

EEE521 Final Year Project

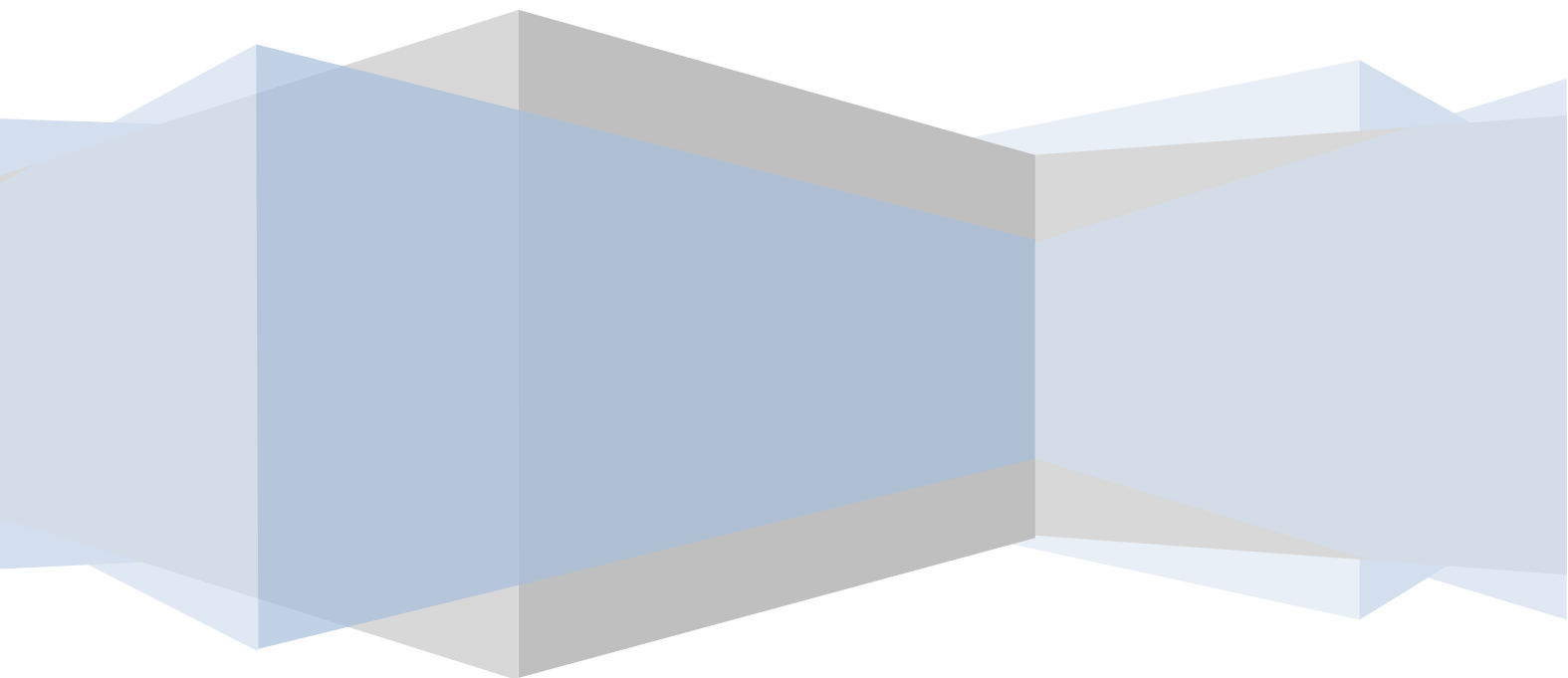
**School of Computing, Engineering &
Intelligent Systems**

Philip Caraher B00699092

**BEng Hons Computer Games Development
Augmented Reality 3D Puzzle Game**

**Supervisor Shane Wilson
Second Marker Dr Pryianka Chaurasia**

21/12/2018

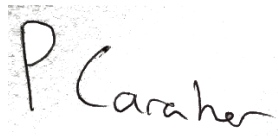


Plagiarism Statement

I declare that this is all my own work and does not contain unreferenced material copied from any other source. I have read the University's policy on plagiarism and understand the definition of plagiarism. If it is shown that material has been plagiarised, or I have otherwise attempted to obtain an unfair advantage for myself or others, I understand that I may face sanctions in accordance with the policies and procedures of the University. A mark of zero may be awarded and the reason for that mark will be recorded on my file.

I confirm that the Originality Score provided by TurnItIn for the interim report is 88%.

As I can only submit once for the final version I can not confirm the final Originality Score.

A handwritten signature in black ink, appearing to read 'P. Caraher', is written over a light grey grid background.

Acknowledgements

I would like to thank my project supervisor and studies advisor Dr Shane Wilson for his advice and guidance.

I would also like to thank Dr Pryianka Chaurasia for their help as a second marker.

Finally, I would like to thank my family for their continued support throughout my time at university and throughout this project.

Table of contents

| | |
|--|----|
| Abstract | iv |
| List of figures | v |
| List of tables | vi |
| 1. Introduction | 7 |
| 1.1 Aims and objectives | 8 |
| 1.2 Literature review | 8 |
| 1.2.1 The growth of Augmented Reality | 8 |
| 1.2.2 Challenges in the design and development of AR games | 10 |
| 1.2.3 AR Puzzle Games; Case Studies | 12 |
| 1.3 Summary of the report | 14 |
| 1.3.1 Analysis Chapter | 14 |
| 1.3.2 Design Chapter | 14 |
| 1.3.3 Implementation and Testing Chapter | 15 |
| 1.3.4 Evaluation and Reflection Chapter | 15 |
| 2. Analysis | 15 |
| 2.1 Introduction | 15 |
| 2.2 Problem Statement | 15 |
| 2.3 Game Overview | 16 |
| 2.3.1 Game Concept | 16 |
| 2.3.2 Game Genre and target market | 16 |
| 2.3.3 Game Features/Challenges | 16 |
| 2.3.4 Gameplay | 17 |
| 2.3.5 Considerations for the game based on critical analysis | 18 |
| 2.4 Functional Requirements | 18 |
| 2.5 Non-Functional Requirements | 19 |
| 2.6 Software Methodologies | 19 |
| 2.6.1 Agile/Programming with Agility | 19 |
| 2.6.2 Waterfall methodology | 20 |
| 2.6.3 Iterative and Incremental Development | 20 |
| 2.6.4 Methodology choice | 20 |
| 2.7 Use Cases | 20 |
| 2.8 Hardware and software requirements | 21 |

| | |
|---|----|
| 2.7.1 Hardware..... | 21 |
| 2.7.2 Software | 21 |
| 2.9 Project Plan..... | 21 |
| 2.10 Conclusions | 21 |
| 3. Design..... | 22 |
| 3.1 Introduction | 22 |
| 3.2 Components and Structure | 22 |
| 3.3 Sequence Diagram | 23 |
| 3.4 Class Diagram | 23 |
| 3.5 UX Design..... | 24 |
| 3.6.2 Taking Account of Industry Guidelines in the Design | 24 |
| 3.6.3 Taking Account of Critical Analysis | 25 |
| 3.6 Conclusions | 25 |
| 4. Testing | 26 |
| 4.1 Unity Profiling..... | 26 |
| 4.2 Technical testing..... | 27 |
| 4.3 Play Testing | 28 |
| 5. Evaluation | 31 |
| 6. Reflection | 34 |
| 7. Conclusions | 36 |
| 8. References..... | 37 |
| 9. Appendices | 40 |
| 9.1 Appendix A – Full Project Plan..... | 40 |
| 9.2 Appendix C - Questionnaire Responses | 41 |
| 9.3 Appendix D - Code Listings..... | 42 |
| 9.4 Appendix E - Implemented Functional Requirements | 48 |
| 9.5 Appendix F - Unity Profiler Screenshots | 49 |

Abstract

Following a review of the literature this project highlighted some of the key problems in the development process of making an AR game. These included specific technical challenges and more general design challenges facing developers in the industry such as the lack of standard guidelines for good UX design. This project explored some of the issues surrounding the guidelines provided by the industry in the context of the development of an AR game; *Match It AR*. A game design document was developed detailing the requirements needed to for the game to be complete, functional, and suitable to properly investigate the challenges outlined above and the game was developed. The game was then play tested in a small group of family members with questionnaires to help evaluate whether certain design considerations are best practice for the specific game and also whether the game was an engaging one for future publishing. The results gained contribute to four issues surrounding UX design, namely; player movement, safety, UI, and interaction. The report makes four recommendations regarding these issues based on both the current project findings and critical analysis of other current AR games.

List of figures

| | |
|--|----|
| Figure 1.1 - Goldman Sachs Report..... | 8 |
| Figure 1.2 - Ofcom, Smartphone Ownership in the UK by Age | 9 |
| Figure 1.3 - AMON AR Sculpture Puzzle | 12 |
| Figure 1.4 - Brickscape AR mode | 13 |
| Figure 1.5 - Knightfall AR, Battlefield in the Living Room | 14 |
| Figure 2.1 - Random 3D Scatter Graph..... | 16 |
| Figure 2.2 - Bejeweled Game Board | 17 |
| Figure 2.3 Use Cases..... | 20 |
| Figure 3.1 Plane Detection and Play Space Initialization Sequence Diagram..... | 23 |
| Figure 3.2 Class Diagram for Match It AR..... | 23 |
| Figure 4.1 Low Poly Torus | 26 |
| Figure 4.2 Original Torus Model..... | 26 |
| Figure 4.3 Profiler Data when a match is made | 26 |
| Figure 4.4 The test scene that was used..... | 27 |
| Figure 4.5 Main Menu | 29 |
| Figure 5.1 Match It AR targets in room..... | 32 |

List of tables

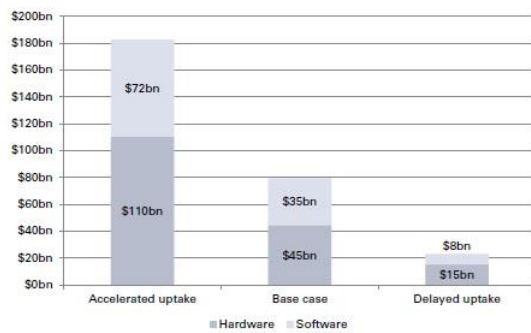
| | |
|---|----|
| Table 2.1 - Game Modes | 17 |
| Table 2.2 - Points Table | 17 |
| Table 2.3 - Hardware Requirements | 21 |
| Table 4.1 Summary of observations | 28 |
| Table 4.2 Questionnaire: What was good or fun | 30 |
| Table 4.3 Was it immersive | 31 |

1. Introduction

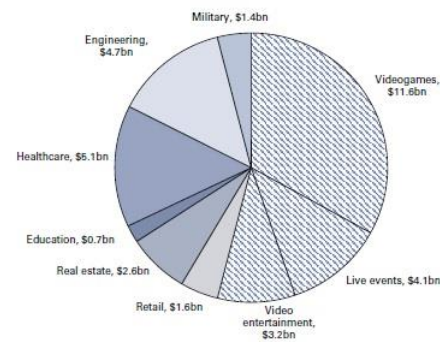
A decade ago virtual and augmented reality were considered by many people to be failed technologies (Liarokapis and Evans, 2018, p89-90 ; Stone, 2009, p1-2). During the later 1990s and early 2000s several companies such as Nintendo with the Virtual Boy and Virtuality with their arcade pods attempted to launch VR solutions for the entertainment industry (Fowle, 2015; Flanagan, 2018). These solutions failed to live up to user expectation for a variety of reasons including high cost, usability, physical discomfort (motion sickness) and poor graphics. (Fowle, 2015; Flanagan, 2018). Early VR medical applications showed some unreliability (Robinson, *et al*, 2006, p57-62) in Boeing's experience with using VR in training aircraft maintenance crews failed because it focused more on the technology than actually meeting the needs of users (Barnett et al, 2000 cited in Stone, 2008). VR/AR research continued over the last two decades with leaders such as Doug Bowman in 3D interaction (Bowman, *et al*, 2004), Mark Billinghurst (Billinghurst and Duenser, 2012, p56-63), and Hirokazu Kato (Kato and Billinghurst, 1999, p85-94). But it wasn't until Palmer Luckey's launch of the Oculus Rift that consumer grade VR got the kickstart into the modern VR/AR revolution. One of the significant strengths of the device meant that the field of view (FOV) was more than double the FOV of previous VR devices from other companies.

