AR Tabletop Defense: An Augmented Reality Game using Unity and AR Foundation

Cornelia Langsenlehner
BSc

ABSTRACT

This project presents the development of a small-scale augmented reality game for iOS using Unity and AR Foundation. The goal was to design an intuitive and stable AR experience with a clear gameplay loop while keeping the overall codebase compact and maintainable. The resulting game demonstrates how simple mechanics, board-constrained enemy movement, and camera-based interaction can be combined to create an engaging tabletop AR experience.

1 INTRODUCTION

The goal of this project was the development of a simple but fully functional augmented reality game for the iPhone. The game was implemented using Unity and AR Foundation and was designed specifically for mobile AR interaction.

During development, particular emphasis was placed on reliable plane detection, intuitive player interaction, and a clear separation between game logic and presentation. In addition, the overall codebase was intentionally kept compact, with approximately 705 lines of code. This constraint encouraged a focused implementation and ensured that all systems remained understandable and maintainable while still covering the core aspects of an augmented reality game.

2 GAME CONCEPT

The developed game is called **AR Tabletop Defense**. It is a simplified tower-defense-style game that is played on a virtual game board placed on a real-world surface such as a table.

After the board has been placed, enemies spawn at the edges of the board and move toward a central base. The player must prevent the enemies from reaching the base by destroying them beforehand. Each enemy that reaches the base reduces the player's remaining lives. Once all lives are lost, the game ends.

Unlike traditional tower defense games, no defensive structures are placed. Instead, the player acts as the sole defensive element through direct camera-based interaction.

3 TECHNICAL IMPLEMENTATION

The project was developed using Unity 2022 LTS (2022.3.62f3) in combination with AR Foundation and ARKit for iOS. Version control was handled using Git and GitHub, allowing for structured and traceable development.

3.1 Technologies Used

- Unity 2022 LTS (2022.3.62f3)
- AR Foundation
- ARKit (iOS)
- C#
- Git and GitHub

3.2 AR Foundation Setup

AR Foundation was used to detect horizontal planes in the user's environment and to place the virtual game board on a suitable surface. The game relies on an AR Session and an XR Origin with an AR Camera. Plane detection is restricted to horizontal surfaces to ensure a stable and consistent gameplay experience. Object placement is handled via raycasting against detected planes.

The spatial properties of the placed board are also used during gameplay. Specifically, the physical size and position of the board are utilized to constrain enemy movement paths. This ensures that gameplay logic remains consistent with the real-world placement of the augmented content.

4 GAMEPLAY LOGIC

The game is structured around three main states: placing, playing, and game over. During the placing state, the player selects a suitable real-world surface and places the game board. Once placement is complete, the game transitions into the playing state, during which enemies spawn and move toward the base. When the player's lives reach zero, the game enters the game-over state.

State management is handled centrally by a dedicated game manager component, which controls transitions and updates the user interface accordingly. Representative screenshots of gameplay and interaction scenarios are provided in the appendix.

4.1 Enemy Behavior

Enemies are spawned at predefined points along the edges of the game board. Instead of moving directly toward the base along a straight line, enemies follow procedurally generated paths that are constrained to the surface of the placed AR board.

For each enemy, a set of random waypoints is generated within the bounds of the board. These bounds are derived from the renderer of the placed board object, ensuring that all movement remains on the tabletop surface regardless of where or how the board is positioned in the real world. The final waypoint of each path is always the central base.

This approach introduces variation in enemy movement while preserving predictable behavior. When an enemy reaches the base, it inflicts damage and is removed from the game. Enemies can also be destroyed by the player before reaching their target.

4.2 Camera-Blocking Enemy Interaction

In addition to standard enemies, the game features a special camera-blocking enemy that introduces a distinct form of augmented reality interaction. This enemy spawns directly in front of the player's camera at irregular intervals and gradually obstructs the field of view.

Unlike regular enemies, this entity does not move across the game board. Instead, it exists primarily in the camera space and visually interferes with the player's perception of the augmented scene. While the camera-blocking enemy is active, all regular enemy movement is temporarily paused, giving the player a short grace period to react.

