5/5/2022

Lukas Krasauskis

Android Game with Augmented Reality

Dissertation

**CO3201 Computer Science Project**

**Department of Informatics, University of Leicester**

Table of Contents

[Abstract 3](#_Toc102607922)

[Introduction 4](#_Toc102607923)

[Aims 4](#_Toc102607924)

[Objectives 4](#_Toc102607925)

[Features 4](#_Toc102607926)

[Hints 4](#_Toc102607927)

[User Interface 5](#_Toc102607928)

[Game World 5](#_Toc102607929)

[Music 5](#_Toc102607930)

[Visuals 5](#_Toc102607931)

[Monsters 5](#_Toc102607932)

[Statistics 6](#_Toc102607933)

[Monster Spawning 6](#_Toc102607934)

[Distance Based Battles 6](#_Toc102607935)

[Game Mechanics 6](#_Toc102607936)

[Accelerometer 7](#_Toc102607937)

[Progression 7](#_Toc102607938)

[Monster Log 7](#_Toc102607939)

[Alternative Scene for Non-AR Phones 7](#_Toc102607940)

[Saving and Loading 7](#_Toc102607941)

[Description of Gameplay 8](#_Toc102607942)

[Battle 10](#_Toc102607943)

[Outcome of the Battle 10](#_Toc102607944)

[Healing 11](#_Toc102607945)

[Bob Minigame 11](#_Toc102607946)

[Gameplay Loop 11](#_Toc102607947)

[Design Implementation and Testing 12](#_Toc102607948)

[Design 12](#_Toc102607949)

[Implementation 12](#_Toc102607950)

[Overview of the Implementation 12](#_Toc102607951)

[Mapbox Implementation 12](#_Toc102607952)

[World Scene 12](#_Toc102607953)

[Creating The World Scene 13](#_Toc102607954)

[Mapbox Points of Interest 13](#_Toc102607955)

[Code Implementation of Points of Interest 13](#_Toc102607956)

[Monster Spawning 14](#_Toc102607957)

[Distance-Based Battle System 14](#_Toc102607958)

[Augmented Reality 14](#_Toc102607959)

[Accelerometer 15](#_Toc102607960)

[Rock Paper Scissors 16](#_Toc102607961)

[RPG (Roleplaying game) System 16](#_Toc102607962)

[Monster Log 17](#_Toc102607963)

[Alternative Non-AR Scene 17](#_Toc102607964)

[Saving and Loading 17](#_Toc102607965)

[Serialization 18](#_Toc102607966)

[Testing 19](#_Toc102607967)

[Automated Testing 19](#_Toc102607968)

[Human Testing 19](#_Toc102607969)

[Testing Overview 20](#_Toc102607970)

[Critical Appraisal 21](#_Toc102607971)

[Social, Commercial and Economic Impact 23](#_Toc102607972)

[Personal Development 24](#_Toc102607973)

[Conclusion 24](#_Toc102607974)

[Appendix 25](#_Toc102607975)

[Questionnaire Questions 28](#_Toc102607976)

[References 29](#_Toc102607977)

# Abstract

Augmented Reality (AR) games such as Pokémon Go have gathered a player base consisting of hundreds of millions of players all around the world [1]. The game would encourage players to go out and explore to capture Pokémon; this increased people’s daily physical activities and promoted a healthier lifestyle [2]. Therefore, for my project where I have learned to use Unity Engine and have developed a similar game to Pokémon Go, which utilizes a phone’s accelerometer and AR functionalities to make a fun and interactive version of the rock paper scissors game. Where players are encouraged to explore their towns to find and defeat various monsters at rock paper scissors. However, to make the game more interesting RPG (Roleplaying Game) elements have been introduced to make every single encounter slightly different and difficult depending on the player’s level. Moreover, to promote physical activity players are encouraged to go to points of interest such as parks or stores, to heal faster or to play a minigame.

# Introduction

I am developing a game which encourages players to do the physical activity by walking around their towns and fighting monsters using their Phone's Augmented Reality, similarly to Pokémon Go, however instead of capturing Pokémon users must play a fantasy-themed game of rock paper scissors to defeat their opponents. Moreover, the game also implements an RPG (Roleplaying game) system to make fights varied in difficulty to increase replay value and the overall fun of the game.

## Aims

The main aim of the project is to develop an Augmented Reality android game where users can interact with AR objects in reality with the help of the phone.

The secondary objective is to write test cases to test the game and give the game to real-life users to play and give feedback for further improvements.

The project takes inspiration from the Pokémon Go game and gives it a twist by implementing uses of technology such as an accelerometer which allows users to shake their phones to simulate a real-life game of rock paper scissors with augmented reality monsters.

## Objectives

1. Define a game logic for the augmented reality game.
2. Learn Unity and C# for development of the game.
3. Learn how to implement augmented reality in the game using Unity.
4. Implementation of a real-world map with GPS tracking in real time.
5. Implementation of Test Cases
6. Real world testing with users.

# Features

The following section covers all final features included in the final version of the game and is divided into separate groups. However, this section does not cover implementations of the features, and this will be discussed in the later parts of the dissertation.

## Hints

This game heavily focuses on players' self-exploration and discovery; however, to not leave new players clueless, a hint system has been implemented to serve as a friendly reminder.

**Letter from the King**: When a new player starts a game, they are greeted with a letter from King Duffrey which describes in detail what are the goals of the players and how they can be tracked.

**Shake Reminder**: While the player has not defeated any monsters, a game will remind the player every time, they enter a battle to try shaking their phone with an animated moving icon. To help new players find card-rotating features enabled by the accelerometer.

**Victory Condition Reminder**: During the battle scene a victory condition circle can be seen on the top left of the screen which is intended to remind both new and returning players of what beats what in this themed version of Rock Paper Scissors.

## User Interface

The game’s interface has been designed with a minimalistic approach to not overwhelm users. This was achieved by placing only the necessary buttons on the UI and ensuring that important information could be accessed with a minimal number of clicks and that every interaction made by the player would have an instantaneous response. As an example, if a player is too far from the monster, a text appears in the middle of the screen to inform the player about the monster being too far from the player.

## Game World

The game is set in the real world; this is achieved by using Mapbox to get the map data and GPS from the phone to accurately display the player’s position on the map. However, since the game is fantasy-themed, I have created a custom layout which only displays certain roads and makes the buildings look more fantasy-themed to fit with the overall design of the game.

**Pots of Healing:** After battling monsters, the player will lose health and need a way to recover. The game provides small passive healing; however, it is intentionally made to be small to encourage players to go out and find Pots of Healing which increase players healing speed significantly when a player gets in range of the pot. Moreover, these pots are designed to spawn around stores to make them not uncommon with a reduced density, so the players would not have a constant increased healing speed.

**Minigame:** To give players a challenge, a minigame monster was created called Bob the Plague Doctor (Figure 4). Bob only spawns in outdoors and recreation places such as parks where they can challenge him in a rock paper scissors game and try to win as many times as possible in a row. Moreover, the plague doctor spawns in predetermined places such as parks to make it easy for players to find.