Today, relatively low-cost AR/VR solutions are available for consumers thanks to advances in mobile technologies. Phones are smaller and more portable, screens have higher resolutions, processors are smaller and faster, memory capacity has increased and takes up less hardware space, and mobile networks are global. With these advances, AR/VR fields are still at an early stage but Figure 1.1 shows projected significant opportunities for commercial developers.

Many problems and challenges within the field have still to be addressed. Problems with the standardization of guidelines for developers and with its use in games. This project explores some of these challenges in the context of developing a mobile AR puzzle game.

Exhibit 3: Our combined 2025 VR/AR hardware and software scenarios

Source: Goldman Sachs Global Investment Research.

Exhibit 4: Our 2025 base case VR/AR software assumptions by use case

Source: Goldman Sachs Global Investment Research.

Figure 1.1 - Goldman Sachs Report

Goldman Sachs. *Profiles in Innovation - Virtual and Augmented reality*. Statista. Accessed November 20, 2018. Available from <https://www.statista.com/study/38357/virtual-and-augmented-reality-as-the-next-computing-platform-2016/>

1.1 Aims and objectives

The aim of this project was to explore issues surrounding the design of UX for AR games, in particular the lack of standards and guidelines for designers. The objectives of the project include:

- Critically evaluate available literature relating to VR/AR, in particular the design and development of AR games;
- Develop an AR exemplar game which will act as a vehicle to test/evaluate some of the existing informal guidelines and some new ones;
- Based on the findings, make some recommendations relating to guidelines for AR games.

1.2 Literature review

1.2.1 The growth of Augmented Reality

Augmented Reality is the process of augmenting the virtual world onto the real world. This can include augmenting virtual 3D objects into the world to interact with them through the augmenting device per Azuma's definition for AR systems (Azuma, 1997, p355-385), augmenting images onto real world objects (Kato and Billinghurst, 1999, p85-94), and among others, augmenting gameplay onto real world locations such as in Pokémon Go or Ingress Prime (The Pokémon Company, 2018 ; Niantic, 2018).

Technological advances in smartphone sensors and cloud-based computing (Shea, *et al*, 2017, p9619 - 9631) were critical to the resurgence of AR. The growth and development of the field also benefitted from ARToolkit developed by Hirokazu Kato (Kato and Billinghurst, 1999, p85-94). An open source C/C++ library which facilitated easier creation of AR applications. Finally, the commercial success of Pokémon Go demonstrated that the technology had matured to a level where it was practical to market an AR game to lots of people; that there was a demand and that significant profits could be made. Today AR apps are being developed in virtually every industrial sector including Education with the metaverse platform for example (Metaverse, 2018); Sport using the Viz Virtual Studio (Vizrt, 2018); medicine via companies like AccuVein (AccuVein, 2018); military where soldiers are deployed equipped with Augmented Reality Head Mounted Displays (Vergun, 2017 ; Osborne, 2018). A game changer for AR has been the move towards mobile phone-based AR applications. Augmented reality headsets are still quite an expensive enterprising tool and are hardly consumer friendly. On the other hand, very large proportions of, for example, the UK population have a mobile device capable of AR (See Figure 1.2). Mobile devices are the entry point for making AR a mainstream technology with a large share of the market. The debut of Apple's ARKit and Google's ARCore in 2017 bring AR into the mainstream world as so many people now own a smartphone.

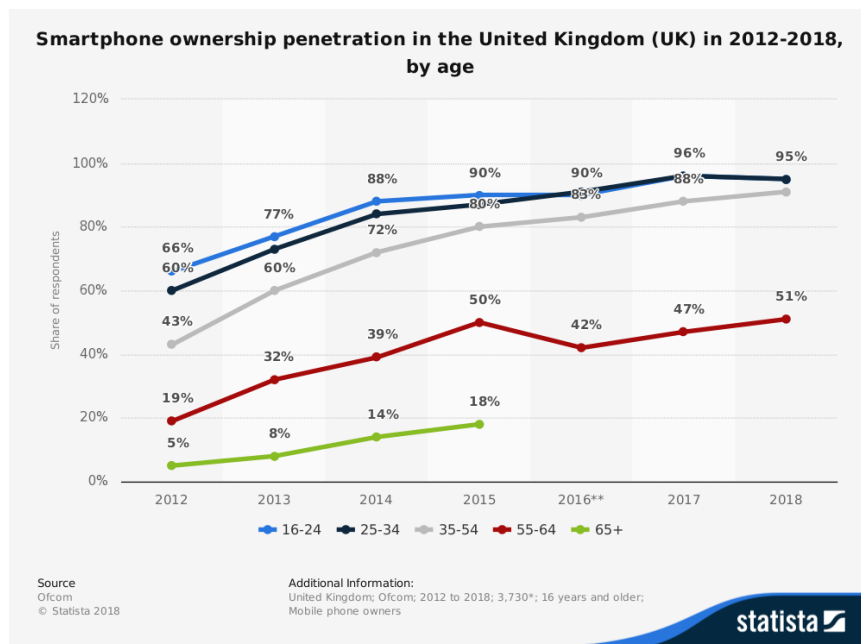


Figure 1.2 - Ofcom, Smartphone Ownership in the UK by Age

Ofcom. n.d. UK: smartphone ownership by age from 2012-2018 . Statista. Accessed November 20, 2018. Available from <https://www.statista.com/statistics/271851/smartphone-owners-in-the-united-kingdom-uk-by-age/>.

Equally significant for the future of AR is the availability of AR Cloud - an open-access digital copy of the real world (Shea, *et al*, 2017, p9619 – 9631 ; Cai, *et al*, 2016, p687 - 691). In essence, the concept of the AR Cloud can be expressed in this quotation;

“The concept dubbed ‘Augmented Reality Cloud’ provides a persistent 3D digital copy of the real world to enable sharing of AR experiences across multiple users.” - Ori Inbar (Inbar, 2018).

As for the significance of the AR Cloud, it will provide a new global platform for shared user experiences allowing multi user collaboration on design, socializing, and playing games.

1.2.2 Challenges in the design and development of AR games

There are many problems and challenges still to be addressed with the use of augmented reality in general and in gaming. Some of these issues will be explored in the planning and development of the current game. General challenges for AR applications using mobile devices include privacy problems (Roesner, *et al*, 2014, p88-96); issues around interfaces (Bowman, *et al*, 2004) and standardization of UX guidelines more generally (Dirin and Laine, 2018, 18pp). In terms of technical challenges, it includes but is not limited to; battery life of mobile devices, the processing power of the device, the size of the screen, and the type of tracking being used i.e. marker or marker less tracking. In terms of UX challenges, the approach to designing applications in AR is different than that of on the traditional 2D screen. Everything should be in the context of a 3D world. Challenges specific to AR games can be found by analyzing innovative AR games in the marketplace (see section 1.2.3).

For an AR game using a mobile device, this raises the issue of user interface and UX challenges. (Bowman, *et al*, 2004) states that 2D elements can be embedded into a 3D Virtual Environment (VE) and that this interaction for users can be significantly easier as there are only two degrees of freedom (2DOF) to manipulate. The developers behind SculptAR (Sangalli, *et al*, 2017, p260-261) also took this lesson as a suitable interface for the purpose of selecting art tools to change the 3D VE. The interface and, to an extent, UX challenges will be explored in the context of this project.

Traditionally in mobile applications there was no ambiguity about what the information on screen was for, the UI was always 2D.

Platform holders and the wider AR industry have yet to agree on a formal and comprehensive set of UX guidelines for AR applications with developers having to use and evolve existing mobile UX guidelines based on ongoing user evaluations from live applications (Dirin and Laine, 2018, 18pp). Google and Apple have defined their own guidelines after the release of ARKit and ARCore (Apple, 2018 ; Google, 2018) which are good starting points but don't offer a standard complete set of guidelines for developers.

Both Apple and Google provide guidelines for Augmented Reality UX design and although they both cover a lot of the same points there are differences. For example, Apple guidelines suggest that haptic feedback is good for the user but Google states that it interferes with the AR tracking on Android. Taking into consideration both Apple and Google guidelines the following was decided upon for the UX design of the game.

Overall the guidelines provide good advice;

- Design the experience for many spaces and environments that aren't optimal for AR by giving the users a clear understanding of the amount of space they'll need
- Remind users that they are in AR and that they can move around in the space.
- Remind the user to be careful and address their safety and comfort.
- Specifically regarding anchoring which are an approximation of a fixed location and orientation in the real world; it's useful to anchor objects that contain additional assets.
- Content manipulation - Allow the users to manipulate objects but favour direct manipulation with the object instead of onscreen controls and set limits on the transformations.
- Have a clear transition into any AR modes and indicate when initialization and plane detection is taking place. It can also be helpful to have the option to quickly reset the AR process.

- Haptic feedback can be helpful but when using ARCore it is not advised because of the variation in technology in Android phones, this can cause the AR tracking to be lost.
- Make use of the entire display on the device and if necessary, provide hints in context of the augmented world. If there needs to be a guide to any of the setup and AR processes, then appropriate terminology should be used as most people aren't familiar with things like plane detection or anchors.

Challenges specific to AR games can be found by analyzing innovative AR games in the marketplace and looking at user reviews. In some ways the obvious game to analyze is the highly successful Pokémon Go; the quickest game to \$1b in revenue (Sensor Tower, 2017). Pokémon Go is a hugely successful AR game; however, it could be argued that the AR camera element is barely relevant to the game (Rapp *et al*, 2018, p1-2; Hamari *et al*, 2018, p1-16). The game is played mostly on the devices screen with user interaction being entirely touch based. A commercially successful AR game with entirely AR gameplay through the camera has not yet been made. An in-depth analysis of Pokémon Go is therefore not included in this report.

Challenges specific to puzzle games are investigated using three case studies of published AR enabled puzzle games that have similarities with the proposed game. These case studies were selected to represent three varying levels of ratings. A below average rated game with a low number of users; a high rated game with a medium number of users; and a high rated game with a large number of users.

1.2.3 AR Puzzle Games; Case Studies

AMON by Lykke Studios (Google Play, 2018) is an AR puzzle game with a similar mechanic to the one proposed in the analysis chapter. On Google Play store it has one rating of 3.0 out of 5. The goal is to line up fragments of a broken statue in order to fix the statue. It was developed soon after the release of ARKit for iOS.

On loading the game, the tutorial is very good at showing what the goal



Figure 1.3 - AMON AR Sculpture Puzzle

is. Interaction includes a 2D UI for selecting levels and on each level load the player has to select the spot to place the anchor for the game to play on. However, this has to be done every level which takes players out of the AR part of the game every time. Intuitive controls, the only thing the player has to do is move around the statue to align the pieces together and it's immediately obvious. Could be argued that a game like this couldn't be as big as Pokémon Go because of the requirement for space to play, Pokémon Go takes place mostly on the devices screen. Possible repercussions for the design of the proposed project are the game needs to be scalable to play in smaller spaces. User interaction with the environment is minimal, the player must move their device and their body to complete the puzzles, no other interaction, is this better for an AR game? The screen space is left almost entirely to the camera with the only UI element navigating the player back to the level select screen, this is good for immersion, therefore other UI on the screen could be intrusive.

Brickscape by 5minLab (Google Play, 2018) is a puzzle game with AR capabilities. The core game is played on the screen but there is an AR mode. On Google Play store it has an average rating of 4.6 out of 5 from 5,752 ratings. The goal is to move bricks within a cube to get a special brick out via the exit. The AR mode of this game is worth investigating because of its scale and how the player interacts with the puzzle in AR. In terms of scale, the playing area could fit onto a desk or a table on a plane so it doesn't require a lot of room to play. The cube is interactable through the screen; it can be manipulated and rotated, the player doesn't have to move their device to

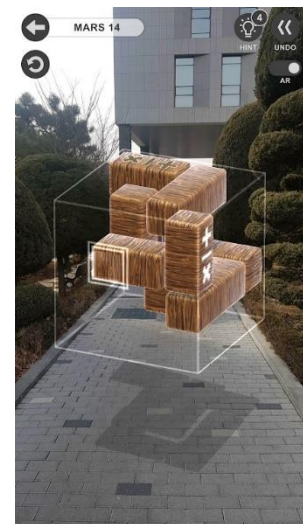


Figure 1.4 - Brickscape AR mode

get different perspectives. Even though the cube is augmented, is the gameplay in augmented reality if the player is doing all the interaction on the devices screen? The games AR mode feels the same as the regular game but harder as the device moves when the player swipes the screen therefore the camera view moves as well. As for what this means for the proposed game; Any interaction with the screen during gameplay must avoid excessively shaking the camera as the proposed project has an element of accuracy in the gameplay.