The only way to remove this enemy is through a physical shaking motion of the mobile device. Device acceleration is monitored to

detect a deliberate shake gesture, upon which the blocking enemy is dismissed and normal gameplay resumes. This mechanic emphasizes physical interaction and reinforces the embodied nature of augmented reality gameplay.

5 Interaction and Controls

5.1 Aiming System

Player interaction is based on a camera-centered aiming system. A fixed crosshair is displayed in the center of the screen, and the player aims by physically moving the device.

A raycast is continuously emitted in the forward direction of the camera. If this ray intersects an enemy, the enemy is destroyed. This interaction method feels natural in an augmented reality context and avoids visual artifacts that can occur with rendered laser beams.

In addition to aiming-based interaction, the game also incorporates gesture-based input. A shaking motion of the device is used to resolve special gameplay situations, such as removing camera-blocking enemies. This interaction leverages the physical affordances of mobile devices and further strengthens the connection between player movement and virtual content.

5.2 User Interface

The user interface provides the player with essential information, including the current score, remaining lives, and status messages indicating the current game state. The interface was implemented using TextMeshPro to ensure clear and readable text on mobile devices.

6 Code Structure

The project is organized into modular scripts, each responsible for a clearly defined task.

- `ARPlaceOnPlane.cs` handles the placement of the game board.
- `GameManager.cs` manages game states, score, and lives.
- `Spawner.cs` controls enemy spawning and difficulty progression.
- `Enemy.cs` manages enemy lifecycle, damage, and scoring.
- `EnemyPathFollower.cs` generates and follows randomized movement paths constrained to the AR board bounds.
- `CameraRayWeapon.cs` implements the camera-based aiming system.
- `CameraBlockerEnemy.cs` implements a camera-blocking enemy that requires a physical shake gesture to be dismissed.
- `HUD.cs` manages the user interface.

7 Conclusion

This project demonstrates the successful development of a functional augmented reality game for iOS using Unity and AR Foundation. By combining camera-based aiming, board-constrained enemy movement, and gesture-driven interactions such as device shaking, a stable and intuitive AR experience was achieved.

The modular structure and clear separation of responsibilities provide a solid foundation for future extensions, such as additional enemy types, wave-based difficulty progression, or further visual and audio polish.

8 Assets and Credits

The development of this project made use of several third-party asset packs. All assets were used in accordance with their respective licenses and are credited below.

8.1 3D Assets

Enemy models and environment elements were sourced from publicly available Unity Asset Store packages:

- "RPG Monster Duo PBR Polyart" by *Dungeon Mason* `https://assetstore.unity.com/packages/3d/characters/creatures/rpg-monster-duo-pbr-polyart-157762`
- "Low-Poly Style Nature" by *Evgenii Nikolskii* `https://assetstore.unity.com/packages/3d/environments/low-poly-style-nature-66322`
- "Low Poly Fruit Pickups" by *Rem Storms* `https://assetstore.unity.com/packages/3d/props/food/low-poly-fruit-pickups-98135`

