Universidad Politécnica de Madrid

Intelligent Virtual Environments

Assignment 1.2 [Github Link]

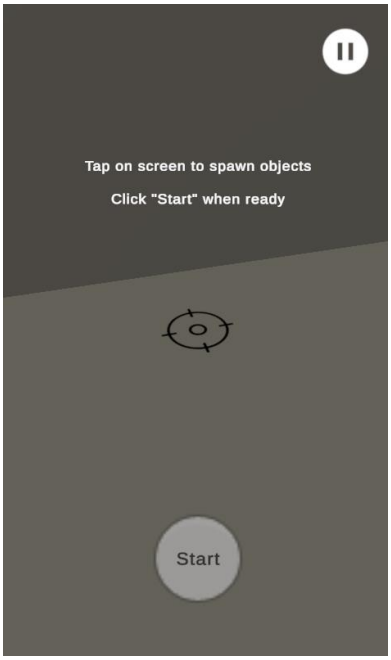
Antonio Franzoso antonio.franzoso@alumnos.upm.es

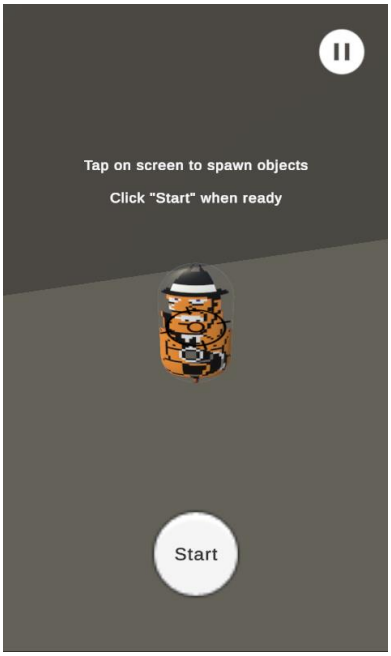
Jose Luis Carrillo joseluis.carrillo @alumnos.upm.es

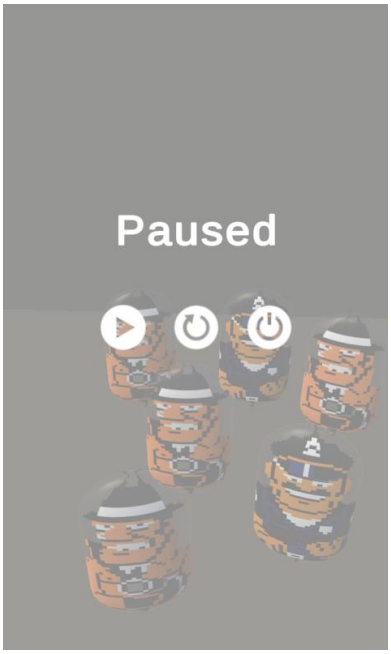
Content

1. Objective

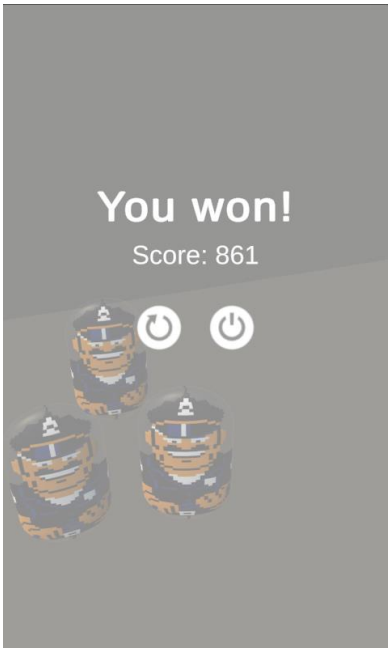
The goal of this project is to adapt the previously developed desktop version of the classic video game Hogan's Alley to a mobile Augmented Reality (AR) platform using Unity. The AR version retains the core mechanics of the desktop game while fully implementing AR-specific features, including plane detection for object placement and interactivity. Players can spawn game objects dynamically on detected planes and engage with the environment in an immersive AR experience.