Knightfall AR by A&E Television Networks Mobile (Google Play, 2018) is an AR strategy game for mobile. The player scans a surface to lay the battlefield on to, they have the option to manipulate the battlefield before starting



Figure 1.5 - Knightfall AR, Battlefield in the Living Room

the game by adjusting the scale of the battlefield and the rotation of the battlefield. This game is worth investigating because of this kind of setup and also the gameplay. The setup gives the player a choice; they can increase the scale of the battlefield and become more active with the game because now they have to walk around the battlefield to interact with all of the elements. Or, they can keep the scale at minimum to play from a stationary position. The interaction with the game is done by aiming at the incoming soldiers using the camera, and pressing buttons on the screen to fire arrows/catapults. The player can also improve their defences in between rounds. This is not a puzzle game, but the interaction with the game through UI doesn't seem intrusive as it is a strategy game board. The game is also played with the device in landscape orientation so the interaction with buttons doesn't affect the accuracy of the camera much because of the extra stability.

1.3 Summary of the report

1.3.1 Analysis Chapter

As a result of the critical analysis in the literature review a problem statement will be presented and following that will be an outline of a game and its requirements to explore a solution to the problem. Also, different software methodologies will be considered and a choice made for which one will be utilized in the development of the proposed game with reasoning.

1.3.2 Design Chapter

The design chapter will lay out design decisions as a result of the functional requirements analysis in the analysis chapter. It will also describe in detail the

architecture of the game and present visual descriptors to help understand the structure of the game including UML Class diagrams. The key challenges of development, along with the specific and general design challenges related to AR puzzle games, will be described as well.

1.3.3 Implementation and Testing Chapter

In this chapter the tools, languages, and anything else that was used in the implementation will be discussed with reasoning for their choice. The development methodology will be discussed with mention to its effectiveness for this project. The testing methods will be described along with their appropriateness for the project with all relevant testing documentation in the appendices. The testing method will involve user questionnaires on various groups to get feedback on the design and how engaging/intuitive it is.

1.3.4 Evaluation and Reflection Chapter

This chapter will evaluate the results from the testing in the previous chapter and reflect upon mistakes during the project and what could be done differently. The overall value of the project will be critically evaluated with reference to the requirements in the analysis chapter.

2. Analysis

2.1 Introduction

The analysis chapter will define the functional and non-functional requirements for the game. An overview of the game design will be presented including the features and challenges of the game along with an analysis of different software methodologies and their appropriateness for the proposed game.

2.2 Problem Statement

Currently the AR industry is approaching the mainstream with great successes and new technology allowing many developers to build AR applications. But what are the specific technical challenges involved in the making of an AR game? This project will explore these challenges in the context of an AR action puzzle game; Match It AR.

There are also a number of more general design challenges facing developers, including the lack of standard guidelines for good UX design. This project also explores the effects of different design choices in the context of this specific genre.

2.3 Game Overview

2.3.1 Game Concept

Match-it AR is a game in which the player generates targets in 3D space which they can view through their phone's camera, and move around to line up targets and match. Once you choose where you want to play then jump in and start exploring all the possible angles to match gems and get points until time runs out. Using Augmented Reality, Match-it AR allows you to interact with the world in 3D through your device and challenge yourself to change your perspective in order to win.

2.3.2 Game Genre and target market

Match it AR is an action puzzle game using Augmented Reality that will appeal to casual mobile gamers who want something new and interesting with replayability. It will appeal to those who enjoy a challenge that can be solved creatively. Match it AR needs to explore the challenges of making camera AR replayable and easy to use for mobile gamers. Creating an action puzzle AR game will provide insights into the challenges of player retention and ease of use.

2.3.3 Game Features/Challenges

The game generates a 3D scatter of targets (Figure 2.1) augmented on the point of the tracked plane that the player chooses. The scatter/play area should be scalable to accommodate for many different spaces which is recommended in the industry guidelines. The game will have multiple competition modes (Table 2.1). The device

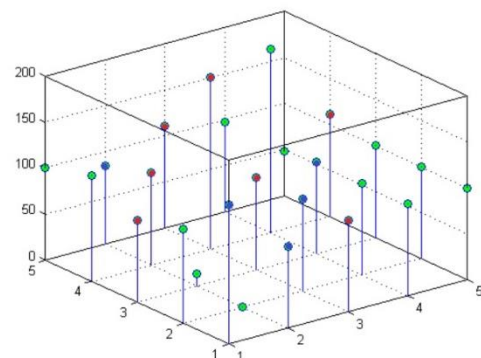


Figure 2.1 - Random 3D Scatter Graph

orientation will not be fixed. If the player doesn't make a move for a certain period of time then the positions of some of the targets will randomly change.

Table 2.1 - Game Modes

| Time Trial Mode | Marathon Mode |
|--|--|
| They must get the required score before time runs out. | Endless mode that saves the players score so they can play with no time limit. |
| Sets the scene for a level-based game for further development. | Intended for casual play |
| The levels vary in difficulty based on the points required, the time left, and gameplay variations e.g. the frequency and speed of targets moving. | |

2.3.4 Gameplay

The gameplay can be described as similar to that of Bejeweled or Candy Crush (Google Play, 2018). It is a match three game mechanic but, in a 3D, augmented reality game. The augmented part of the game is the play area where the interaction with the game objects takes place. Within this play area, the player matches targets much like in Bejeweled, by lining them up in a row of three or more, but in this case the line-up is done using a raycast from the camera. If there are three or more same type



Figure 2.2 - Bejeweled Game Board

targets in contact with the raycast at the same time then a match can be made through some interaction and the player gets points (Table 2.2), there is no penalty for missing or hitting incorrect targets. Once the lined-up targets are destroyed, new ones are spawned in the scene at new positions and the points are added to the player's score.

Table 2.2 - Points Table

| Match 3 | Match 4 | Match 5 |
|------------|--|---|
| 300 points | 400 points Combine into explode target at the midpoint coordinate | 500 points Match any targets for the next 10 seconds |

2.3.5 Considerations for the game based on critical analysis

The shooting mechanic will be explored and tested to evaluate an appropriate method within the context of an AR action puzzle game.

The orientation of the phone during gameplay should be considered. Two-handed use can give better accuracy for an aiming mechanic but doesn't allow as much movement as one-handed use.

Could an adjustable scale give a better experience and should the devices field of view restrict the upper scale?

Where the player decides to play. With this specific kind of game there should be explicit instructions on where it should/can be used as it can't be played in every context.

2.4 Functional Requirements

Launch - The game should launch without issue.

Load into main menu - The game should load into a home screen where the player can choose game modes and change settings. It should be clear and concise.

Start game - When the player selects start game the scene will change to the surface scan mode where the player will find scan their environment. After scanning, the game will generate the virtual gameplay scene and overlay it on top of the real-world feed from the camera.

Allow the player to aim using the camera - The player should be able to clearly see where they're aiming and the trajectory of their aim.

Allow the player to shoot - The player should be able to perform some action to match the targets they're aiming at.

Making a match - The player should receive points for making a match of three or more targets.

Targets - Targets will be spawned in random places with equal numbers of each type. And when they are matched, will destroy and respawn at a different position in the scene.

Win Condition - If the player reaches the required score for the level before the timer runs out then they complete the level.

Lose Condition - If the player runs out of time then they lose and the game is over.

Timer - The player should be able to see how much time they have left.

Objectives - The objectives should be made clear to the player, through something similar to a tutorial.

Directions - Directions for how to use the game should be conveyed effectively.

Music - Music should be appropriate for the context of the scene i.e. Main menu and gameplay.

2.5 Non-Functional Requirements

Privacy - The game will have analytics working in the background to track the players interactions with the game. This means that the game must be compliant with privacy laws, for example General Data Protection Regulation (GDPR) (Unity, 2018). To use game analytic services, the developer and the player must consent to the collection of personally identifiable data.

Emotional - The game should fulfil the player emotionally i.e. be fun.

Performance - The game should perform smoothly and efficiently use resources. Be able to detect surfaces efficiently and function as a game equally as efficiently. Good frame rate on mobile or else it will feel less real.

Response Time - There should be no more than a 0.5 second delay between the user interaction and the games response i.e. the player firing and the game detecting/destroying the targets.

User Interface - The UI is being investigated as part of this project and the effects on gameplay. Should it be included, as a non-functional requirement, it should be minimal on the screen during gameplay. Any large UI will obscure the view of the world.

Safety - As the game encourages moving around to explore different perspectives of the play area, the game must not cause any harm to the player.

2.6 Software Methodologies

2.6.1 Agile/Programming with Agility

The Agile software methodology, or programming with Agility (Thomas, 2014), is a method where software is created in iterations to minimize risk as there is an emphasis on early iterations of the software. Within Agile development the core idea is to build iterations of the software, and using what the team learns from the previous iteration to improve the next (Beck, *et al*, 2001).

2.6.2 Waterfall methodology

The Waterfall methodology is a traditional approach to software development. It is linear and sequential i.e. follows a defined order of stages of development; Requirements, Analysis, Design, Implementation, Testing, and Deployment (Royce, 1987, p328-338). The Waterfall methodology works for projects that need management oversight and that work on deadlines for each stage. Therefore, there are no iterations on the software. This is a disadvantage in the case of this project where iterations and revision are key.

2.6.3 Iterative and Incremental Development

Iterative and Incremental development is an agile approach with smaller iterations on the software. The approach splits the functionality of the system into manageable chunks where bugs can be resolved on separate parts of the project before moving onto the next functionality to be implemented.

2.6.4 Methodology choice

For this project which will be investigating design challenges and will require iterations with design alterations, the choice of methodology will be Iterative and Incremental Development. This choice is flexible and best fitted for a project that will undergo multiple iterations with user testing for feedback on design considerations.

2.7 Use Cases

The functional requirements can be presented as use cases in Fig 2.3. The player can start the game, they then enter the setup phase where ARCore starts getting trackable planes. After the setup phase the player can use the camera on their device to aim and when they align their perspective correctly, they can match targets.

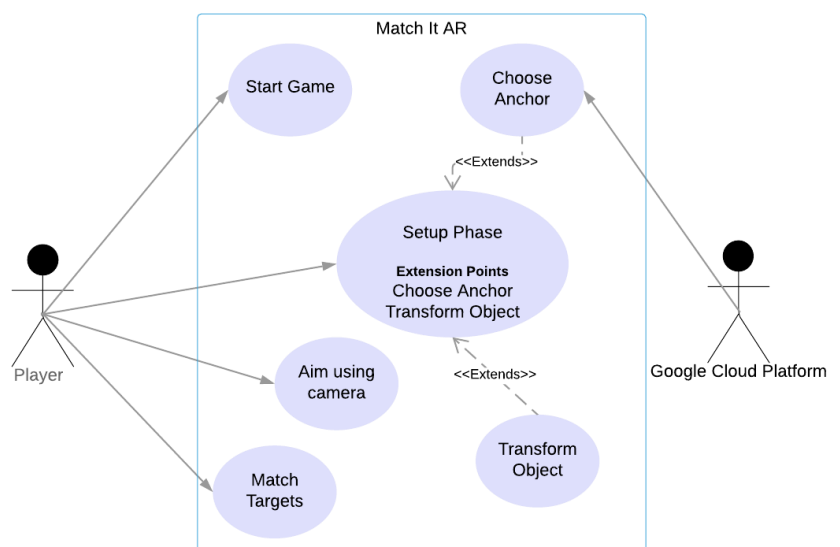


Figure 2.3 Use Cases

2.8 Hardware and software requirements

2.7.1 Hardware

The game will be developed on a desktop PC with the following specifications (Table 2.3). A PC with these specifications will be more than capable to turn out multiple iterations of the game in a timely fashion, appropriate for testing multiple iterations;

Table 2.3 - Hardware Requirements

| | |
|-----------|-----------------------------|
| OS | Windows 10 Enterprise |
| RAM | 32.0GB |
| GPU | NVIDIA GeForce GTX 1080 8GB |
| Processor | Intel Core i7-6700K 4GHz |

2.7.2 Software

The following software will be used to create and test the game;

Unity 2018, Visual Studio IDE - Used for the development of the game. Unity has been supporting AR on its platform for a long time and supports the ARCore SDK. The Visual Studio IDE is integrated easily with Unity and is a lightweight and easy.

ARCore SDK for Unity - The ARCore SDK is available from Google and comes as an easy to install package for Unity.