## Music

To make the game feel more alive and enjoyable, two royalty-free music tracks have been used that fit with the theme of fantasy. The main world scene uses uplifting and encouraging exploring music, while the battle music is purposefully selected to make the battles feel intense.

## Visuals

Game visuals have been achieved with the help of the Unity asset store and other various other free sources of assets. However, not all of the assets were used without any modifications; for example, the tutorial message and the monster log monster layout were created by me using tools such as Photoshop. Furthermore, all assets that are used within the project can be modified and used without consulting the author. References for assets are listed on a GitLab repository page.

## Monsters

The Player’s goal is to defeat all the monsters in the game and complete their monster log. These monsters do not attack the player by themselves, and the combat only starts after the player presses the desired monster on their phone when they are near the player.

**Varied Monster Difficulties**: All monsters are similar in the way the battle system works except for the Bob minigame. However, to make the game challenging and fun the game has incorporated RPG (Roleplaying game) statistics such as an attack, defence, health and an experience reward.

## Statistics

It is important to note that a player has different statistics compared to a monster since player statistics purely depend on the level. For example, if the player is level 1, all their stats except for health are 1.

**Attack**: this statistic determines how hard the monster hits the player. For example, if the player is level 3 and the monster attack is 5, a player would take 2 points of damage (|3-5|= 2). However, if the attack of a monster is lower than the player's level, a monster deals 1 point of damage in case of a hit.

**Defence:** The defence stat determines if a player can hit the monster efficiently. In cases where the player's level is lower than the monster defence stat, a player would only be able to deal one point of damage to the monster. However, if the player’s level is higher than the monster's damage would be equal to the amount the player's level is.

**Monster Health:**  Monster health is a made-up number which is dependent on how difficult of a challenge a monster is supposed to be to the player.

**Player Health:** The health of the player is calculated by dividing a player level and multiplying it by ten.

**Experience:** Arbitrary number decided based on the challenge level of the monster to reward the player give a sense of progress and powering up trough battles.

**Monster Spawn Rate:**  Arbitrary number that decides how often a specific monster spawns within the players spawn radius.

**Level:** The Level of the player depends on the total accumulated experience from defeated monsters. Furthermore, after every level up an additional 100 experience is required to level up.

## Monster Spawning

Monsters are spawned based on a set of arbitrary relative distances from a player's character. This allows players to find and fight monsters with relative ease since monsters do not rely on regions and area information like Pokémon Go, where people from rural areas have a hard time finding Pokémon to catch. However, this is a persistent problem in location-based games since rural areas are more likely to have less data compared to urban cities.

## Distance Based Battles

To ensure that players are physically and actively participating while playing the game, a distance-based system has been implemented which only allows to challenge monsters that are near the player. Additionally, the system tries to remove the incentive of waiting for monsters to spawn nearby to challenge them by making it more efficient to approach monsters that are further away.

## Game Mechanics

These are the rules every battle scenario fallows.

**Health Bar:** The player and the monsters have health bars that indicate the progress of the battle and the entity who survives is deemed a victor of the battle.

**Win/Lose Scenarios:** If the monster loses all their health the player is deemed a victor and receives experience as a reward and if it was the first time the monster was defeated it is added to the monster log. Otherwise, if the player runs out of health it loses and is brought back to the world scene with zero health and needs to recover, either by waiting or looking for points of interest.

**Escape:** The player is given a choice to escape the battle at any point during the fight if they deem the monster to be too strong to avoid losing their health.

## Accelerometer

The game incorporates phone accelerometer sensors to simulate the rock paper scissors game experience; thus, the players are encouraged to shake their devices as they would be playing a real game of rock paper scissors to trigger their accelerometer sensors and make the cards rotate different choices.

## Progression

To enhance player’s experience and to give a sense of progress various systems and considerations were implemented.

**Experience system:** This allows players to get stronger over time by fighting and winning against monsters. Fights allow players to accumulate experience which increases the overall strength of the player, and monsters that were difficult to defeat become easier and can be challenged appropriately.

**Challenge Level:**  The challenge level of the game is in the hands of the player since the game never stops you from fighting more difficult monsters, therefore, players can challenge the most difficult monsters from the very beginning, requiring players to take high risk for high reward.

## Monster Log

The monster log functions as a progress tracker for the player since it allows players to see how many monsters they have defeated and how many are there left. Furthermore, it also acts as a tracker for the Bob minigame to allow players to see their current high scores. Moreover, the log also acts as a liner progression indicator to the player, since all of the monsters are ordered in order of difficulty to defeat. As it can be seen from this image – Figure 5. The weakest monster is a slime; however, the most challenging one is a shapeshifter in the form of a chicken, except for Bob, which is not meant to be defeated but instead a fun minigame.

## Alternative Scene for Non-AR Phones

The Android game system incorporates an alternative scene for phones that do not support Augmented Reality. This functions exactly like an AR scene, but instead of a real world it uses a manually created scene, thus allowing the game to be played on a larger variety of phones.

## Saving and Loading

It is crucial for all video games to save the user's progress, since losing progress and having to repeat actions that were already accomplished may make the players frustrated to the point where they stop playing the game.

**Saving:** The game ensures the player's progress is saved every time they defeat a monster or improve their bob high score. More importantly, the created save is encoded in order to prevent tampering.

**Loading:** Loads players last saved progress from the stored save file within the device, every time the game is launched.

## Description of Gameplay

It is important to note this game only gives the bare minimum information to the player to encourage player exploration and discovery of how the game works and the challenges that await. Moreover, this description is written from a perspective of a new player.

Upon launching the game players are greeted with the short introductory flavour text which explains to the player the goals of the game and what they must do to achieve those goals (Figure 6). Once, the message is dismissed the player sees the main world screen of the game where the players will spend most of their time.



Figure World Scene

**The World Scene:** The 3D objects that have brown and blue coloured roofs are buildings. Serve as aesthetics and help players orientate regarding where they are. Additionally, this scene utilizes the phone's GPS and Internet to accurately portray the player's location in real-time and is responsible for showing where the monsters are. Furthermore, here players can also find their progression tracking mechanics such as – the monster log on the bottom left screen. Health bar, experience bar and level information on the top left of the screen.

Continuing from the point of player dismissing the flavour text, players should see at least one monster on the screen. However, if the player is too far from the monster, the game will inform the player that it is too far from the monster. Thus, if that were to happen, a player would have to get closer to the monster in order to challenge it. Once the player taps on the monster, the game will transition to a new scene. Here, depending on if the phone supports Augmented Reality, a player will transition to an AR Battle Scene and upon plane detection a selected monster would be placed.

A picture containing text, indoor

Description automatically generated

Figure AR Battle Scene

**AR Battle Scene:** The scene handles player and monster AR battle interactions. Here players challenge the monsters to a game of rock paper scissors. Moreover, if the player is yet to defeat a monster, the game gives a friendly animated reminder to shake their phone to rotate the game choices. Furthermore, looking at the image above, we can see the Augmented Reality scene interface. The bottom of the screen showcases the player choices which rotate upon detection of a shake. Additionally, a player health bar is displayed on top of the screen so players can track their health throughout the battle while the monster's health is displayed above the creature. However, to make the game easier to learn for new players, left corner of the screen contains the reminder chart which displays victory conditions. And more importantly, the players are always given a choice to escape difficult encounters. However, if a player is playing a game in a dark room there might be issues detecting a plane to place a monster upon; thus, players can also utilize the switch in the top right to change between AR and Non-AR Scenes.

