

Because the AR Trainer uses visual cues, forced-choice stimulus processing, and high-frequency directional changes, it inherently reinforces these validated perception-action pathways. Comparing user performance in **Random cone mode** versus **Pattern-Based mode** (via *SportsPattern.cs*) may offer researchers a clean method for quantifying the cognitive load present in predictable versus unpredictable training stimuli.

Long-term agility development

Systematic reviews confirm that **agility is a trainable quality**, particularly among youth, with responsiveness to both motor and perceptual-cognitive training components [8]. The platform's continuous difficulty adaptation and detailed historical tracking can therefore support research into **how quickly different populations improve**, and which factors (age, sex, prior sport involvement) predict enhanced trainability.

7.3 Technical Development: Advancing the ML-Based Movement Pipeline

The current ML extension, implemented via the Python YOLO/pose-estimation pipeline already extracts meaningful rally-based movement sequences for use in **SportsPattern.cs**. However, several academically supported enhancements can significantly expand the system's research and clinical utility.

Expanding the Training Suite

In addition to lower-limb agility and locomotor tasks, the system can be extended towards rapid hand–eye coordination and upper-limb reflex exercises. These forms of perceptual–motor challenges are often used in cognitive–motor rehabilitation and performance research [8]. By extending the current pose-estimation framework, the system could include fast-reach targets, lateral tapping tasks, and cue-driven upper-limb responses, enabling an AR implementation of established reflex and hand–eye coordination training methods.

Biomechanical Modelling

Although the current pipeline already estimates body joint positions and wrist kinematics, additional biomechanical parameters such as **joint-angle trajectories**, **segmental velocities**, and **movement variability over time** would allow the system to approximate the types of metrics used in clinical gait and motor-control assessment. Research on motor performance across the lifespan highlights how such parameters correlate with developmental changes in coordination, stability, and reaction time [5][6]. Incorporating biomechanical feature extraction would therefore support more fine-grained analysis of movement efficiency and potential injury-risk indicators.

Automated Movement Classification

Deep learning research in sports analytics demonstrates that CNN–RNN and LSTM architectures can robustly segment and classify complex movement patterns in video data [9][10]. Incorporating such models would enable automatic classification of sequences into drill categories (e.g., lateral recovery, defensive footwork), supporting highly personalized training prescription.

Enhanced Filtering and Normalization

The current difficulty-adjustment mechanism employs a window-based heuristic to remove anomalous values in recent performance. Future iterations could adopt more robust statistical or machine-learning–based outlier modelling to distinguish between temporary fluctuations (e.g., fatigue) and genuine performance deviations. Such approaches are consistent with motor-learning research that emphasizes variability analysis and longitudinal tracking of neuromuscular adaptation [8]. Enhancing this module would produce smoother difficulty transitions, more stable measurement of improvement, and higher fidelity in long-term performance profiling while still preserving the responsiveness of the existing implementation.

References

[1] *Effects of Exergames on Motor Skills, Psychological Well-Being, and Cognitive Abilities in Schoolchildren and Adolescents: Scoping Review.* https://pediatrics.jmir.org/2025/1/e71416

[2] *Effects of Virtual Reality Motor Games on Motor Skills in Children with Cerebral Palsy: A Systematic Review and Meta-Analysis.* https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2024.1483370/full

[3] *The Effect of Mixed Reality Technologies for Falls Prevention Among Older Adults: Systematic Review and Meta-analysis.* https://aging.jmir.org/2021/2/e27972

[4] *Berg Balance Testing.* https://www.ncbi.nlm.nih.gov/books/NBK574518/

[5] *From Children to Adults: Motor Performance Across the Life-Span.* https://pubmed.ncbi.nlm.nih.gov/22719958/

[6] *Age Differences in Reaction Time and Attention in a National Telephone Sample of Adults: Education, Sex, and Task Complexity Matter.* https://pmc.ncbi.nlm.nih.gov/articles/PMC2586814/

[7] *The Impact of Perception-Action Training Devices on Quickness and Reaction Time in Female Volleyball Players*. https://www.mdpi.com/2411-5142/9/3/147

[8] *Development and Trainability of Agility in Youth: A Systematic Scoping Review.* https://pmc.ncbi.nlm.nih.gov/articles/PMC9496649/

[9] *Deep Learning for Sports Video Event Detection: Tasks, Datasets, Methods, and Challenges.* https://arxiv.org/html/2505.03991v3

[10] *A Narrative Review of Deep Learning Applications in Sports Performance Analysis: Current Practices, Challenges, and Future Directions.* https://pmc.ncbi.nlm.nih.gov/articles/PMC12382096/