A smartphone with Android 7.0 or higher - A full list of ARCore supported devices is available on the developer's section of the ARCore site (Google, 2018). For the purposes of this project, a Huawei P20 Pro will be used to test the game on a device.

ARCore framework for android - For ARCore applications on Android, the device needs to have the ARCore framework installed in order to run the application. The framework for Android can be downloaded from the Google Play store.

2.9 Project Plan

The project plan outlines the timeframe for the project. The project has been split up into manageable chunks and shows the project deliverables. The full project plan is available in the appendix A.

2.10 Conclusions

Through the definition of the functional and non-functional requirements in this chapter it will inform the steps to take in the next chapter. Per the chosen design methodology, the next step will be to lay out the structure of the game and its underlying components.

3.Design

3.1 Introduction

The Design chapter will present the technical design process for the game. The chapter will define the system structure and components using a sequence diagram and a system class diagram. Specific UX design considerations from the analysis chapter will be considered. Some questions for testers will be identified which will eventually form part of the evaluation.

3.2 Components and Structure

The core components of the system have been identified from reviewing the functional requirements and this is how they will generally interact.

ARCore Device

The ARCore device holds the ARCore session config file and the AR Camera. The AR Camera renders virtual objects and overlays them on top of the camera view to make it look like there are in the real world.

AR Controller

The AR Controller is the setup phase of the game. It detects the trackable planes and creates a visual representation of the planes. Once the player selects a point on a tracked plane it places the play space object and starts the Loader script.

Loader

The Loader generates the targets after the play space anchor is placed. Generates based on the scale of the parent GameObject, so when the player adjusts the scale it will change the number and size of the targets.

Level Controller

Controls the objects in the play area, for example, the spawning of new targets and the Physics.RaycastAll script to get matches. Also controls the timer depending on the game mode.

3.3 Sequence Diagram

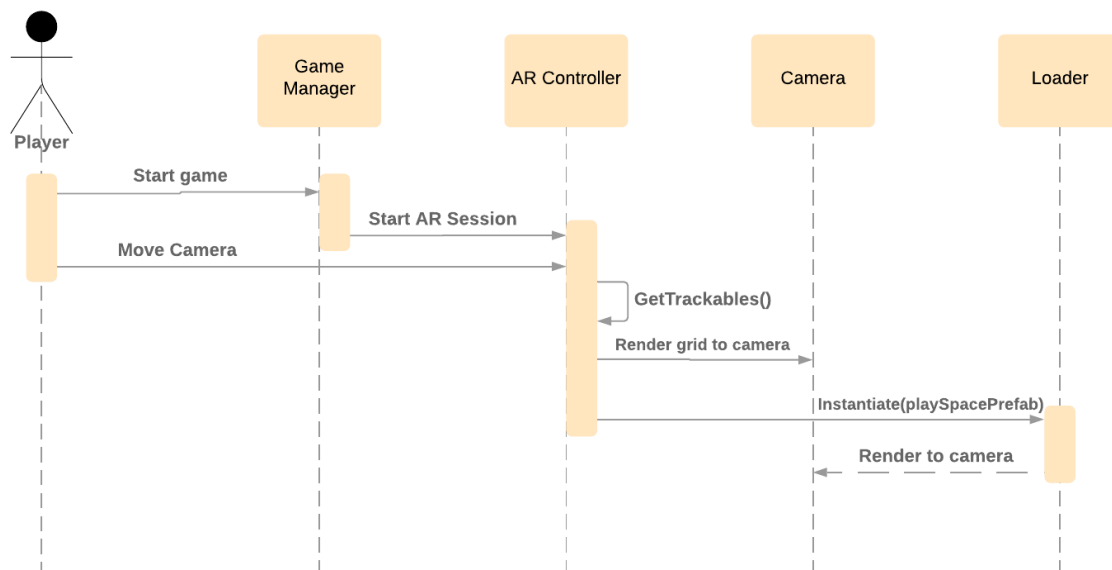


Figure 3.1 Plane Detection and Play Space Initialization Sequence Diagram

The player starts the game mode and this creates a game manager object. The game manager starts the AR session. When the AR session starts the controller gets all the trackable planes detected by the AR camera, stores them and renders them to the camera. The player needs to move the camera in order for the AR controller to detect new planes. The player then selects a point on the tracked planes to instantiate the play space object.

3.4 Class Diagram

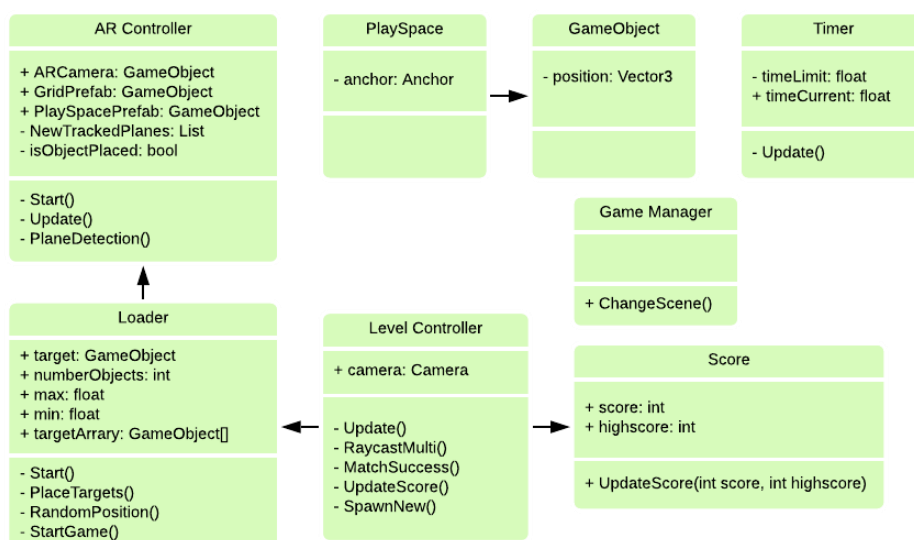


Figure 3.2 Class Diagram for Match It AR

The class diagram shows the C# data structure for Match It AR. The AR controller will take care of the plane detection and whenever the PlaySpacePrefab is instantiated the Loader script takes care of the target generation. When the player is finished with the setup the Level Controller will handle functions needed for gameplay to function.

3.5 UX Design

Each of the general design guidelines developed based on Google/Apple materials are worth discussing in turn and how they will be addressed in the game design.

3.6.2 Taking Account of Industry Guidelines in the Design

Physical Environment - The game will be suitable for many environments by allowing the user to scale to the size that they need. Once the player selects their anchor point whether it be on the kitchen table or outside in their garden, the game will be suitable for the environment.

Encourage Movement - For players that have never used AR before, they need to be shown that they can move around the AR play space. This can be done by placing something that can only be reached by moving there or having a goal that can only be accomplished by moving i.e. changing perspective to see things differently.

Safety & Comfort - Whenever a game encourages the player to move about there is an element of safety that the designer is responsible for. The game will encourage the user to be mindful of their surroundings and refrain from any large and sudden movements through an onscreen message before entering the AR session.

Anchoring - The play space will be parented to an Anchor object. Anchors are an ARCore object that ensure that game objects appear to stay in the same place in the world as the AR session persists. Making the play space a child of an anchor is recommended as the play space is not a static object, it changes throughout the game.

Content Manipulation - The play space, once placed, can be manipulated by the player to suit the environment they are playing in. They can rotate and scale it directly instead of using onscreen buttons.

Initialisation - The transition into the AR mode is important as it needs to be clear to the player what is happening. The scene transition will be gradual using a fade and

then an onscreen graphic indicating that the device is scanning will be displayed on the players screen.

Engagement - The guidelines suggest that UI should be kept to a minimum and to avoid cluttering the screen with controls and information as it will diminish the immersive experience. This seems good practice. The game will only have a single reticle on screen. There will be no buttons and no information displayed on the screen.

3.6.3 Taking Account of Critical Analysis

Using the insight from the critical analysis in the literature review, below are some design features that the game will implement. These will also form the basis for some of the questions in the play testing for the evaluation.

- It would be best to keep players in AR and not break the immersion by having frequent interactions with the device through UI. There needs to be some form of UI to return to the main menu, so that will be the only UI visible during gameplay. For instance, a small, low opacity button in the corner or just keep the devices back button active.
- The interaction with the game will be through the positioning of the device and not require any interaction with the devices screen. This will hopefully give the player a sense of interaction with the world rather than the device. The main menu will have a regular interaction scheme of buttons.
- For this particular game, one that requires an element of accuracy, as the player will be aiming the camera on their device. It should be recommended that the device be used in a landscape orientation for maximum stability. This orientation should also make up for any camera shake if there has to be any screen interaction during gameplay.

3.6 Conclusions

The structure of the game's system in terms of its components, classes, and objects through the sequence diagram, and a class diagram has been laid out. UX design guidelines from Google and Apple have been taken into account in the design of the game. Specific design issues raised by the critical analysis from the literature review chapter have also been taken into account.

4. Testing

4.1 Unity Profiling

Unity profiling showed what parts of the game if any were resource intensive and could be improved. In an effort to reduce the cost of rendering the targets, they were made into a low poly torus with 81 vertices and 114 triangles.

From profiling, it was also found that that destroying game objects and re-instantiating them takes time that could be reduced. When a match is made, the particle system is instantiated then destroyed and two



Figure 4.1 Low Poly Torus



Figure 4.2 Original Torus Model

targets are also destroyed before two new ones are instantiated. The reason they are destroyed is because the new targets will be random colours according to the generation script, so they shouldn't be deactivated and then reactivated. An alternative to destroying the targets could have been to deactivate them and before reactivating them, change the materials colour and the objects tag. Deactivating an object keeps its memory allocation but no longer uses CPU time, so it doesn't create any work for the Garbage Collector. The profiler showed that whenever a match happens there is a spike in the generator script usage and after investigating it there appears to be a lot of GC Allocations(ref) which could be slowing down the update speed.

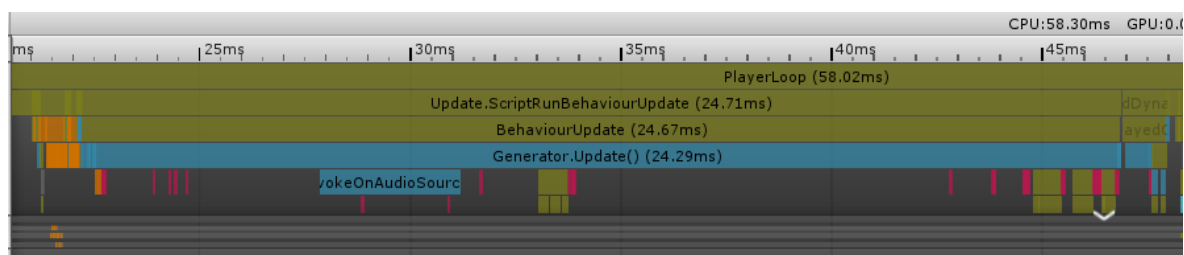


Figure 4.3 Profiler Data when a match is made

With the amount of code that is running while playing the game there is no significant hit on performance and it appears to players to run smoothly. Additional screenshots taken at various events can be found in Appendix B.

In the ARController script the detected planes were stored and visualised using a grid prefab. The prefab was a simple cube with a mesh provided by Google ARCore. When

the game would start in early prototypes these objects would remain in the scene and would be rendered by the camera. Now in the current prototype, the detected plane list is emptied and the grid objects are deactivated to save on rendering costs.

4.2 Technical testing

The summary of the system for the game is short and is as follows; The player starts the game and the ARCore session begins tracking. The player selects a point on the tracked planes to instantiate the play space prefab. The targets are generated in unique positions randomly. The Update function of the generator script uses raycasting to check for targets from the centre of the screen. Once the player lines up two targets the code uses the raycast to determine if they are the same type/colour and then removes them and generates two more in different positions.