A screenshot of a video game

Description automatically generated

Figure non-AR Scene

Alternatively, if the phone did not support Augmented Reality, a player would transition to a Battle scene. Which functions just like the AR Battle Scene, with the exception of not using the camera of a phone to portray the real world.

### Battle

When the battle commences, a player would shake their phone until it lands on the desired card and clicks it to confirm their choice. Once this happens a computer picks one out of three possible outcomes, and an animation plays out to showcase both the player and the monster's choices to announce the winner of the round. If the player wins the round, the monster takes damage, but if the monster wins, the player takes damage. Thus, this battle of attrition continues until either the player or the monster runs out of health. However, if a draw were to happen, no one would take damage.

### Outcome of the Battle

If the player wins it gains the monster's set experience for the battle, and if the monster has not been beaten before, the monster is added to the monster log. However, if the player loses, it gains nothing and cannot challenge others until they recover at least one health point.

### Healing

After battling, players lose health and need to recover. The game provides small passive healing while out of combat for the player. However, since passive healing is designed to be very slow and time-consuming, players will have to look for ways to heal faster; hence they are encouraged to go out and find pots of healing around the map to recover significantly faster.

### Bob Minigame

The Bob minigame also provides players with healing simply by participating in a game of rock paper scissors. Although it is also a monster, it does not act like the others, meaning it cannot be defeated. Therefore, instead, of the usual battle for experience, this minigame focuses on the player's ability to predict the opponent’s choice and to win as many times as possible in a row against Bob while playing rock paper scissors. Moreover, the minigame introduces a counter which tracks their current streak against Bob, and when they lose. Bob sometimes gives a message mocking the player for being too weak.

### Gameplay Loop

The overall gameplay can be summarized into a simple gameplay loop where the player finds and fights monsters to get stronger to be able to fight stronger monsters. Nonetheless, players would have to go out and walk around their towns to find healing pots to make their fighting more efficient. Moreover, this would repeat until the player would complete their monster log.

# Design Implementation and Testing

## Design

The project was created with Object-Oriented and Defensive programming principles in mind, which means both monsters and the player act as objects with properties which define them, and all used objects are always checked for not being null to avoid and catch errors early. All used assets were organized into appropriate categories and placed into folders named: Animations, Audio, Fonts, Mapbox, Materials, Models, Prefabs, Resources, XR (Extended Reality), Scenes, Scripts, Sprites. Most of the folders are self-explanatory for their purposes, however. Scripts folder has additional subfolders**:**

**AR\_SceneScript:** Subfolder contains all the necessary scripts for the AR scene.

**BattleScripts:** This folder contains all scripts related to the rock paper scissors battle.

**WorldScripts:** Subfolder contains scripts used for POI’s, currently only has Pot of Healing script, since Bob minigame is part of the battle scripts.

**AnimationScripts:** Stores scripts related to animation.

**Tests:** Subfolder contains all the automated unit tests for the project.

**Utilities:** Contains elements from both the main scene and the AR scene to control certain aspect of the game such as an accelerometer or singleton.

Nevertheless, this project structure allows to find necessary assets for the project easily and makes overall structure of the project easily manageable and maintainable.

## Implementation

### Overview of the Implementation

The augmented reality game has been created using a Unity Engine [3] 2020.3.26f1, a well-established and known engine for developing all sorts of games. This provides with all the necessary tools to create augmented reality mobile games and run live tests on a computer with an emulated android environment without the need of building and running the application every time to test the implemented features which improves the overall efficiency of the project development. Moreover, due to Unity being an established platform for game development, various third-party APIs (Application Programming Interfaces) have been developed and can be used to achieve difficult tasks such as real-time map data. In my case, Mapbox API [4] was used to create a Pokémon GO-like user interface where the player character is displayed in a world map that corresponds to the real-world map data. Moreover, augmented reality would not have been possible without the help of the ARCore API [5] which enabled the development of augmented reality experiences.

### Mapbox Implementation

Before I go over the implementation of the world and how I created it, I would like to go over the implementation aspects of the Mapbox. Due to me not being experienced in using this powerful tool majority of the implementation has followed their provided tutorials and examples, thus I cannot claim that I have written all of the scripts for tracking and displaying players location accurately since some of the Mapbox functionalities are handled by their own provided example scripts. [6]

### World Scene

This is the main scene where the player’s hero is displayed, and monsters can be found to be challenged (Figure 1). Moreover, this scene was fundamental to achieve likeness akin to the Pokémon Go game, where the player would be able to track themselves on a real-world map using their phone's GPS and mobile data. Furthermore, this was achieved with the help of Mapbox SDK (Software Development Kit) for Unity, which acted as a foundation for the game. Hence, using the Mapbox SDK allowed me to construct a custom design that would fit the game's fantasy theme and display the world map accurately and display the user's position in real-time.

### Creating The World Scene

Reading through documentation of the Mapbox [4] and reading various forums to learn how to design the game world led to utilizing their website's inbuilt tool, which allows developers to choose from their pre-set themes to use as the foundation for the creation of custom maps. As I did in my case, I have selected a basic theme to start building the game world for the players. Choosing a basic theme as a foundation for building a map sped up the cleaning up process, meaning I did not have to manually remove a lot of unnecessary elements such as small streets and railroads. Hence, I could focus on deciding what aspects of the real world I want to display for the players to not oversaturate player device screens with too much information ranging from street names, city names, church names, etc. since they served no purpose for the gameplay. Furthermore, after this clean-up was finished a game world reflecting real-life was left without any unnecessary information and only displayed the roads and buildings. However, the buildings might seem unnecessary, but I have modified theme settings to make every building look fantasy-themed and thus giving the players the immersion of walking around a fantasy world.

### Mapbox Points of Interest

Mapbox is a very powerful mapping tool which allows developers to customize the maps and set up points of interest. Hence, to encourage players to walk around town and do physical activities, Mapbox was used to set up points of interest by selecting specific locations where desired objects would appear on the real-world map. Moreover, further discussed game points of interest were intentionally placed in areas where players would find them easily, such as parks or stores.

**The Bob Minigame**: The minigame point of interest was created to encourage players to go to parks or other outdoor areas by designating Bob to appear in the outdoors and recreation locations such as parks.

**Pot of Healing:** The pots allow players to heal themselves a lot faster compared to the passive regeneration rate. Since after almost every battle player’s will lose their health, and users will have to decide whether to wait with for very slow passive regeneration or go out and search for pots of healing in their local area to heal faster. Moreover, the system was implemented this way to encourage players to go out instead of sitting and waiting for nearby monsters.

Finally, it is important to note that Mapbox does not write any actual code for these objects and only displays them in selected locations. Everything from the effects and how the points of interest behave has been created by me using scripts and Unity features.