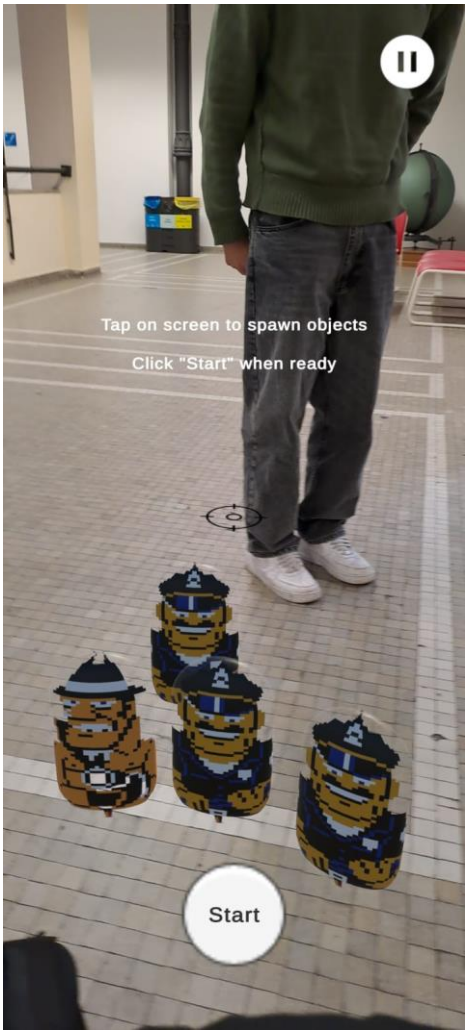







0
4 more
00:14


You won!
Score: 861


Tap on screen to spawn objects
Click "Start" when ready
Start

2. Game Features

2.1 Game Mechanics

- **Dynamic Gameplay Phases**
  - ***Place Phase:*** At the start of the game, the user can spawn as many characters as desired by tapping on detected planes while moving the camera. The "Start" button becomes active once at least one enemy is placed, enabling the transition to the next phase.
  - ***Play Phase:*** Once the game starts, players shoot enemies (while avoiding allies) by tapping the screen. Each tap spawns a bullet from the center of the screen, hitting any objects in its trajectory.
- **Objective:** Players must hit all enemies within a 20-second time limit to win. The game ends in failure if the user either runs out of time or hits even a single ally. A scoring system rewards players based on how quickly they eliminate enemies.

2.2 AR-Specific Features

- **Plane Detection:** The game fully implements plane detection to spawn NPCs dynamically. During the "Place" phase, the user can tap on detected horizontal planes to spawn characters (randomly selected as either enemies or allies) at the location of an indicator. This enables a highly interactive and flexible setup, allowing players to position objects in their environment before starting the game.

2.3 User Interface (UI)

- **Timer:** Displays the remaining time during the "Play" phase.
- **Score:** Updates dynamically based on the player's performance.
- **Start Button:** Allows the user to begin the "Play" phase after placing objects.
- **Pause, Resume, Win, GameOver Menus:** Accessible throughout gameplay for ease of interaction.

2.4 Logging System

- Key game events, including object placement, shots fired, their outcomes, and game-ending scenarios, are logged in a CSV file with timestamps for detailed analysis.

3. How to play

1. **Place Phase**
   - When the application starts, detected planes in your environment will be highlighted with an indicator.

- Tap on the highlighted planes to spawn objects (randomly selected as enemies or allies) at the indicator's location.
- You can move the camera around to detect new planes and place as many objects as you like.
- Once at least one enemy is spawned, the "Start" button (located at the bottom of the screen) becomes active. Press it to begin the game.

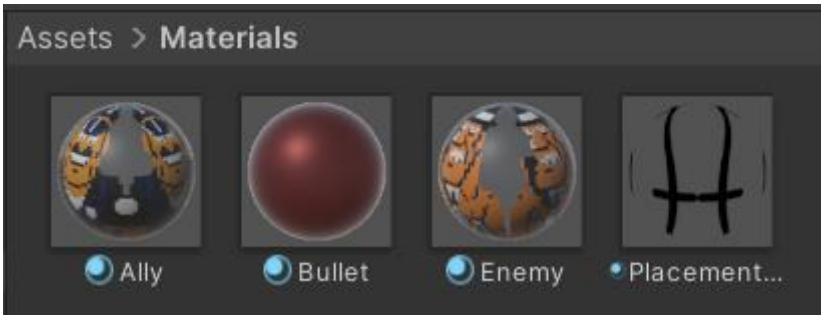
2. **Play Phase**
    - After pressing "Start", the gameplay begins. Your objective is to shoot all the enemies while avoiding allies within a 20-second time limit.
    - To shoot, tap the screen. A bullet will spawn from the center of the screen and hit any objects in its trajectory.
    - Be careful! Hitting an ally results in immediate game over.
3. **Winning and Scoring**
    - You win the game by successfully shooting all enemies before the time runs out.
    - Your score is based on how quickly you eliminate enemies, encouraging accuracy and speed.
4. **Losing Conditions**
    - The game ends in failure if time runs out or you hit even a single ally.
5. **Additional Options**
    - Use the pause button in the UI to manage the game during any phase.