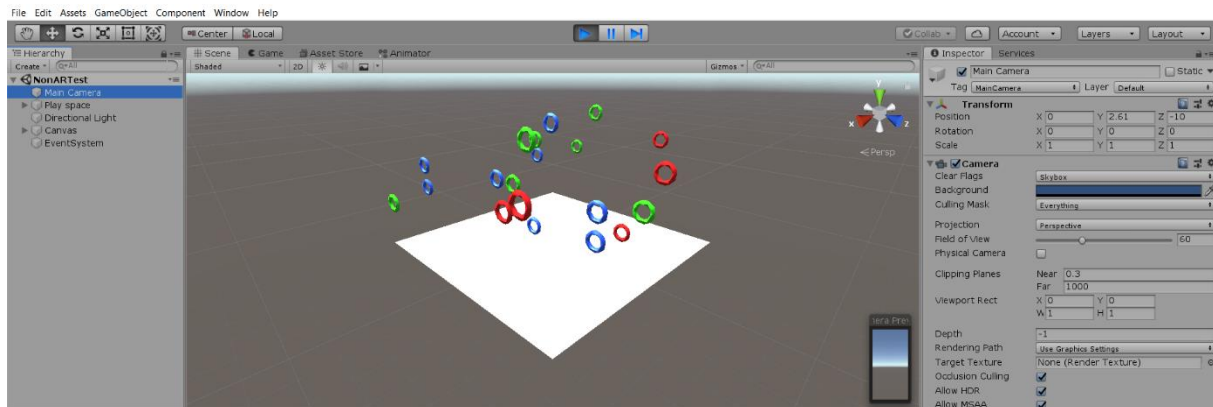


Figure 4.4 The test scene that was used

Developing the code for an AR game can be confusing when compared with the traditional Unity method of making objects in a scene. Therefore, trial and error was used to test the code along with creating a test scene in Unity in which the world that was going to be generated in AR was first tested in a 3D scene. It was helpful to see the world in the unity editor before moving it to the AR code as it could be confusing at times when testing the game in AR. It was easier to see what was causing the errors.

4.3 Play Testing

Play testing was limited by the software requirements of the players phone. Some of the usual methods of surveys of user testing were therefore difficult for an AR project such as this. In-depth play-testing was carried out on four family members to observe how they played the game and to collect detailed feedback on their views on the game and UX features. A spacious room with a very large clear space was used for testing. Two males and two females took part. One male and one female had played AR games before. The others had never played anything other than traditional mobile games. Table 4.1 shows the observations recorded. Two interventions were made as they played. If they had not changed their phone orientation since they started the game this was suggested and the player was asked which they preferred. If they had not pinched the screen to scale or change the view this was demonstrated. Players were asked a series of questions following their playing experience and the exact questions and the answers are listed in Appendix C.

Table 4.1 Summary of observations

| | Player 1 | Player 2 | Player 3 | Player 4 |
|--|----------|----------|----------|----------|
| Did they need help with the setup? | N | N | N | N |
| Did they need help with playing the game? | Y | Y | N | N |
| Did they start to move around without being prompted? | N | N | Y | Y |
| Did they touch the screen to scale it? | N | N | N | N |
| Did they touch the screen at all? | N | Y | Y | N |
| Did they walk backwards while looking into the camera? | Y | Y | Y | Y |
| Did they look for buttons or a menu? | N | N | N | N |
| Did they change the phone orientation? | N | N | N | N |
| Time to first mine clearance | >5 mins | ~ 2 mins | < 1 min | < 1 min |

Did players move around unprompted?

Guidelines suggested that players should be encouraged to move around however for this game it was tested whether players would do so intuitively. It's very difficult to play this game without moving around and it was of interest to see if this was a necessary

instruction. Two players moved around unprompted and two players had to be prompted. Therefore this instruction is probably necessary.

Was landscape or portrait orientation better?

Two people preferred landscape orientation as both users felt it gave them more control of movement, one preferred portrait orientation and one had no preference. This was inconclusive.

Did minimizing interaction with the screen keep the player focused on the AR aspect of the game?

No context was given to suggest that the player could manipulate the scale and rotation of the scene. None of the players tried to do this. When this function was pointed out, three of the users continued to ignore it and move around the game crouching and angling the phone as before, sitting on the floor etc. However, one user immediately removed himself from the AR environment and stood static clearing targets by rotating and scaling the targets on the screen.

Did they need help with the setup?

The guidelines suggest using illustrations and animations for plane discovery. This project tested whether simple text in the main menu could achieve the same. Nobody needed help with the setup as they followed the instructions literally and were able to scan to create a grid and generate the targets. However, they all started from a sitting position looking at the screen as they would on a traditional mobile game and with the camera pointed directly at the ground at their feet and moving the phone in a circular motion. So, the instructions were easy to follow but the outcome wasn't as intended and all did in fact have to be helped to position the grid in the centre of the room. So this was unsuccessful with the current main menu text shown in Figure 4.5.

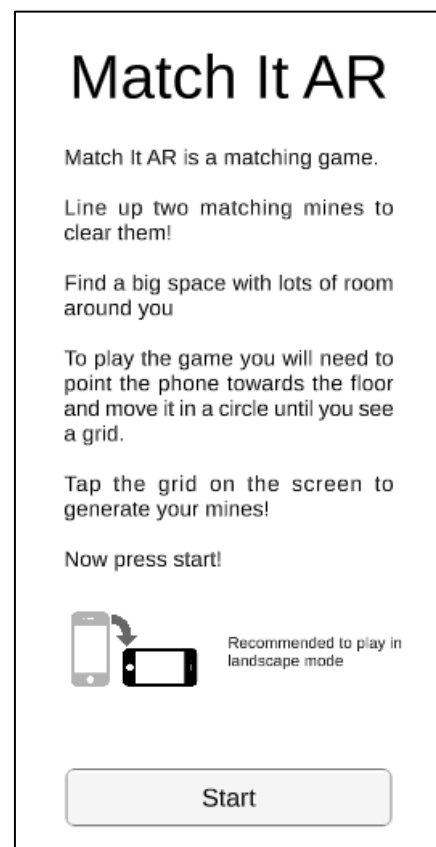


Figure 4.5 Main Menu

Did they need help to play it?

The two users who had played AR games before did not need any help playing the game and they started to move around without being prompted. The two novices needed help with the game and needed to be told to move around. Nobody touched the screen to scale it unprompted and two players touched it for other reasons. Nobody looked for or asked for buttons or a menu. Three of the players cleared their first mines within 2 minutes. One player struggled with the game and did not clear any mines for more than 5 minutes. In terms of needing help to play it, the results were mixed.

Did they walk backwards while looking into the camera?

Everybody walked backwards constantly while looking into the camera. Within the guidelines it is suggested that the developer avoid creating situations where the user has to walk backward while looking into the camera. However, this is a requirement for this game and others like it. The implications of this guideline will be discussed in the evaluation.

Did they look for buttons or a menu?

The current game had only a single reticle on screen, a small, low opacity white circle. There were no buttons and no information. Guidelines suggest this is good practice for immersion. Did this work? It seems so. Certainly, players didn't look for buttons or menus and had no difficulty playing the game once they were familiar with the objective.

Was it fun?

Players were asked in the questionnaire about the positive aspects of the game and responded as follows in tables 4.2 and 4.3. The feedback was very positive.

Table 4.2 Questionnaire: What was good or fun

| |
|--|
| What was good or fun about playing a phone game like this? |
| <i>It's fun for me because I never played that kind of AR game before. It's interesting to walk about the room and go for different heights and angles.</i> |
| <i>It was quite fun after the first one to try and rotate the screen to get them to be superimposed upon each other and I liked the noise it's a nice little tinkly magic sound. It was satisfying particularly because you had to do the fine movement and then you got the reward of the little sound.</i> |
| <i>Well it was challenging and there was a reward when you got the nice...when you got it done. Because it was tricky, when you started to hit targets that was enjoyable because you got a bit of a reward. I think what would make it better would be if it was a bit easier.</i> |
| <i>It was cool. I mean seeing them hanging there suspended was so cool. I've never done anything or played anything like that before and it was like being in bubble land. It was mesmerizing.</i> |

Table 4.3 Was it immersive

| |
|--|
| Was it immersive? |
| <i>Yeah. I had to move about the room I wasn't sat still using my fingers. I had to use my body to get the technique.</i> |
| <i>Well I was focused on it yes.</i> |
| <i>Well yeah you get hooked on it. If it was a little easier to score points I could probably get immersed in it. It works definitely.</i> |
| <i>Yeah I was right in there with the bubbles.</i> |

5. Evaluation

A decade ago virtual and augmented reality were considered by many people to be failed technologies. Today, relatively low-cost AR/VR solutions are available for consumers thanks to advances in mobile technologies. Phones are smaller and more portable, screens have higher resolutions, processors are smaller and faster, memory capacity has increased and takes up less hardware space, and mobile networks are global. With these advances, AR/VR fields are still at an early stage but projected opportunities for commercial developers are significant.

For an AR game using a mobile device, this raises the issue of user interface and UX challenges. Platform holders and the wider AR industry have yet to agree on a formal and comprehensive set of UX guidelines for AR applications with developers having to use and evolve existing mobile UX guidelines based on ongoing user evaluations from live applications. Google and Apple have defined their own guidelines after the release of ARKit and ARCore which are good starting points but don't offer a standard complete set of guidelines for developers.

The aim of this project was to explore issues surrounding the design of UX for AR games, in particular the lack of standards and guidelines for designers. The objectives of the project include:

- Critically evaluate available literature relating to VR/AR, in particular the design and development of AR games;
- Develop an AR exemplar game which will act as a vehicle to test/evaluate some of the existing informal guidelines and some new ones;
- Based on the findings, make some recommendations relating to guidelines for AR games.

The exemplar game

The exemplar game is similar to that of Bejeweled or Candy Crush. It is a match three game mechanic but, in a 3D, augmented reality game. The augmented part of the game is the play area where the interaction with the game objects takes place. Within this play area, the player matches targets much like in Bejeweled, by lining them up in a row of three or more, but in this case the line-up is done using a raycast from the camera. Once the lined-up targets are destroyed, new ones are spawned in the scene at new positions.

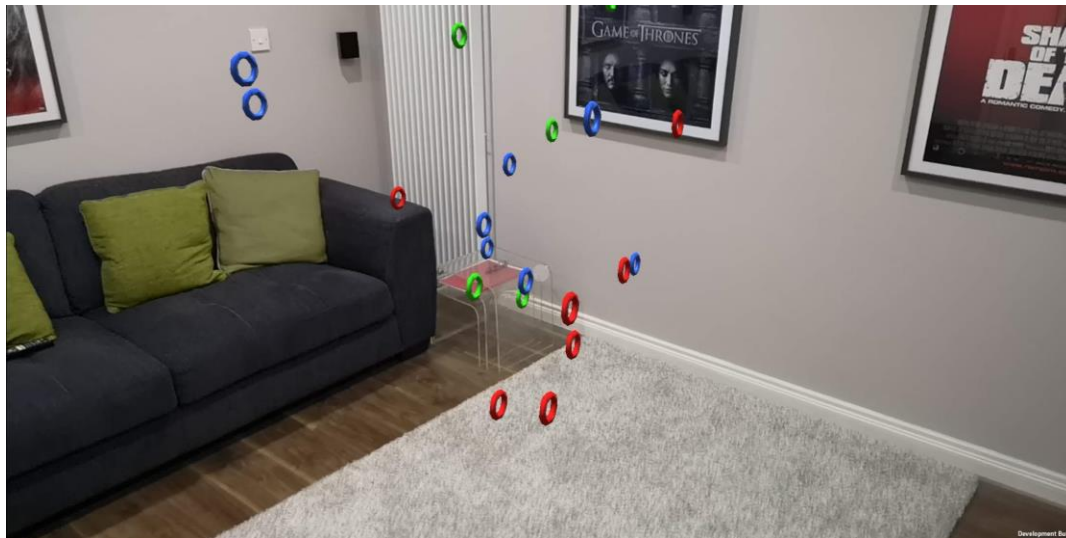


Figure 5.1 Match It AR targets in room

What worked in the technical development of the game? What didn't work?

In Appendix E are the functional requirements laid out in the analysis section of the report. In order to make the game totally functional these features should have been implemented. Not all requirements were met however and the core requirements became the focus.

The game does implement most of the functional requirements but the definition of them changed throughout the implementation. For instance;

Load into Main Menu - The game loads into a Main menu with instructions for use and a start button.

Making a Match - The player can make a match but they can only match two targets at a time and they do not receive points.

Targets - The targets are spawned in unique positions randomly with a random number of each type. They can be matched and destroyed. New targets will spawn somewhere else.

Directions - The directions were conveyed but not in an effective way and didn't yield an ideal outcome.

In the initial design, the Components and Structure were defined in four parts; The ARCore Device, AR Controller, Loader, and Level Controller. This was not the approach taken in the implementation of the game. Instead the structure was made relatively simple; ARCore Device, AR Controller, and Generator. The functionality of the Loader and Level Controller were combined into the same component, so the Generator does almost all of the gameplay code at runtime.