### Code Implementation of Points of Interest

**The Bob minigame** was created by adding a special tag to the plague doctor prefab asset named – “Bob“ and modifying the gameplay controller script which is responsible for all battles, to check if a monster with the tag “Bob” has been clicked by the player. And if so, initiate a special version of the rock-paper-scissors game where the player's high score is tracked and updated accordingly in the monster log.

**The Pot of Healing** was created by using an asset from a witch’s cauldron and adding it to my created animation of bubbling to make the pot of healing seem more active and interesting for players to approach. Additionally, the pot only works when a player is within a certain radius of the pot. This was achieved by adding a sphere collider to the pot asset and checking if the player's game object is staying inside the sphere collider's radius using a C# script.

### Monster Spawning

While I was designing the monster spawning script, I decided to avoid Pokémon Go’s approach to spawning Pokémon where they would only spawn them in specific areas or regions. Because from my own experience it causes players from rural areas to not get any Pokémon in their hometowns due to the lack of map data in those areas and therefore makes players quit the game. Hence, my implementation of monster spawning is based on distance from the player. This ensures that monsters spawn no matter where you are in the world. Moreover, the minimum spawn distance for monsters to spawn is around the edge of the clickable range, so players are still required to walk around their hometowns to fight them. Simple diagram illustrating how does the spawning works in principle – Figure 7.

### Distance-Based Battle System

The distance-based system behaves similarly to the Pot of Healing point of interest by incorporating Unity’s engine component– the sphere collider on the player model and designing a script which uses the ends of the collider as triggers for detecting monsters that enter or leave the player radius. Moreover, to make sure that the right objects have entered the player's vicinity, the script checks if an object has a Monster tag, and if it does it sets the Monster class property clickable to true, which allows the monster to be clicked to transition to a battle scene.

To help players know their exact clickable range, an animated pulsating circle has been introduced to showcase the reach of the player. Interestingly, the implementation of this in principle is quite simple, and to achieve this effect I just resize the circle sprite over time up to the outer bounds of the sphere collider and it goes back to being small and the process repeats. Here in Figure 8 can be seen how the pulsating circle looks like in the game.

### Augmented Reality

This section takes a high-level approach to the implementation of augmented reality within the mobile video game project.

Augmented Reality functionality is achieved with the help of Unity’s AR Foundation [7] and ARCore [5] APIs, which allows developers to create augmented reality experiences, track the device's position and orientation in a physical space, and detect the plane where the desired objects can be placed within reality. Although the AR foundation by itself only enables multi-platform development when it comes to AR, the actual augmentation functionalities come from an ARCore plugin which is specifically designed for Android AR development.

To explain how ARCore enables augmented reality experiences, it utilizes a process called simultaneous localization and mapping – SLAM, to understand the phone's relative position and the surrounding area. Furthermore, it uses the phone's camera to detect distinct features called feature points and internal measurements from the device to estimate the position and orientation of the device, known as pose. Afterwards, the device tries to align the pose of the virtual AR camera (which is responsible for the rendering of the 3D content) with the camera of the device to render the virtual 3d models or in my case monsters. [8]

However, this project only utilizes a fraction of the ARCore API’s capabilities, and more details are described and can be found in the official documentation [5]. Although these are fundamental APIs for the project, only a select few features were utilized such as horizontal plane detection, tracking of the monsters, and the device's position in real space.

Before going in-depth about utilizing the AR within the project, it is important to understand how the ARCore finds the horizontal planes. As mentioned before, it utilizes the camera of the phone to find feature points which can be found on horizontal or vertical surfaces. Thus, in our case where monsters can only be rendered on a horizontal plane. AR Core finds a cluster of horizontal feature points and renders the monster there. Additionally, the official documentation also mentions that it struggles to detect feature points on flat surfaces without texture, such as white walls.

To utilize AR functionality fully within developed game, the user needs to have a phone that is at least Android 7.0 version and preferably has a gyroscope and accelerometer sensors within the phone to detect and track the position of the phone in real space more accurately. Additionally, a phone must have a camera or any other optical sensor to create an augmented reality session.

Furthermore, it is important to note that majority of scripts to create an augmented reality environment are used from the ARCore package with some slight modifications, since even the official documentation states that most of the projects should use already written scripts as basis for the AR environment [9] due to it satisfying most of the use cases.

Before we can create a working augmented reality session within the project is important to state that every scene that utilizes AR needs to have an AR session to enable motion tracking, environmental understanding and lighting estimation and AR Session Origin which is responsible for transforming the AR Session coordinates into Unity world coordinates which I can interact with to determine placed object orientation. Additionally, for these objects, the Main camera has to be replaced with an AR Camera which acts as the previously before-mentioned virtual camera to render 3D objects in reality.

Once the AR scene starts a camera tries to detect a horizontal plane and places a monster facing toward the camera (Figure 10). This is achieved by changing the monster's orientation manually during the plane detection phase to always look at the player once it spawns in the real world. Additionally, this enforced spawning condition makes it so that the player does not have to go around the monster to face it and makes the overall experience more enjoyable.

Additionally, once the monster is spawned a health bar is set to be active and is visible above the monster. Since the game does allow the player to fight the monster without looking at it, technically it would invalidate the whole point of the augmented reality system. Nevertheless, this was done intentionally in case there is a problem with a device, or it is taking too long to detect a horizontal plane so the players could still play the game without AR elements. However, to force players to look for the monster and prevent the abuse of the monster not needing to be on the screen, a health bar was specifically placed above the spawned monster, meaning that the player who has not found a placed monster will not know how much health the opponent has and if they are winning.

### Accelerometer

An accelerometer sensor was used to detect the phone shaking since the sensors track the movement of the phone in real space, allowing to detect when the phone is shaking. Nevertheless, the current implementation of shaking detection is contained within Accelerometer.cs file within the utility folder was created by reading the documentation and finding help within Unity’s Forums thread [10] where other developers have claimed to have found the best way to implement the phone shaking detection and thus, I cannot claim that the code is fully mine. Therefore, the current implementation of detecting the shake works by setting a shake detection threshold, meaning that the user must shake their phone up to a certain amount to be detected by the code. The current implementation of the accelerometer may have quite a performance impact on older phones due to it checking every frame for a shake. Nevertheless, this is necessary to ensure that all shakes are detected. Once, I understood the principles behind utilizing accelerometer sensors I have taken the Accelerometer class and used it as a basis to create a separate DetectShake class which is responsible for controlling the animations once a shake is detected. Nevertheless, the current implementation is still not perfect and requires a lot more testing, because sometimes it detects too many shakes and the animation that gets activated whenever a shake happens may overlap with other animations creating animation glitches.

### Rock Paper Scissors

To make the game interactive and interesting to the players, a simple game of rock paper scissors was used as a basis for the combat system within the game. Where players must play a well-known game, however, with a different coat of fantasy-themed paint. In principle, the implementation of a game is simply a player picking one of three options and a random number generator-based code decides an opponent’s pick. However, to make the game more active and to reduce the number of rounds that turn out to be a draw and artificial stopper has been implemented which prevents fifty per cent of draw rounds from happening. To put it in simple terms, every time a draw would have happened, a computer performs a coin toss check to see if it should pick a different card. This makes the game feel less sluggish, and the rock paper scissors game more enjoyable.