4. Project Structure

4.1 Assets folder

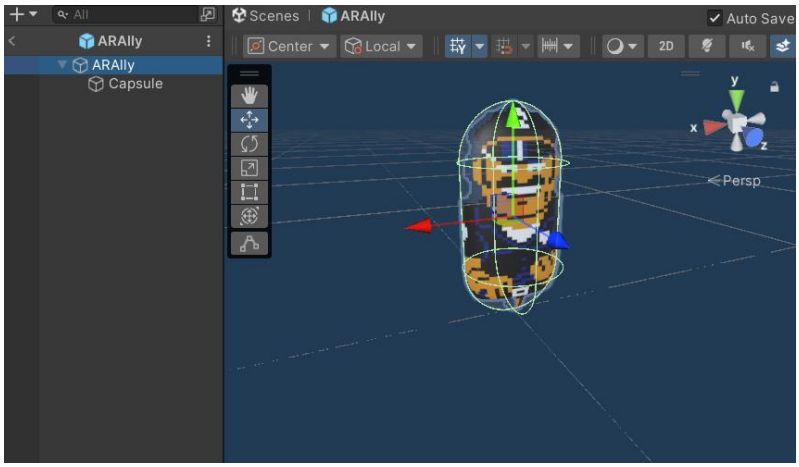
The “Assets” folder is organized as follows:

- **Materials**
    - Used for storing material assets, such as textures and shaders.

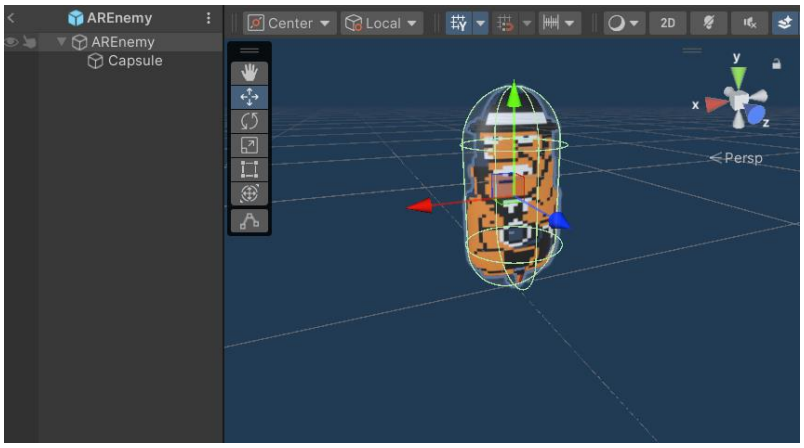


- **Prefabs**
    - Contains reusable GameObjects for spawning and gameplay.