What existing and potential guidelines worked? What didn't work?

The existing guidelines suggest that it is important to remind players that they can move around in AR games. This seemed a slightly odd way of looking at things. It would make more sense to try and design games where moving around is intuitive rather than delivering an instruction. As AR games are increasingly available it may be that the need for this instruction becomes redundant as people become familiar with such games. For the current project it was decided not to include such an instruction and to see if players moved around unprompted as it was extremely difficult to play the game without moving. Four family members agreed to take part in an informal play test. In the play test two of the four players spontaneously moved around and two had to be instructed so there may be some potential in this approach but it was certainly not completely successful.

Existing guidelines also stress safety issues and specifically that players should be instructed not to walk backwards while looking through the camera. While safety is clearly paramount this poses a dilemma. The current game and other similar AR games require players to walk backwards (and sideways and to duck and weave) in order to play the game. It is simply impossible to play unless this can be done. Again, this focus seems a slightly odd way of looking at things. Given the huge potential of AR games this would be a significant limiting factor. The focus of safety should perhaps be much broader - on how to play these games safely and not on the need to provide a specific warning. Should 'Room Scale' games be a specific genre? Or 'Party Games'? Should there be a rating for games according to the style of game or space that they require? The current game was tested in a spacious room with a very large

playing space. As expected, all players moved backwards constantly while looking through the camera. This was not a test of a specific guideline so much as a demonstration that another way should be found to deal with this issue.

One of the case studies in the critical analysis focused on minimizing interaction with the screen and it seems likely that this would help keep the player focused on the AR aspect of the game. There were few user reviews of this game and it seemed worth testing in the current game. This provided some interesting results. No context was given to suggest that the player could manipulate the scale and rotation of the scene. None of the players tried to do this unprompted. When this function was pointed out, three of the users continued to ignore it and move around the game crouching and angling the phone as before. However, one user immediately removed himself from the AR environment and stood static clearing targets by rotating and scaling the targets on the screen.

6. Reflection

The technical difficulties in developing an AR game of the type initially envisioned were significantly more time-consuming than anticipated. Technically speaking, the process of generating the targets is not complicated but I had to go through a couple of methods of generation before deciding on the current version. It was initially intended that the position of the targets be decided totally at random but that a CheckSphere physics object would make sure that all of the targets were spaced out and not overlapping. This proved difficult to understand and I would like to go back and try it again after learning from the current method of generation.

During the process of matching targets, I decided to go with matching two instead of three in the interest of not complicating things and getting a working version going. The unity Physics.RaycastAll raycast returns all objects that the ray hits as an array, but it doesn't necessarily return them in order of first hit to last hit. So I thought it might be unreliable and after testing, it showed that it might have been. Also, I chose to go with matching two targets for a more fast paced mechanic, as matching three targets may take the player much longer than just two.

There were ways to overcome these difficulties but certainly the final game was not what was initially planned. The final product was a reasonable working prototype of a game mechanic in an augmented reality environment. The planning of the project probably did allow for sufficient time for testing and evaluation after implementation but I didn't think through the fact that I might need ethical approval for play-testing. I had thought that I could host a play-testing session and video the players (particularly to illustrate the nature of the physical movements necessary to play the game). Given that I was trying to look at UX guidelines this made more sense than just sending out a questionnaire to potential users none of whom might be able to test the game without the software on their phones. The ethical issues only occurred to me just prior to testing and so I restricted the playtesting to four family members in an informal setting and did not video the play. If I were doing this again I would clarify if ethical approval were needed from the outset and obtain this if needed.

The aims and objectives of the project evolved as work progressed. Initially it was difficult to see beyond simply making an AR game and the fact that the project needed to have a purpose or address a problem was something that took me a long time to grasp. I emerged with a much greater understanding of the research process. When I did begin to look critically at the industry guidelines this turned out to be an interesting and challenging process. The project objectives are framed as a test of certain guidelines which would lead to recommendations and possibly standardisation. The evidence that I have collected is not as clear-cut as this implies and is a more of a mixture of what does seem to work in terms of UX design (eg. minimal screen interaction) along with a discussion of what is a way forward in AR mobile development (the need to build in safety rather than providing safety instructions and to build games where it is intuitive that you have to move rather than telling people to move). This is a very limited contribution to the field (especially as based only on play-testing by four people) but I think that the observations of the project themselves are fair.

7. Conclusions

The results gained contribute to four issues surrounding UX design, namely; player movement, safety, UI, and interaction. The report makes four recommendations regarding these issues.

Existing guidelines say to remind players that they can move around, now due to the findings from testing it would seem that players who are familiar with AR games would need no reminder of how AR works and that they can move around. However, for the players who have had no experience at all with AR before, it would seem that a more direct instruction in place of a reminder would help them realise that they can move around.

Safety reminders for AR games make sense, it is important to keep the users safe as moving while looking at a device is inherently unsafe. However, in terms of guidelines for designing good AR experiences, it doesn't make sense to limit developers on what they can do. Instead of restricting the player from doing something intuitive, recommend to them the best environment to use the application and remind them of the risks therefore guaranteeing as much safety as possible.

One of the industry guidelines states that it is good for immersion to not clutter the screen with controls and information and to make use of the entire display on the device. This is recommended as a good practice and the results of this project confirm it.

The game was developed to minimize player interaction with the screen as it was believed that this would increase the immersion with the game. It is recommended in this report that if the AR environment is to be the sole focus of the game/application then reducing the amount of interaction with the devices screen is effective, and that other interaction schemes should be explored as ones that are familiar for the user could take away from the AR experience.

8. References

AccuVein (2018) *AccuVein® Vein Vein Visualization: The Future of Healthcare is Here*. Available at: <https://www.accuvein.com/blood-draw/> (Accessed: 20/11/2018).

Apple (2018) *Human Interface Guidelines*. Available at: <https://developer.apple.com/design/human-interface-guidelines/ios/system-capabilities/augmented-reality/> (Accessed: 20/12/2018).

Azuma, R.T. (1997) A survey of augmented reality, *Presence: Teleoperators & Virtual Environments*, 6(4), pp. 355-385.

Barnett, B., Helbing, K., Hancock, G., Heininger, R. and Perrin, B. (2000) *An evaluation of the training effectiveness of virtual environments*.

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A. and Jeffries, R. (2001) *Manifesto for agile software development*, .

Bowman, D., Kruijff, E., LaViola Jr, J.J. and Poupyrev, I.P. (2004) *3D User interfaces: theory and practice*, *CourseSmart eTextbook*. Addison-Wesley.

D. Rapp, F. Niebling and M. E. Latoschik (2018) *The Impact of Pokémon Go and Why It's Not about Augmented Reality - Results from a Qualitative Survey*. pp. 1.

Dirin, A. and Laine, T.H. (2018) User Experience in Mobile Augmented Reality: Emotions, Challenges, Opportunities and Best Practices, *Computers*, 7(2), pp. 33 (18 pp.). doi: 10.3390/computers7020033.

Flanagan, G. (2018) *The incredible story of the 'Virtual Boy' - Nintendo's VR headset from 1995 that failed spectacularly*. Available at: <https://www.businessinsider.com/nintendo-virtual-boy-reality-3d-video-games-super-mario-2018-3?r=US&IR=T> (Accessed: 18/12/2018).

Fowle, K. (2015) *A look back at the doomed virtual reality boom of the 90s*. Available at: <https://killscreen.com/articles/failure-launch/> (Accessed: 18/12/2018).

Google (2018) *Augmented Reality Design Guidelines*. Available at: <https://designguidelines.withgoogle.com/ar-design/> (Accessed: 20/12/2018).

Google (2018) *Supported Devices*. Available at: <https://developers.google.com/ar/discover/supported-devices> (Accessed: 19/12/2018).

Google Play (2018) *AMON*. Available at: <https://play.google.com/store/apps/details?id=com.lykkestudios.amon> (Accessed: 20/12/2018).

Google Play (2018) *Bejeweled Classic*. Available at: <https://play.google.com/store/apps/details?id=com.ea.gp.bej3> (Accessed: 21/12/2018).

Google Play (2018) *Brickscape*. Available at: <https://play.google.com/store/apps/details?id=com.fiveminlab.brickscape> (Accessed: 20/12/2018).

Google Play (2018) *Candy Crush Saga*. Available at: <https://play.google.com/store/apps/details?id=com.king.candycrushsaga> (Accessed: 21/12/2018).

Google Play (2018) *Knightfall AR*. Available at: <https://play.google.com/store/apps/details?id=com.aetn.games.android.history.knightfall.ar> (Accessed: 20/12/2018).

H. Kato and M. Billinghurst (1999) *Marker tracking and HMD calibration for a video-based augmented reality conferencing system*. pp. 85.

Hamari, J., Malik, A., Koski, J. and Johri, A. (2018) Uses and Gratifications of Pokémon Go: Why do People Play Mobile Location-Based Augmented Reality Games? *International Journal of Human-Computer Interaction*, , pp. 1-16.

Inbar, O. (2018) *The AR Cloud is Making it Rain!* Available at: <https://medium.com/@oriinbar/the-ar-cloud-is-making-it-rain-66c4c0f164f9> (Accessed: 21/12/2018).

Liarokapis, F. and Evans, A. (2018) Editorial for special issue on interactive virtual environments for serious games, *Virtual Reality*, 22(2), pp. 89-90.

M. Billinghurst and A. Duenser (2012) Augmented Reality in the Classroom, *Computer*, 45(7), pp. 56-63. doi: 10.1109/MC.2012.111.

Metaverse (2018) *Education*. Available at: <https://medium.com/metaverseapp/tagged/education> (Accessed: 20/11/2018).

Niantic (2018) *Ingress Prime*. Available at: <https://www.ingress.com/game/> (Accessed: 20/12/2018).

Osborne, J. (2018) *Microsoft's HoloLens will be used in US Army combat missions*. Available at: <https://www.techradar.com/uk/news/microsofts-hololens-will-be-used-in-us-army-combat-missions> (Accessed: 20/12/2018).

R. Shea, D. Fu, A. Sun, C. Cai, X. Ma, X. Fan, W. Gong and J. Liu (2017) Location-Based Augmented Reality With Pervasive Smartphone Sensors: Inside and Beyond Pokémon Go! *IEEE Access*, 5, pp. 9619-9631. doi: 10.1109/ACCESS.2017.2696953.

Robinson, M., Eckhoff, D.G., Reinig, K.D., Bagur, M.M. and Bach, J.M. (2006) Variability of landmark identification in total knee arthroplasty, *Clinical Orthopaedics and Related Research*, 442, pp. 57-62.

Roesner, F., Kohno, T. and Molnar, D. (2014) Security and privacy for augmented reality systems, *Communications of the ACM*, 57(4), pp. 88-96.

Royce, W.W. (1987) *Managing the development of large software systems: concepts and techniques*. IEEE Computer Society Press, pp. 328.

Sensor Tower (2017) *Pokemon Go Has Grossed \$ 1 Billion Worldwide Since Launch*. Available at: <https://sensortower.com/blog/pokemon-go-one-billion-revenue> (Accessed: 20/12/2018).

Stone, R. (2009) Serious games: virtual reality's second coming? *Virtual reality*, 13(1), pp. 1-2.

Stone, R.J. (2008) Human factors guidelines for interactive 3D and games-based training systems design, *Human Factors Integration Defence Technology Centre Publication*, .

The Pokemon Company (2018) *Pokemon Go*. Available at: <https://www.pokemon.com/us/app/pokemon-go/> (Accessed: 20/12/2018).

Thomas, D. (2014) *Agile is Dead (Long Live Agility)*. Available at: <https://pragdave.me/blog/2014/03/04/time-to-kill-agile.html> (Accessed: 19/12/2018).

Unity (2018) *Unity, GDPR and Data Privacy - FAQ*. Available at: <https://unity3d.com/legal/gdpr> (Accessed: 21/12/2018).

V. A. Sangalli, T. V. de Oliveira, L. P. Soares and M. S. Pinho (2017) *SculptAR: An augmented reality interaction system*. pp. 260.