Additionally, to simulate a real-world game of rock paper scissors, an accelerometer sensor was used to detect the shakes of the phone, which meant players could either pick the three cards normally or shake their phone to rotate their choices. Likewise, this was introduced hoping to improve the game's fun factor and to introduce a unique interaction method between the player and the game.

### RPG (Roleplaying game) System

However, since a normal game of rock paper scissors without any changes would get boring fast, the game would lose the interest of players quite quickly. To mitigate this, an RPG-type system was implemented to both introduce a challenge to the game and make the experience of the player overall more enjoyable. This RPG system incorporated statistics both for the players and the monsters as described in the previous sections making all the combat encounters varied in challenge levels and increasing the replayability value for the player. Furthermore, the introduction of an RPG system required an overhaul of the original rock paper scissors implementation and made it severely more complex since now combat encounters would last more than one turn and both players and monsters would take differing amounts of damage depending on the players’ level.

Additionally, with the introduction of the RPG system, a health system was introduced to make encounters feel dangerous and to teach the player how strong each of the monsters is. Since if the player's health dropped below one point, they could not participate in combat encounters and would require players either to go out and find a healing pots or wait while a passive regeneration would heal them. Moreover, if the game was meant to be monetized, this element of waiting could be used to encourage players to spend real-world money on healing items.

Furthermore, to not punish players when they start a fight against a monster that is too strong players always have an option to escape the battle and keep their current health. However, players are not warned in advance if they challenge a monster that is too strong, and this is something a player would discover during their playtime.

### Monster Log

While creating the monster log (Figure 5) I have decided to make it easily modifiable and scalable to make it easier to add more monsters if needed in the future. Hence, the monster log utilizes Unity’s grid element system, which makes sure every monster Icon is actually an element which fits in a designated grid and can be automatically resized depending on a phone screen size without the need for manual creation of the monster logs for differing phone sizes and resolutions. Moreover, since every element in the monster log is within a grid, it makes it easy for the developer to update the monster log every time a player beats a monster. In this case, the shadow monster element is just switched out for a coloured monster element to showcase that the monster has been defeated.

### Alternative Non-AR Scene

This feature was not originally within the scope of the project. However, I felt like I should not isolate the player base and design and implement a scene where users who use older phones that do not support augmented reality could still enjoy the game. Hence, I had to write a script to detect whether the phone supports an AR Session, and thankfully the official documentation from the AR Foundation provides examples of how such a method would be implemented [7]. And once I detected whether the phone used AR or not, I could just redirect them to different scenes utilizing the above-mentioned script. Thus, the Battle scene was created utilizing Mapboxe’s world map as a background and 3D buildings for the visuals. I am quite happy with how it turned out since visually it looks like a huge monster is attacking a city and our hero must defeat it. Moreover, functionally the alternative scene utilizes the same scripts and battle mechanics as the AR scene, with the obvious exclusion of scripts utilizing Augmented Reality to place the objects in reality. Nevertheless, I have designed and implemented this scene with the intent to make it easily modifiable and updatable with the rest of the project. Thus, the scene repurposes rock-paper-scissors gameplay controlling scripts and game objects to create almost an identical copy of the AR scene, but without the AR elements. Additionally, since the non-AR scene repurposes the same scripts as the AR scene, I can develop and update both scenes' code simultaneously.

### Saving and Loading

The game serializes the current player and monsters captured monsters’ data and saves the player's progress in a DAT format file. Originally the save data was supposed to be stored in a JSON file format; however, I have changed it to DAT since it encodes the player information and reduces the chances of players cheating within the game. Unlike how JSON would just display the data in plain text and would allow players to easily modify their information.

**Saving:** Saving happens when the player gains experience, adds a monster to the log, or increases their high score in a Bob minigame. Once this occurs, the game serializes all of the information about the player and captured monsters, then it stores it within an encoded DAT file which is located at “/storage/emulated/0/Android/data/DefaultCompany/files/playerSave.dat”. However, according to Unity documentation location might differ on older phones [11].

**Loading:** When the game launches it checks if a save file exists within the device and if it does exist it deserializes the data stored within a file and loads the data to the game.

### Serialization

Since the game utilizes this technique to save and load, I decided to elaborate on how the process works in principle. “Object serialization is the process of writing the state of an object to a stream. Deserialization is the process of rebuilding the stream back into an object” [12]. Therefore, for the data to be serialized and stored I have created two new classes PlayerData.cs and MonsterData.cs which store both the player and monster statistics in a serialized form. However, before saving the player data, every defeated monster has to be serialized separately, due to being a separate game object. The design choice to save monsters as objects was made so it would be easier to get data from defeated monsters. Currently, I only utilize their names to enable the monster log. Nevertheless, if the development of the game continued and the monster log was to be expanded, data from the captured monsters could be easily extracted and displayed for the player if needed. Diagram showcasing how does the saving and loading process works – Figure 9.

## Testing

Unity Engine comes with an inbuilt Unity Test Runner, which can perform two types of tests PlayMode and EditMode. The Playmode tests are run as a coroutine meaning the actual game session starts and then it tries to perform tests at the same time. Unlike how EditMode tests can be run without an actual instance of the game running since they are used to test functionalities that don’t need all game elements. However, as powerful as the Unity testing suite is, Unity itself does not have any inbuilt tools to provide Augmented Reality testing without human interaction. Hence, all the testing related to AR needed a human tester to interact with the game and make sure everything behaves as expected.

In order to understand further limitations of Unity’s testing suite I have looked into research paper which went over the process of testing a mobile puzzle game. [13]

### Automated Testing

All of the automated testing was carried out using the EditMode functionality since most of the feature code could be isolated for testing and to utilize PlayMode tests I would have to reconstruct entire scenes in code, hence when it came to visual testing I would do it manually. However, once I started implementing automated testing scripts, I soon realized that I had to rework some of the code functions. For example, the rock paper scissors class needed to be reworked to include a Boolean value – testing to prevent running coroutines that would cause tests to break. Moreover, the DetermineWinner function initially was void and did not return anything. Hence, I reworked it to return integer values ranging from -1 – loss, 0 – draw and 1 – victory to track if the game was returning a correct response. The tests were used to test all of the player statistics, including monster addition, levelling and more importantly the rock paper scissors minigame and both saving and loading functionalities. Additionally, automated testing allowed me to detect various bugs and fix them. A worthy mention would be that during battle encounters players were still regenerating health, meaning the player could just wait when they were low and fight the monster again. This was caused due to an oversight of my placing auto-regeneration on the game manager and not checking if the battle was happening. Thus, the player would keep regenerating health in all instances and had the potential to break the game's difficulty.