9 Appendix: Screenshots

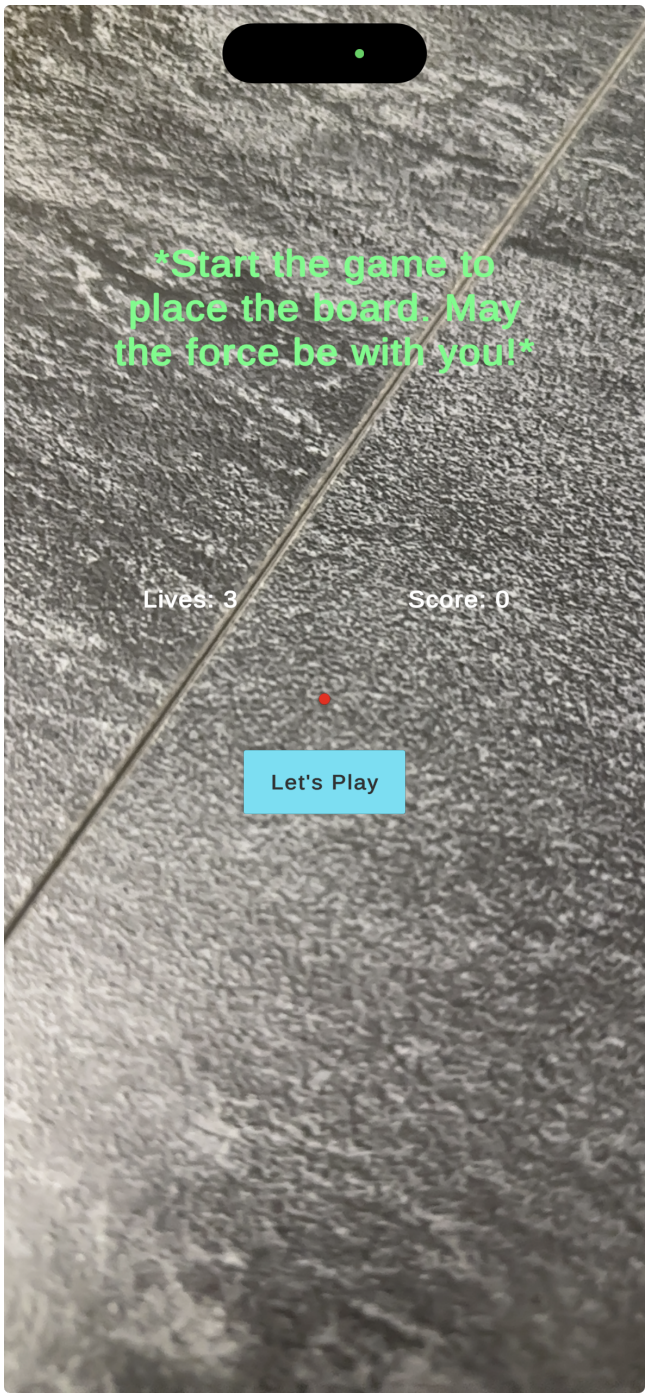

Figure 1: Board placement on a detected horizontal surface.

Figure 2: Gameplay with enemies following randomized paths across the AR board.