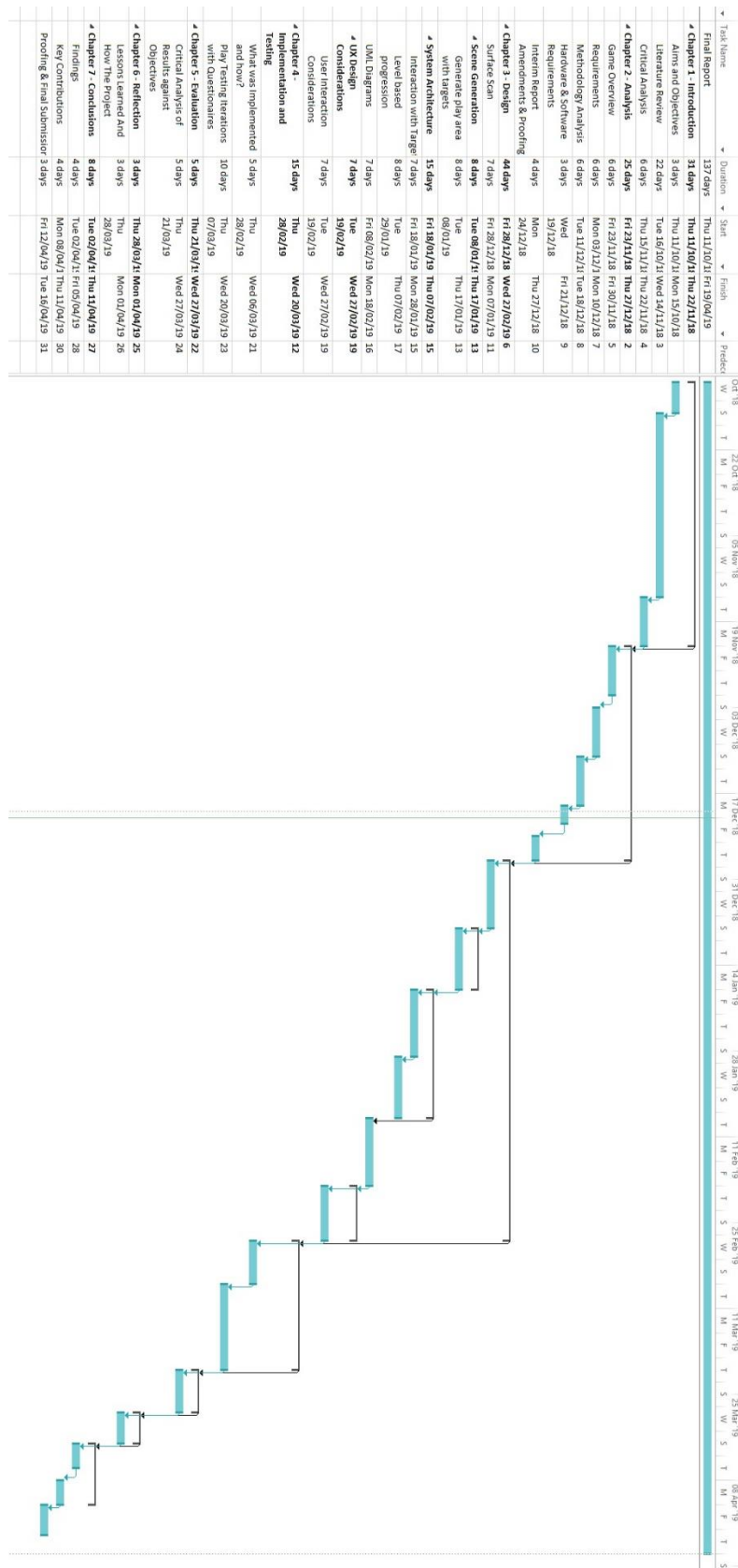
Vergun, D. (2017) *Heads-up display to give Soldiers improved situational awareness*. Available at: https://www.army.mil/article/188088/heads_up_display_to_give_soldiers_improved_situational_awareness (Accessed: 21/11/2018).

Vizrt (2018) *Viz Virtual Studio*. Available at: http://www.vizrt.com/products/viz_virtual_studio/ (Accessed: 20/11/2018).

W. Cai, R. Shea, C. Huang, K. Chen, J. Liu, V. C. M. Leung and C. Hsu (2016) The Future of Cloud Gaming [Point of View], *Proceedings of the IEEE*, 104(4), pp. 687-691. doi: 10.1109/JPROC.2016.2539418.

9. Appendices

9.1 Appendix A – Full Project Plan



9.2 Appendix C - Questionnaire Responses

| |
|---|
| How easy or difficult was it at the beginning to understand that you had to scan for a surface? |
| <i>It was easy to understand the scan I just didn't realise I had to touch the grid.</i> |
| <i>To look for the grid that was straightforward.</i> |
| <i>It was relatively difficult I didn't quite understand what that meant. A scan for a surface. You meant a surface on which to position the array. A sizeable free area.</i> |
| <i>I think it was fine.</i> |
| How quickly did you understand what you were supposed to do in the game? |
| <i>Very quickly</i> |
| <i>I understood quickly what to do but I found it more difficult to practically do it.</i> |
| <i>Not very quickly but that was down to me more than anything.</i> |
| <i>I understood what I had to do quickly. Getting it lined up was harder. It's just quite tricky. Its just you need to be precise but it is quite engrossing because of that. Challenging in a good way.</i> |
| Once you knew what to do did you feel in control of the game? |
| <i>Yeah</i> |
| <i>No I didn't feel in control because I thought the hoops had to be the same size whereas in fact they match on colour. Because some were big and some were very small I was originally looking for two blue ones of the same size. I thought it was sizing AND colour.</i> |
| <i>Yeah definitely.</i> |
| Would you have liked more guidance? With which bit? |
| <i>I didn't know you had to touch the grid because it just kept saying scan and then I suppose I didn't know I could go back to portrait mode. I didn't know you could pinch.</i> |
| <i>What it meant by matching</i> |
| <i>If there was an example that would have been quite good but maybe that would be hard to do. Or if something came up on the screen that said No you need to move your body so that you can see through both rings. When you said to think of it like threading a needle it became clear. After that it was just a question of how to move around.</i> |
| <i>No I think it was ok.</i> |
| What was good or fun about playing a phone game like this? |
| <i>It's fun for me because I never played that kind of AR game before. It's interesting to walk about the room and go for different heights and angles.</i> |
| <i>It was quite fun after the first one to try and rotate the screen to get them to be superimposed upon each other and I liked the noise it's a nice little tinkly magic sound. It was satisfying particularly because you had to do the fine movement and then you got the reward of the little sound.</i> |
| <i>Well it was challenging and there was a reward when you got the nice...when you got it done. Because it was tricky, when you started to hit targets that was enjoyable because you got a bit of a reward. I think what would make it better would be if it was a bit easier.</i> |
| <i>It was cool. I mean seeing them hanging there suspended was so cool. I've never done anything or played anything like that before and it was like being in bubble land. It was mesmerizing.</i> |
| What was bad or difficult about playing a phone game like this? |
| <i>If you don't have enough room then it's bad but there's enough room here.</i> |
| <i>It was difficult to learn to manipulate the hoops on the screen to get them to line up. Some of them were big and some of them were small and walking around the room trying to get them to orientate so that they</i> |

were superimposed so you could fire the dot at it. That was difficult to do all that tilting because I had never done it before. Just getting used to a 3d image. If you leave me alone for 15 minutes I'd be more adept.

For me I am not able to move my body physically very flexibly. Up and down, side to side. I needed to understand that I needed to be moving like this (demonstrates). And what I was doing was kind of this (demonstrates). Very high or very low targets were a problem for me. When they were horizontally spaced it was fine. Would it work if you lay on the floor? For some of them you would need a step ladder to get the angle right. Maybe you should narrow the area so that you don't have to go down really really low.

I think one of them didn't ping when I had it lined up. It worked every other time.

Was it immersive?

Yeah. I had to move about the room I wasn't sat still using my fingers. I had to use my body to get the technique.

Well I was focused on it yes.

Well yeah you get hooked on it. If it was a little easier to score points I could probably get immersed in it. It works definitely.

Yeah I was right in there with the bubbles.

9.3 Appendix D - Code Listings

References

Lean Touch - <https://assetstore.unity.com/packages/tools/input-management/lean-touch-30111>

Codelabs intro to ARCore -

<https://codelabs.developers.google.com/codelabs/arcore-intro/#0>

ARController Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using GoogleARCore;
using UnityEngine.UI;

public class ARController : MonoBehaviour
{
    public GameObject GridPrefab;
    public GameObject PlaySpacePrefab;
    public GameObject ARCamera;
    public GameObject CanvasUI;
    private bool isObjectPlaced;

    //We will fill the list with the planes that ARCore detects in the current frame
    private List<DetectedPlane> m_NewTrackedPlanes = new List<DetectedPlane>();

    void Start() {isObjectPlaced = false;}

    void Update()
    {
        //Check ARCore session status
        if(Session.Status != SessionStatus.Tracking)
        { return; }
    }
}
```

```

        //Only perform this code if there is no object placed
        if (!isObjectPlaced)
        {
            //The following function will fill m_NewTrackedPlanes with the planes
            that ARCore detects in the current frame
            Session.GetTrackables<DetectedPlane>(m_NewTrackedPlanes,
            TrackableQueryFilter.New);

            //Instantiate a grid for each Tracked plane
            for (int i = 0; i < m_NewTrackedPlanes.Count; i++)
            {
                GameObject grid = Instantiate(GridPrefab, Vector3.zero,
                Quaternion.identity, transform);

                //This function will set the position of the grid and modify the
                vertices of the attached mesh
                grid.GetComponent<GridVisualizer>().Initialize(m_NewTrackedPlanes[i]);
            }

            //Check if the player touches the screen
            Touch touch;
            if (Input.touchCount < 1 || (touch = Input.GetTouch(0)).phase !=
            TouchPhase.Began)
            { return; }

            //Check if the player touched any of the tracked planes
            TrackableHit hit;
            if (Frame.Raycast(touch.position.x, touch.position.y,
            TrackableHitFlags.PlaneWithinPolygon, out hit))
            {
                //Change the isObjectPlaced to true so we no longer check if the
                player touches the tracked planes
                //Clear the m_NewTrackedPlanes list to remove it from the game scene
                //Enable the UI Canvas
                isObjectPlaced = true;
                GameObject[] gridObjects;
                gridObjects = GameObject.FindGameObjectsWithTag("Grid");
                foreach (var f in gridObjects)
                { f.SetActive(false); }
                m_NewTrackedPlanes.Clear();
                CanvasUI.SetActive(false);

                //Create a new anchor
                Anchor anchor = hit.Trackable.CreateAnchor(hit.Pose);

                //Place the object on top of the tracked plane that was hit
                //Instantiate the Play space game object at the position of the hit
                //ARCore will keep understanding the world and update the anchors so
                we need to attach the object to the anchor
                Instantiate(PlaySpacePrefab, hit.Pose.position, hit.Pose.rotation,
                anchor.transform);
            }
        }
    }
}

```

Generator Script

```

using System;
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Generator : MonoBehaviour
{
    public GameObject[] targetArray;
    public GameObject[] groups;
    public GameObject Explosion;
    private new Camera camera;
    private AudioSource audio;

    private bool startRay = false;
    private string usedlocations = " ";
    private int targetCount;
    private int c;
    private float[] randomLocationX;
    private float[] randomLocationY;
    private float[] randomLocationZ;

    void Start()
    {
        camera = Camera.main; //Camera used for raycasts
        audio = GameObject.Find("Audio Source").GetComponent<AudioSource>();
        Vector3 center = gameObject.GetComponent<Renderer>().bounds.center; //Get
the position of the center of the play space plane

        //The unique locations for the targets to spawn at
        randomLocationX = new float[] { -0.5f, -0.4f, -0.3f, -0.2f, -0.1f, 0.0f,
0.1f, 0.2f, 0.3f, 0.4f, 0.5f };
        randomLocationY = new float[] { 0.5f, 0.6f, 0.7f, 0.8f, 0.9f, 1.0f, 1.1f,
1.2f, 1.3f, 1.4f};
        randomLocationZ = new float[] { -0.5f, -0.4f, -0.3f, -0.2f, -0.1f, 0.0f,
0.1f, 0.2f, 0.3f, 0.4f, 0.5f };
        targetCount = 20;
        c = 0;

        while (targetCount > 0)
        {
            targetCount--;
            int i = UnityEngine.Random.Range(0, groups.Length); //Select a random
target prefab to be used

            //Select a random coordinate from the list of unique numbers
            int x = UnityEngine.Random.Range(0, randomLocationX.Length);
            int y = UnityEngine.Random.Range(0, randomLocationY.Length);
            int z = UnityEngine.Random.Range(0, randomLocationZ.Length);

