# Assignment 2
## *Hogan's Alley*

Intelligent Virtual Environments:
Technologies, Architectures And Applications

Birk Bregendahl

*30/12 2025*

# 1 Introduction

This assignment focuses on the development of a custom Virtual Reality (VR) game for smartphone, implemented in Unity and deployed as an Android application package (.apk). Unlike the tutorial-based tasks of the earlier assignment, this project involves the design and implementation of an original interactive VR experience, inspired by the classic *Hogan's Alley* arcade shooting game.

The primary goals of the assignment are:

- To design and implement a playable VR game using mobile VR constraints
- To apply Google Cardboard XR Plugin for head-tracked viewing
- To implement gaze-based interaction and shooting mechanics
- To produce a deployable Android APK
- To document the project structure, interaction design, and game mechanics

The resulting application is a stationary VR shooting gallery, where the player aims by moving their head and shoots by tapping the screen. Targets appear on moving platforms, and the player must distinguish between hostile and civilian targets.

# 2. Project Overview

The game recreates a simplified Hogan's Alley–style shooting range adapted for mobile virtual reality. The player is placed at a fixed position within the environment and observes the scene through a stereoscopic VR camera provided by the Google Cardboard XR Plugin. Targets appear on platforms at different heights and positions, creating a dynamic shooting gallery. Each target represents either a hostile character (goon) or a civilian, and the player must quickly identify the correct type before acting.

Interaction is based on head-tracked aiming and tap-based shooting. The player aims by rotating their head, with the device's gyroscope controlling the viewing direction, while shooting is triggered by tapping the screen. When a hostile target is shot, the hostile hit counter is incremented. Civilian targets are designed to disappear automatically after a configurable lifetime if left untouched; however, if a civilian is shot, the civilian hit counter is incremented instead. This mechanic encourages careful target identification rather than indiscriminate shooting.

The game is intentionally designed without artificial locomotion. The player remains stationary at all times, and all interaction is performed through head rotation and gaze-based aiming. This design choice follows best practices for mobile VR by minimizing motion sickness and ensuring a comfortable experience on smartphone-based VR platforms.

# 3. Project Structure

## 3.1 Game Scene

The project consists of a single main scene: *HogansAlley*

This scene serves as the entry point for the Android build and contains the complete game environment, logic, and interaction setup. The scene is configured as the first scene in the build order to ensure it launches immediately when the APK is run.

## 3.2 Assets and GameObjects

**Player and Camera**

- **Player**
    - Contains the Main Camera, which represents the player's viewpoint
    - The camera is driven by the Tracked Pose Driver, receiving head rotation data from the Cardboard XR Plugin
      Includes an Audio Listener for spatial audio

- **CardboardReticlePointer**
    - Attached to the camera
    - Renders a reticle at the center of the view
    - Emits a ray forward to detect interactive objects
    - Uses an Interactive layer mask to filter valid targets

- **Revolver**
    - A low-poly 3D revolver model attached to the camera
    - Serves as a visual representation of the player's weapon
    - Does not affect aiming; aiming is camera-based

**Environment**

- **Floor**
- **Stairs**
- **Plateau Cubes**
- **Moving Platform**
    - Animated platform that carries targets
    - Enhances visual dynamism without requiring player movement

**Targets**

- **Target**
  - Parent GameObject representing a complete target unit
  - Contains:
    - Sprite-based character (bandit or civilian)
    - Collider for hit detection
    - Frame geometry (destroyed together with the target)

**Management Objects**

- **TargetSpawner**
  - Controls when and where targets appear
  - Spawns targets at predefined spawn points

- **SpawnPoints**
  - Collection of transform positions used by the spawner

- **GameManager**
  - Oversees score tracking and general game state

- **Canvas**
  - Displays UI elements such as hit counters

- **EventSystem**
  - Handles UI interaction logic

# 4. Interaction and Game Mechanics

## 4.1 Aiming and Shooting

Aiming is performed using head movement. The player's view direction is determined by the device's gyroscope via the Cardboard XR Plugin. A reticle fixed to the center of the view indicates the aiming direction.

Shooting is triggered by a screen tap on the mobile device. When a tap occurs:

1. A raycast is emitted from the camera forward
2. The ray checks for collisions with objects on the Interactive layer
3. If a target is hit, the target's Hit() method is called

This interaction design is appropriate for Cardboard-based VR, where external controllers are not available.