### Human Testing

Throughout, the year of developing the game I have tested augmented reality numerous times to understand how it works and to achieve the expected outcome. However, to gather less bias and accurate testing results, I have asked my acquaintances to test the game from the perspective of an end-user to find all the shortcomings and possible future improvements for the project. The testers have been given a stable version of the game to try out for a day and afterwards to fill out a questionnaire. The questionnaire focused on two elements: the fun factor of the game and how the augmented reality functions in the hands of the players.

Going into the test I was worried that there might be some resolution issues with various phones since all the testing was performed on my own device and I have applied techniques that are supposed to mitigate this issue. But it was never tested during the development process. Thankfully, this issue did not seem to have happened during the human testing and user interface elements were scaled appropriately on various phone resolutions.

Once the game has been installed on the player's phones, I have not been given any guidance on how to play the game access or whether the game itself was doing a good job at explaining the gameplay elements to the players. After the players tested the game for a day and filled out their questionnaires, I had discussions regarding what could be improved with the game and if there were any issues. The list of the questions asked can be found in the appendix – Questionnaire Questions section. Thus, I will only discuss the information I found interesting and should be mentioned. Firstly, I have asked players to describe their gameplay experiences and whether it was easy to follow. The majority of players have confirmed that the gameplay elements were intuitive and that the initial flavour text has explained sufficiently what they are supposed to do in the game world. However, there were a few players who accidentally closed their initial flavour text and had trouble understanding what they were supposed to do. Nevertheless, this issue was addressed in the newer version of the game with the implementation of an X button instead of tapping anywhere on the screen to dismiss the message, thus reducing the chance of accidentally closing the introduction text.

Furthermore, I have found that players who were using older phones sometimes had trouble finding monsters and apparently, switching between AR and Non-AR scenes would act as a refresh and monsters would appear on an AR scene normally. Due to the lack of time restraints, I was not able to find an appropriate solution for this problem other than refreshing the scene or just playing without AR elements. Moreover, I was glad to find out that the majority of the players preferred to play the game in AR mode since it was the main feature of the game and players found it fun to fight monsters with the camera of a phone.

However, when it came to questions regarding utilizing the accelerometer sensor to switch between choices, half of the participants have said that they enjoyed the feature but thought of it more as a gimmick, therefore, after a couple of battles, stopped utilizing it for battles. While the other half appreciated this new interaction but pointed out that sometimes the card switching animation would glitch out, which was a known bug.

Additionally, since the participants were both from rural and urban areas, the world map scene could be tested appropriately. From this, I have gathered that people who live in urban areas had no issues when it came to finding Mapbox's points of interest (POI). However, rural area players would find it more difficult to find POI due to the lack of data from their region within the Mapbox. Nevertheless, this was expected behaviour and after some discussions with players from rural areas, a suggestion has been made. For a feature which would allow players to request additional POIs in their area if the game continued its development.

Finally, during the discussions, some of the participants raised concerns about the longevity of the game and possible balance issues. Claiming that the experience for some monsters was too generous concerning their difficulty, thus some changes have been made for the current monsters with the player's feedback in mind to reduce the experience gained from weaker monsters. However, when it came to games longevity one of the players pointed out that the levelling system was too simplistic and did not give enough freedom to the player and it seemed like most players agreed with that statement. Therefore, if development were to continue, it seems like a rework of the current levelling system would be needed.

### Testing Overview

Thanks to both automated and human tests, I have identified the shortcomings of the game and was able to address some of the issues within the game. Additionally, the tests made the game turn out to be almost bug-free except for an animation glitch caused by the accelerometer. Perhaps, more importantly, I have found out that the game was fun to play for the players and encouraged them to go out and explore while improving their wellbeing. Out of the two tests, I prefer the human method since it gives me a direct insight into a player's mind and helps me to understand as a developer what I should change or improve in future updates.

# Critical Appraisal

In order to evaluate my game, I will first investigate what I originally set out to accomplish and what I have made. “The primary aim of the project is to develop a mobile game utilizing AR technology that would allow users to interact with AR (alternative reality) objects with the help of mobile phones.” – an extract from the interim report. Interpreting this statement at face value, it can be said that I have accomplished what I have set out to achieve. However, to do my due diligence I will go in-depth on how the development went, what could have been done better, and what still could have been improved upon.

To provide a brief description of the final developed product, I have developed an augmented reality android game which takes inspiration from roleplaying games such as dungeon and dragons for its fantasy themes and encompasses similar gameplay elements to Pokémon Go to encourage players to explore their hometowns while doing physical activities and completing their monster logs. Moreover, the game makes a twist on the original Pokémon Go’s catching monsters mechanic and makes the player fight monsters instead by playing a well-known game of rock paper scissors. Additionally, the game utilizes devices with accelerometer sensors to improve the immersion of the players and make them feel like they are actually fighting the monsters in a game of rock paper scissors.

Only by comparing the goals, I have set out originally to what I have created, I can accurately analyse the overall success of the project. The initial objectives of the project were not directly related to the game itself, but more of the necessary preparation for the project. During my first months, I studied the C# language, Unity Engine, and Mapbox to learn everything that was needed for developing an augmented reality android game which used real-world map data as a foundation for the game world. Due to my being unfamiliar with any of these technologies beforehand, I have made some critical errors which required reworking the project from the ground up. Thankfully, I did not need to scrap everything I had created initially and was able to move crucial functions to a new version with slight modifications. Hence, the final project is version 2. Throughout this period, I have learned a lot about developing games in general and how Augmented Reality functions. Moreover, this made developing version 2 of the game go much smoother and with fewer mistakes. Thus, I can say that I have achieved the first objective of the project since now I am confident around Unity Engine, C# and Mapbox.

Furthermore, one of the objectives was to write custom scripts to allow players to interact with the monsters. Hence, the gameplay element was introduced within the project and players could not only observe the placed monsters in reality through their cameras but also play a game of rock paper scissors against them. The initial code for the rock-paper-scissors battle system was just a simple implementation where both a random number generator and a player would pick a card and one of them would win. However, I deemed the concept to be too simplistic and therefore everything was overhauled. New classes were created – Player and Monster which would implement RPG statistics both to the player and the monster assets and thus make the game more complex. Moreover, I think this was a good design choice since this object-orientated implementation made it easy to modify monsters’ statistics and made it simple to create battles for various challenge levels. Also, the classes were specifically designed to allow me as a developer to change values during testing, meaning that monster or player statistics could be changed mid-battle and thus make the balancing of the game an easier process. More importantly, the new system made it so new monsters could be added to the roaster with relative ease simply by attaching a script to an asset and changing the default values to reflect the monster’s strength. Nevertheless, I am very happy with the current implementation of the system since it can be easily modified and expanded.

To add to the battle system, an accelerometer was introduced to the battles to make them more interesting. But due to it being my first time using such a sensor, I underestimated the number of responses the sensor provides, and this caused glitches to happen sometimes when a phone is shaken due to an overwhelming number of inputs going into the function handling the card rotation animation. To mitigate this bug from happening I have increased the necessary threshold for activating the animation which seemed to reduce the glitches; however, they were still appearing in rare instances. Nevertheless, I am satisfied with the current implementation of the battle system since it introduced new technology and gave a twist on battling monsters with the introduction of an accelerometer which still could be improved upon.