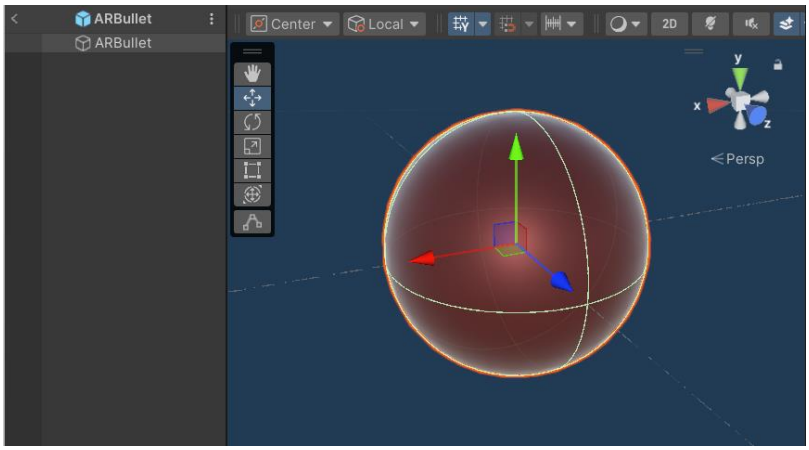
- *ARAlly* – Prefab for ally NPCs



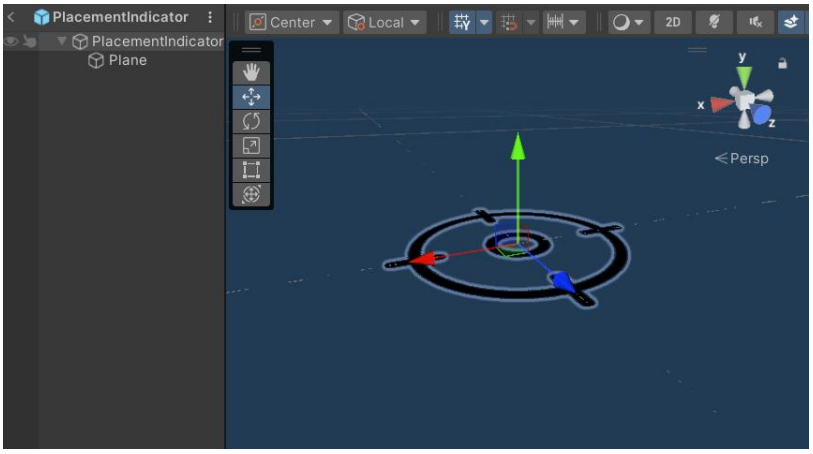
- *AREnemy* – Prefab for enemy NPCs



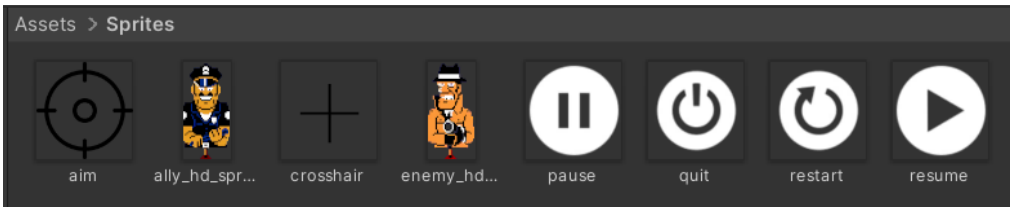
- *ARBullet* – Prefab for bullets fired by the player



- *PlacementIndicator* – Prefab for the indicator used in the Place phase



- **Scenes**
  - *IndicatorScene* – Main scene of the AR game
  - *SampleScene* – Template AR scene
- **Scripts**
  - Includes C# scripts for implementing game logic and functionality.
- **Sprites**
  - Holds 2D assets or images used in the game.



- **TextMeshPro**
  - Automatically generated by importing TextMeshPro.
- **XR**
  - Automatically generated by importing AR Foundations.

4.2 GameObject in the scene

- **Directional Light** - General lighting for the scene.
- **XR Origin** - Represents the player's position and orientation in the AR space, including the camera for capturing the AR environment.
- **AR Session** - Manages the AR functionality.
- **PlacementIndicator** - A visual indicator that shows where objects can be placed in the AR environment.
- **Canvas** - Contains UI elements organized into different states for the game.
- **EventSystem** - Manages input events.
- **UIController** - Handles the logic for displaying and transitioning between different UI modes.
- **GameController** - Overall game logic, such as switching between the "Place" and "Play" phases, and managing win/lose conditions.
- **TimerController** - Manages the countdown timer during gameplay, ensuring time constraints are enforced.

4.3 Interaction, Mechanics and Developed Scripts

- **BulletController** - Manages the behavior of bullets, including their movement and interaction with objects in the game.
- **GameController** - Coordinates the overall game logic, such as transitioning between the "Place" and "Play" phases, win/lose conditions, and game flow.
- **Helper** - Includes utility functions or shared methods to simplify repetitive tasks across different scripts.

- **PlaceOnIndicator** - Controls the spawning of objects (enemies/allies) on the placement indicator on the plane.
- **ShootController** - Implements bullets spawning at user’s touch.
- **TimerController** - Handles the game's timer functionality, ensuring the countdown during gameplay.
- **UIController** - Manages the user interface elements, such as enabling/disabling buttons and displaying menus.