## 4.2 Target Behavior

Targets in the game are divided into two distinct categories: hostile targets (goons) and civilian targets. While both types share a common structural setup and core behavior, they differ in how they affect gameplay and scoring.

When a target is spawned, it appears with a short scale-based "pop-in" animation, which provides clear visual feedback and draws the player's attention. Each target can be hit only once; after being struck, its collider is disabled to prevent repeated hits. Upon impact, one of several randomized hit sounds is played to add auditory variety, and the target performs a backward falling animation achieved through a rotation around its horizontal axis. After this animation completes, the entire target object—including its frame and collider—is destroyed, ensuring no residual elements remain in the scene.

Civilian targets include additional autonomous behavior to prevent overcrowding of the scene. If a civilian target is not shot within a configurable time window, it automatically despawns. This despawn process is accompanied by a brief shrink animation, providing visual feedback distinct from the hit animation used for shot targets. This mechanic ensures that civilian targets naturally clear from the environment over time while still penalizing the player if they are mistakenly shot.

## 4.3 Feedback Systems

Feedback in the game is provided through visual, audio, and UI elements to clearly communicate player actions and game state. Visual feedback includes the central reticle used for aiming, hit animations where targets fall backward when shot, and short spawn and despawn animations that make target behavior easy to read.

Audio feedback consists of a gunshot sound played on every shot and randomized hit sounds triggered when a target is hit. Background ambient audio is used to reinforce the shooting-gallery atmosphere.

UI feedback is presented through an on-screen Canvas displaying counters for hostile and civilian hits, allowing the player to track performance continuously during gameplay.

# 5. Scripting Overview

## 5.1 Target Script

The Target script encapsulates all target-related behavior:

- Initialization via a spawner
- Hit detection and prevention of multiple hits
- Sound playback using randomized audio clips

- Visual effects (flash, fall animation)
- Civilian auto-despawn logic using coroutines

The script ensures that entire target hierarchies are destroyed together, preventing leftover frames or colliders.

### 5.2 Shooter Script

The Shooter script is attached to the Main Camera and:

- Listens for screen tap input
- Performs raycasting based on camera forward direction
- Plays gunshot audio
- Calls Hit() on valid targets

This script bridges input handling and game interaction logic.

### 5.3 Supporting Scripts

- **TargetSpawner**
    - Randomly selects spawn points
    - Controls spawn timing

- **ScoreManager**
    - Tracks hits on hostile vs civilian targets

- **PointerReceiver**
    - Receives gaze pointer events required by the CardboardReticlePointer
    - Prevents runtime warnings from SendMessage calls

## 6. VR and XR Configuration

The project is configured using Unity's XR Plug-in Management with the Google Cardboard XR Plugin enabled for Android. XR is initialized on startup, and stereoscopic rendering and head-tracked camera rotation are handled automatically by the Cardboard plugin using the device's gyroscope. No artificial locomotion is implemented; the player remains stationary and interacts solely through head rotation and gaze-based aiming. As expected, full head-tracking behavior cannot be observed in the Unity Editor, since desktop systems do not provide mobile sensor input, and is only available when running the application on a physical Android device.

## 7. Build and Deployment

### 7.1 Android Build

The project was successfully built as:

- Android APK (.apk)

Build configuration includes:

- Minimum API level: Android 8.0 (API 26)
- Target API level: Android 15 (API 35)
- Single input system (new Input System)
- Cardboard XR enabled for Android

The APK launches directly into the Hogan's Alley scene and does not include the Cardboard Hello demo scene.

### 7.2 Deliverables

The following files are included in the submission:

- Android APK (playable demo)
- Unity project source files
- Documentation (this report)

Due to file size limitations on GitHub and Moodle, the complete Unity project sources are provided via an external download link, referenced in the repository README.

## 8. Summary of Work Performed

This assignment demonstrates the complete development of a custom mobile VR game using Unity and Google Cardboard. The project integrates XR configuration, gaze-based interaction, animated targets, audio-visual feedback, and Android deployment. Unlike tutorial-based work, the Hogan's Alley project required original design decisions regarding interaction, feedback, and game flow, while respecting the constraints of mobile VR platforms.

The final result is a fully playable VR shooting gallery that adheres to best practices for Cardboard-based VR experiences and fulfills all assignment requirements.