Continuing to look at further objectives, one of them was to implement a real-world map with the help of Mapbox SDK. I believe this objective was accomplished as intended since the game's main scene utilizes a Mapbox custom map to display users’ location and surroundings using real-world map data as intended. Nonetheless, Mapbox was one of the reasons why I had to reconstruct my initial project from the ground up since it caused a lot of compilation errors when initially important and required a lot of trial and error to make it work. However, once the SDK was implemented within Unity Engine, it acted as a fundamental tool when it came to world-building. Moreover, with the implementation of the Mapbox, I was able to utilize its vast world map data to generate Points of interest (Healing Pot, Bob minigame) at designated locations. Furthermore, if development continued for the project due to Mapbox's versatility, the world map could be expanded with relative ease by adding new objects or creating custom regions for specific monsters. Overall, I am quite happy with the design choice to implement Mapbox since once it was fully set up it allowed me to modify the world map to the tiniest detail with minimal effort.

One of the objectives was to design and implement a progress tracking system known as a monster log. I am quite happy with how it turned out in the end since I have designed it by taking inspiration from how developers create game inventories by using Unity’s grid-based layout which automatically sorts out and makes sure all grid elements fit in the designated frame. This means if there was a need, more monsters could be added to the monster log with minimal effort due to the current implementation only requiring the creation of two pictures: a shadow and a coloured version of the monster to be added. However, I do find some flaws in the current monster log system, since it is based on the monster’s name and not the object itself. This makes addition of monsters a bit more tedious than it should since I have to manually get the newest added monster and check its name to enable it. Preferably in future projects, if I were to develop such a progress tracker, I would implement checking based on the object itself since it would make the code more efficient.

Nevertheless, there were some changes to the original objectives when it came to developing a game. For example, I initially thought to implementation of a database was crucial to storing monster and player data. However, when I started implementing the classes for the monsters, I soon realised that the creation of a database would be unnecessary, due to the overall scale of the project and the simplicity of monster and player data modification from the developer’s site. But, in hindsight, an implementation of a database would only be needed if a game-used online functionality required users to log in to store data across multiple devices. And, since the system does not implement any online functionality, it was deemed to be unnecessary, because the game stores save files locally and players can transfer them between devices manually.

Continuing on changes to the initial project, originally, I have planned to create a dungeon system where players would have to battle out a series of monsters for experience, however due to the gameplay being dependent heavily on luck and the possibility of it breaking the progression of the game. I have designed a minigame called Bob which challenges the player's luck in a minigame where a player must win as many times in a row as possible to increase their high score. From a critical point of view, a dungeon system might have encouraged players to sit in one spot and level up by participating in dungeons instead of doing physical activity by fighting monsters. Additionally, the Bob minigame does not break any fundamental game systems and is just there to help players recover health and test their luck.

Additionally, I have implemented features that were outside the scope of the initial project with the suggestion of a marker to include a scene which would allow players who do not have an AR-supported phone to play the game. I am quite happy with this design choice due to not only allows more players to enjoy the game but also acts as a safeguard for the players who play with augmented reality, since they can switch to a non-AR mode scene freely if there are problems with plane detection and monsters are not being placed.

To analyse the product, I will look at the overall design of the software. In hindsight, I am very pleased with the final version of the game and the majority of my design choices when it came to the creation of object classes since they were designed to be easily extendable and modifiable. Making it possible to add new monsters, and points of interest to the world map with minimal effort and tweaking. Another good design choice is an implementation of a loader game object which was specifically designed as a singleton pattern [14] (a design pattern which restricts classes to one single instance) to hold Game manager and scene transition scripts allowing to always track the player class and transition between scenes with minimal effort due to being able to access both the player values and the Scene Manager class from any other class in the project. However, looking at the negatives of the design I can say that analysing the final product I find the RPG system a bit lacking on the player side, due to all statistics being based on the level of the player and thus if I had more time I would rework the whole levelling up system with more freedom allowing the player to select what statistics they would like to improve and thus allowing the game to have more variety in player strategy. For example, allowing players to create heroes who are very strong on offence but very weak when it comes to defence.

In concerns to human testing, I believe it was carried out very well since it gave insight into a new player's perspective. However, if I was to criticize the testing it would have to be the small testing period and the low number of participants, since it is real-world game development, there would be a lot more testers and the testing would take longer. Hence, I believe if the test would have taken longer, the player's interest in the game could have been measured more accurately, and with a bigger sample size of participants, changes to the game could have been justified.

This led to my overall evaluation of the project and what I would do differently. Concerning the use of Unity Engine for my future projects, I would say that the engine itself did not hold back my development; in fact, it allowed me to achieve everything I wanted with one exception. I was not able to test AR functionality within the engine itself without a paid plugin, which meant every time I needed to run a test related to AR, I needed to wait 3-7 minutes of build time to load it up on my phone and test it manually. Hence, if I was to develop another augmented reality, have, I would heavily consider looking for a different tool suite or just buy a plugin which allows in engine AR testing.

# Social, Commercial and Economic Impact

Due to the project being an android mobile game the future success of this project heavily depends on whether it can be commercialised similarly to Pokémon Go by implementing monetization systems such as allowing players to buy healing potions for real money instead of going around to healing points. Moreover, since the game works on phones which run on Android 7.0 Nougat at least and according to the statistic from 2020 [15] showing that more than 50% of android users used version 7.0 or newer it can be estimated that the game will not suffer from the lack of supported devices. Socially the game in its current condition only promotes people’s health wellbeing by encouraging players to go out, explore their hometowns and do physical activities. Additionally, I believe an introduction of an online system which would allow players to interact with each other not only would help out the game's commercial success but also may create a snowball effect where people would invite more people to play the game together similarly to Pokémon Go.

# Personal Development

After creating this project, I feel like I have gained an understanding of new tools and learned a new programming language – C# which will be useful in my future career endeavours. Additionally, I have gained insight into how the process of game development happens and how the augmented reality experiences are achieved on a technical level and can be utilized within a concept of a game. Although the most useful skill I have improved while developing this project would have to be the ability to research and understand various programming problems better than before.

# Conclusion

For this project, I have designed and developed an augmented reality-based mobile video game which incorporates mechanics from well-known games such as Pokémon Go and Dungeons and Dragons to create an enjoyable gameplay experience. The game features support for both AR and non-AR phones by implementing two different scenes, one with AR functionality and another without. Additionally, these scenes were created to be easily modifiable and updatable simultaneously from the coding perspective due to utilizing mostly the same scripts allowing developers to make changes on both scenes simultaneously. Moreover, the system utilizes device accelerometer sensors to simulate a real-life rock-paper-scissors game by rotating the user's game choices. Although there seem to be some minor visual bugs caused by the current implementation of the rotation animation, however, steps were taken to mitigate the bugs. But they still appear from time to time, depending on how heavily the phone is shaken. Furthermore, the future of the work should aim to completely fix the visual bug caused by the accelerometer and implement monetization systems to create a commercially viable game.