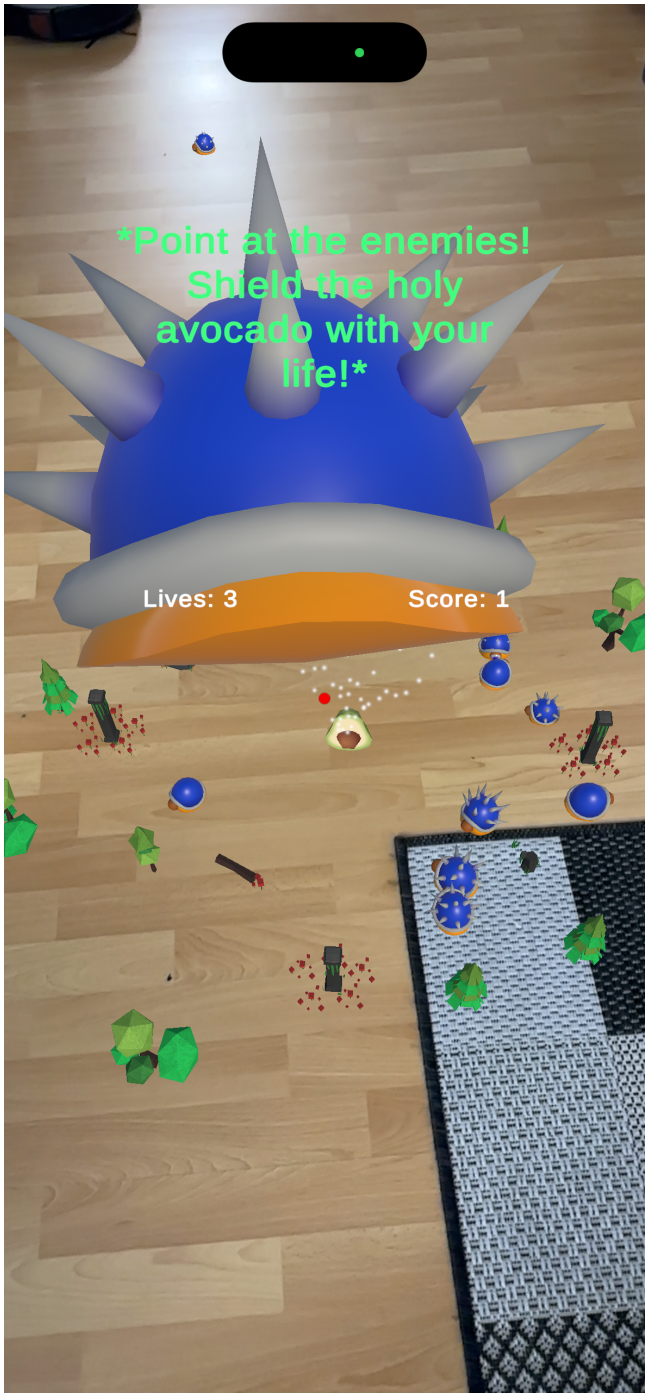

Figure 3: Gameplay temporarily paused while a camera-blocking enemy obstructs the player's view.

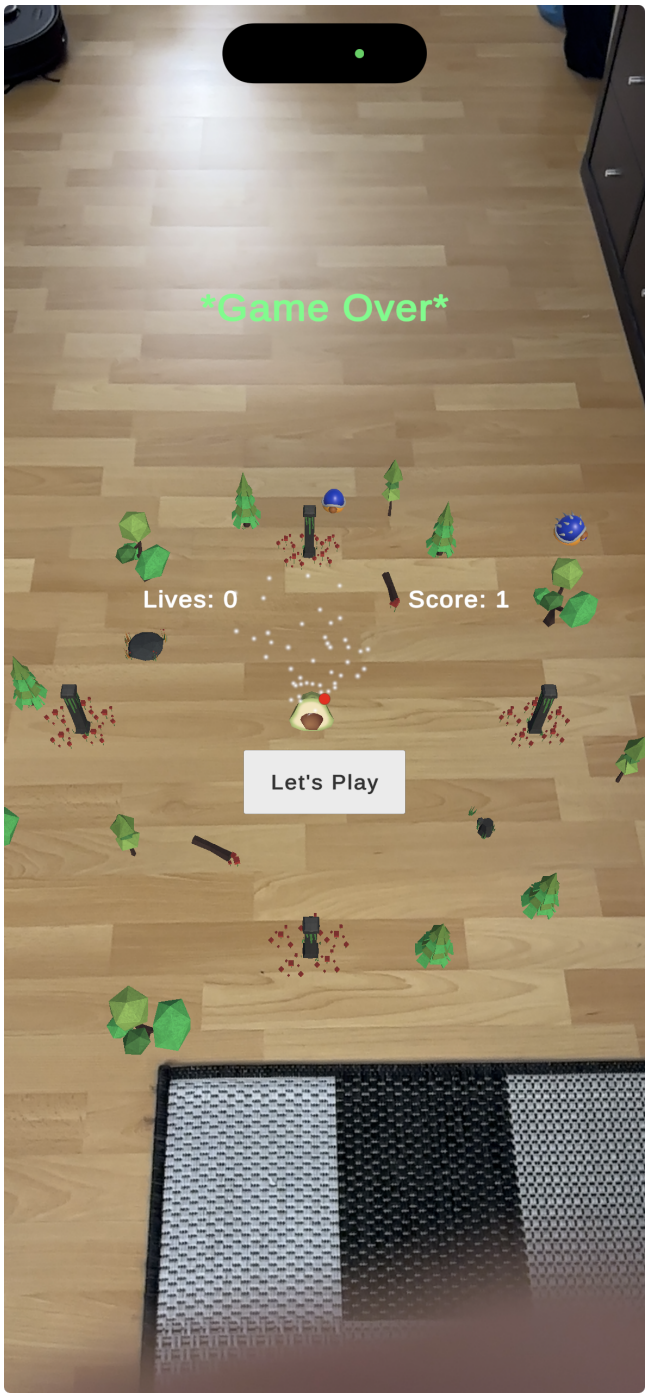

Figure 4: Game-over state after all player lives are lost.