            //While the location list already contains the random location
            //get a new location
            while (usedlocations.Contains(" " + x + y + z))
            {
                x = UnityEngine.Random.Range(0, randomLocationX.Length);
                y = UnityEngine.Random.Range(0, randomLocationY.Length);
                z = UnityEngine.Random.Range(0, randomLocationZ.Length);
            }
            usedlocations = usedlocations + " " + x + y + z;
        }
    }
}

```

```

        Vector3 pos = new Vector3(randomLocationX[x], randomLocationY[y],
randomLocationZ[z]);
        pos = pos + center; //Set the position of the target relative to the
plane
                                //Targets get instantiated as a child of the play
space plane
                                //but the random position alone would be relative to
the world origin
                                //not the planes origin

        GameObject go = Instantiate(groups[i], pos, Quaternion.identity,
transform) as GameObject;
        go.name = "Target" + c;
        c++;
    }
    startRay = true; //The raycast can now be generated in the Update method
}

/// <summary>
/// Every update there is a ray being cast from the center of the screen
/// If the ray hits an object with a color tag then it passes the ray
/// into the MatchTest function
/// </summary>
void Update()
{
    if (startRay)
    {
        Ray pRay = camera.ViewportPointToRay(new Vector3(0.5f, 0.5f, 0));
        RaycastHit p;
        if (Physics.Raycast(pRay, out p))
        {
            switch (p.collider.transform.tag)
            {
                case "Red":
                    MatchTest(pRay);
                    break;
                case "Blue":
                    MatchTest(pRay);
                    break;
                case "Green":
                    MatchTest(pRay);
                    break;
                default:
                    return;
            }
        }
    }
}

/// <summary>
/// The function uses the ray and performs a Physics.RaycastAll on it
/// to get all of the objects that the ray hits in the scene.
/// The order of the hits is not guaranteed so just the first two hits
/// are checked
/// </summary>
/// <param name="_ray"></param> The ray from the update method. It has hit a
target
void MatchTest(Ray _ray)
{
    //Create an array of raycasthits to store all the collisions from the
RaycastAll
    //RaycastAll returns all hits from the scene

```

```

RaycastHit[] hits;
hits = Physics.RaycastAll(_ray, 3.0f);
string tag = hits[0].collider.transform.tag;

//If the first hit is a target then check the next to make sure both hits
are the same type
for (int i = 0; i < 2; i++)
{
    if (hits[i].collider.transform.tag != tag)
    { return; }
}
MatchSuccess(hits);
}

/// <summary>
/// Called whenever the player has successfully lined up
/// two targets in a row. We get the vector3 for the centre of the plane
/// again because we're using the same code to get a random position.
/// </summary>
/// <param name="hits"></param> The array of hits, need to access the positions
void MatchSuccess(RaycastHit[] hits)
{
    Vector3 center = gameObject.GetComponent<Renderer>().bounds.center; //Get
the center of the playspace to make the target positions

//relative to the plane
for (int i = 0; i < 2; i++)
{
    Vector3 t = hits[i].collider.transform.position; //Get the current
position of the target
    audio.Play(); //Play the sound effect
    GameObject e = Instantiate(Explosion, t, Quaternion.identity);
//Instantiate the explosion particle system at the target position
    t = t - center; //Change the vector3 t to be relative to the world
origin again
                                //The string usedLocations are all relative to the world
origin 0,0,0

    string oldLocation = " " + t.x + t.y + t.z; //Get the string that was
used when creating the target in the Start function
    Destroy(hits[i].collider.gameObject); //Destroy the target

    //Create a new target at a new position
    int j = UnityEngine.Random.Range(0, groups.Length);
    int x = UnityEngine.Random.Range(0, randomLocationX.Length);
    int y = UnityEngine.Random.Range(0, randomLocationY.Length);
    int z = UnityEngine.Random.Range(0, randomLocationZ.Length);

    while (usedLocations.Contains(" " + x + y + z))
    {
        x = UnityEngine.Random.Range(0, randomLocationX.Length);
        y = UnityEngine.Random.Range(0, randomLocationY.Length);
        z = UnityEngine.Random.Range(0, randomLocationZ.Length);
    }
    string newLocation = " " + x + y + z;
    usedLocations = usedLocations.Replace(oldLocation, newLocation);
//Replace the old location of the destroyed target
    Vector3 pos = new Vector3(randomLocationX[x], randomLocationY[y],
randomLocationZ[z]);
    pos = pos + center;
    GameObject go = Instantiate(groups[j], pos, Quaternion.identity,
transform) as GameObject;

```

```
        go.name = "Target" + c;
        c++;
    }
}
```

SceneManager Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.SceneManagement;

public class SceneManager : MonoBehaviour
{
    //The screen won't timeout while playing
    void Start()
    { Screen.sleepTimeout = SleepTimeout.NeverSleep; }

    //Change the scene
    public void ChangeScene(string scene)
    { UnityEngine.SceneManagement.SceneManager.LoadScene(scene,
LoadSceneMode.Single); }
}
```

ParticleSystemAutoDestroy Script

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class ParticleSystemAutoDestroy : MonoBehaviour
{
    private ParticleSystem ps;

    //Attached to a particle system, this will destroy the gameobject after
    //the effect has finished
    public void Start()
    { ps = GetComponent<ParticleSystem>(); }

    public void Update()
    {
        if (ps)
        {
            if (!ps.IsAlive())
            { Destroy(gameObject); }
        }
    }
}
```

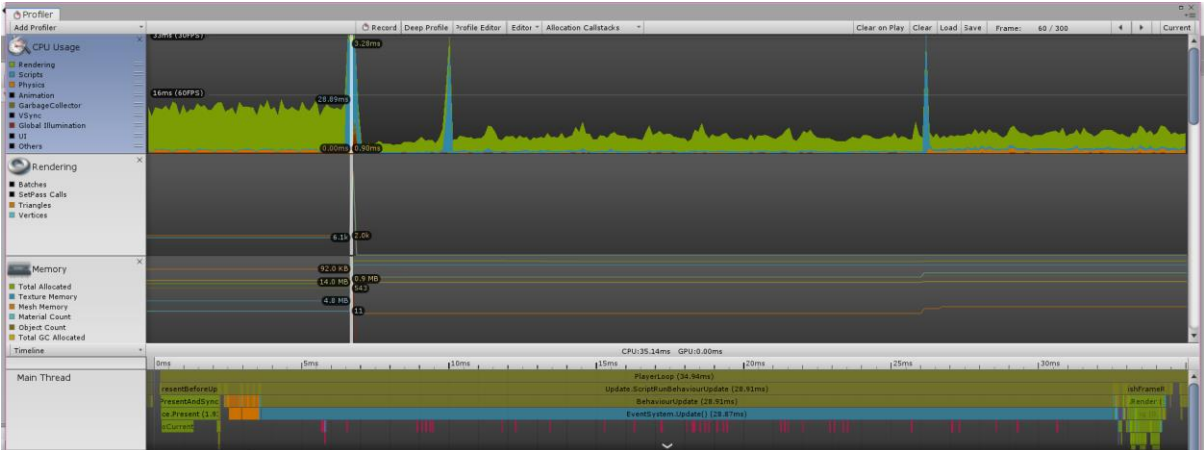

9.4 Appendix E - Implemented Functional Requirements

| Functional Requirement | Implemented? |
|---|--------------|
| <u>Launch</u> - The game should launch without issue. | Yes |
| <u>Load into main menu</u> - The game should load into a home screen where the player can choose game modes and change settings. It should be clear and concise. | Yes |
| <u>Start game</u> - When the player selects start game the scene will change to the surface scan mode where the player will find scan their environment. After scanning, the game will generate the virtual gameplay scene and overlay it on top of the real-world feed from the camera. | Yes |
| <u>Allow the player to aim using the camera</u> - The player should be able to clearly see where they're aiming and the trajectory of their aim. | Yes |
| <u>Allow the player to shoot</u> - The player should be able to perform some action to match the targets they're aiming at. | Yes |
| <u>Making a match</u> - The player should receive points for making a match of three or more targets. | Yes |
| <u>Targets</u> - Targets will be spawned in random places with equal numbers of each type. And when they are matched, will destroy and respawn at a different position in the scene. | Yes |
| <u>Win Condition</u> - If the player reaches the required score for the level before the timer runs out then they complete the level. | No |
| <u>Lose Condition</u> - If the player runs out of time then they lose and the game is over. | No |
| <u>Timer</u> - The player should be able to see how much time they have left. | No |
| <u>Objectives</u> - The objectives should be made clear to the player, through something similar to a tutorial. | No |
| <u>Directions</u> - Directions for how to use the game should be conveyed effectively. | Yes |

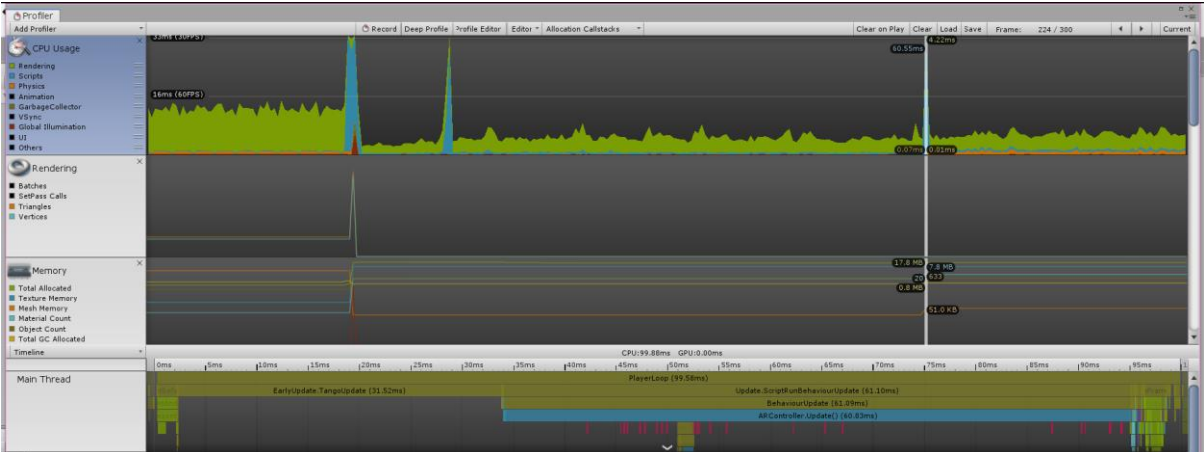
| | |
|---|----|
| Music - Music should be appropriate for the context of the scene i.e. Main menu and gameplay. | No |
|---|----|

9.5 Appendix F - Unity Profiler Screenshots

ScanStart



ScanDetectedPlanes



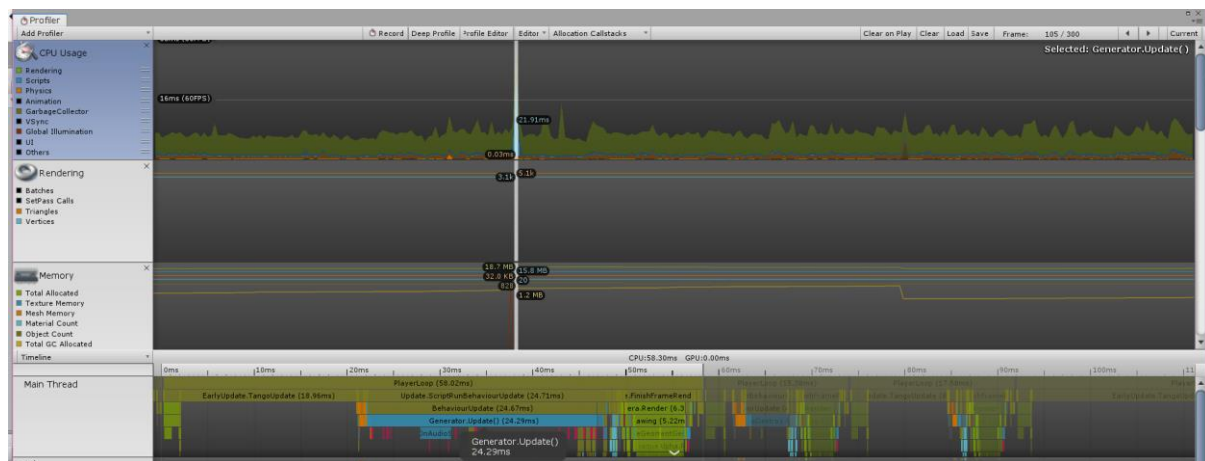
ScanUpdate



SpawnGeneratorStart



MatchGeneratorUpdate



MatchParticleAutoDestroy