Due to game systems being designed to be modifiable and upgradable with minimal effort. New content such as monsters or points of interest can be added rapidly to increase the longevity of the game. Additionally, the augmented reality systems could be improved to implement vertical plane monsters to make some monsters appear on the walls instead of the floor or tables to make the battle system more intriguing and find monsters more fun. Overall, I believe I have achieved my original aims and objectives with slight modifications and developed a game I have set out to create.

# Appendix

Diagram

Description automatically generated with medium confidence

Figure Bob The Plague Doctor

Graphical user interface

Description automatically generated

Figure Monster Log

Text, letter

Description automatically generated

Figure Introductory Flavour Text

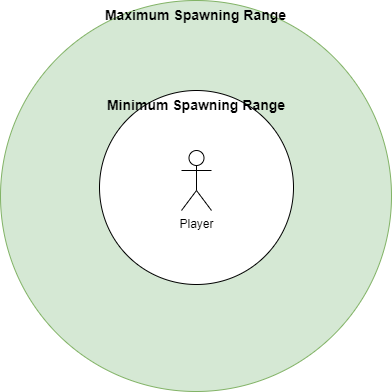


Figure Diagram Showcasing how does the spawning work, the green area is where the monster spawn.

A picture containing indoor

Description automatically generated

Figure Pulsating Aura, showing the players clickable range.

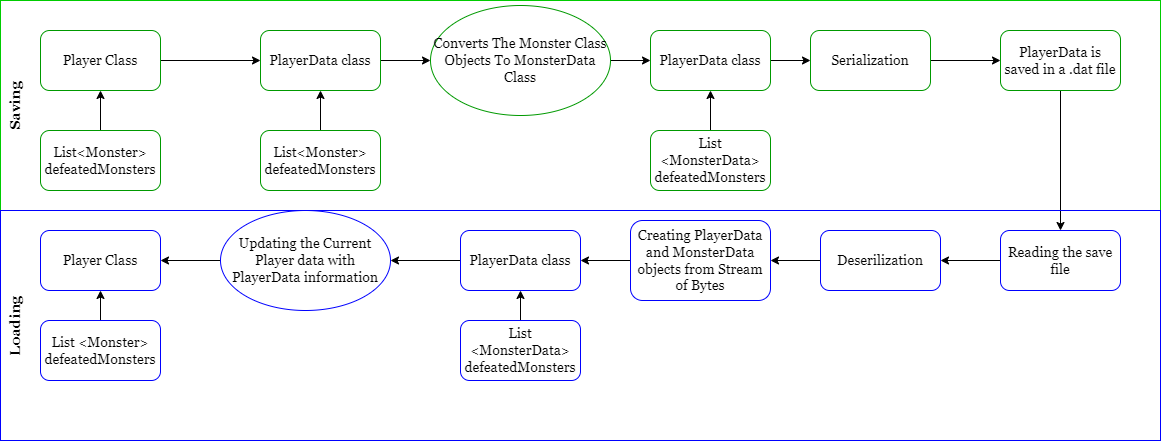


Figure Saving and Loading Diagram



Figure AR scene with plane detection visualised

## Questionnaire Questions

The questionaries’ goal was to access the implemented systems within the game and to know how easy it is to pick up for new players. The test itself was done when I was back at home in Lithuania therefore all the answers, I was provided to these questions are not in English. However, if proof is needed, I can provide the participant answers.

* What is your phone’s Android version?
* Was the game providing with enough information to understand the goal of the game?
* Were you able to navigate the game?
* Was it Difficult to find monsters in AR Scene?
* Were there any issues utilizing Augmented Reality?
* Did You spend more time fighting monsters in AR or Non-AR Scene?
* Were you able to rotate between your card choices by shaking your phone?
* Did you prefer using the shaking motion to pick the card or did you just tap on the screen to select your cards without shaking?
* Please elaborate the answer to why was that preferred choice.
* Were there any issues regarding GPS Tracking?
* Was it easy to find Monsters to battle?
* Did you go out to find Points of Interest (Plague Doctor or Pot of Healing) to recover faster?
* Did you feel the game was unbalanced in terms on difficulty?
* Please write down any additional comments you have.

# References

|  |  |
| --- | --- |
| [1] | M. Iqbal, “Pokemon Go Revenue and Usage Statistics (2022),” BusinessOfApps, 11 01 2022. [Online]. Available: https://www.businessofapps.com/data/pokemon-go-statistics/. [Accessed 10 04 2022]. |
| [2] | T. Althoff, R. W. White and E. Horvitz, “Influence of Pokémon Go on Physical Activity: Study and Implications,” *J Med Internet Res,* vol. 18, p. 315, 2016. |
| [3] | “Unity Documentation,” Unity, [Online]. Available: https://docs.unity.com/. [Accessed 04 05 2022]. |
| [4] | “Mapbox Documentation,” Mapbox, [Online]. Available: https://docs.mapbox.com/. [Accessed 04 05 2022]. |
| [5] | “API Reference - ARCore Extensions for AR Foundation,” Google, [Online]. Available: https://developers.google.com/ar/reference/unity-arf. [Accessed 04 05 2022]. |
| [6] | “mapbox-unity-sdk,” Mapbox, [Online]. Available: https://github.com/mapbox/mapbox-unity-sdk. [Accessed 04 05 2022]. |
| [7] | “About AR Foundation,” Unity, [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@4.2/manual/index.html. [Accessed 04 05 2022]. |
| [8] | “Fundamental concepts,” Google, [Online]. Available: https://developers.google.com/ar/develop/fundamentals. [Accessed 04 05 2022]. |
| [9] | “Using ARCore XR Plug-in,” [Online]. Available: https://docs.unity3d.com/Packages/com.unity.xr.arcore@4.2/manual/index.html. [Accessed 04 05 2022]. |
| [10] | “Unity Forums: Iphone Shaking Thread,” 17 November 2008. [Online]. Available: https://forum.unity.com/threads/iphone-shaking.15029/. [Accessed 2022 05 04]. |
| [11] | “Application.persistentDataPath,” Unity, [Online]. Available: https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html. [Accessed 04 05 2022]. |
| [12] | M. Hericko, M. B. Juric, I. d Rozman, S. Beloglavec and A. Zivkovic, “Object Serialization Analysis and Comparison in Java and .NET,” *SIGPLAN Not.,* vol. 38, no. 8, August 2003. |
| [13] | A. C. Barus, R. Deddy Hasiholan Tobing, D. N. Pratiwi, S. A. Damanik and J. Pasaribu, “Mobile game testing: Case study of a puzzle game genre,” in *2015 International Conference on Automation, Cognitive Science, Optics, Micro Electro-Mechanical System, and Information Technology (ICACOMIT)*, 2015. |
| [14] | I. Astahovs, “Use of design patterns for mobile game development,” 2012. |
| [15] | “Android operating system share worldwide by OS version from 2013 to 2020\*,” Statista, April 2020. [Online]. Available: https://www.statista.com/statistics/271774/share-of-android-platforms-on-mobile-devices-with-android-os/. [Accessed 04 05 2022]. |