Bachelor Project

Czech
Technical
University
in Prague

F3

Faculty of Electrical Engineering
Department of Control Engineering

Multiplayer mobile game development using augmented reality

Ondřej Trojan

Supervisor: Ing. David Sedláček, Ph.D.
Field of study: Cybernetics and Robotics
Subfield: System control
May 2018

Acknowledgements

I would like to thank my family and my girlfriend Eva for their intensive support during my studies. Furthermore, I appreciate my supervisor Ing. David Sedlacek PhD. for willingness, help and given consultations. Lastly, I thank the University of Cagliari especially prof. Giuliano Armano and prof. Giorgio Giacinto for their offer of advice and kind services.

Declaration

I declare that the presented work was developed independently and that I have listed all sources of information used within it in accordance with the methodical instructions for observing the ethical principles in the preparation of university theses.

In Prague, 28. May 2018

Abstract

This work describes the development of a multiplayer mobile game build on top of the performed analysis and research of the new augmented reality usability concept with the use of selected state-of-art augmented reality library which is to be output from their comparative analysis. The best evaluated from functional, technological and quality of tracking aspect is Vuforia which with its real-time target tracking offers new utilisation approach. That said, our demonstration game design associates Augmented reality, Unity editor, Android platform and Multiplayer guidelines whereas the implementation leverages high-level network approach from Unity together with Vuforia planary image tracking. On the top of the realised application, the five people's testing was executed from which factual conclusions and observations were obtained.

The game is targeted to Android system phones with SDK larger than 24. The development was undertaken in C# using Unity Editor 2017.4 and Vuforia 7.1

Keywords: Augmented reality, Vuforia, Unity, C#, Mobile application, Multiplayer game, Game development, Android

Supervisor: Ing. David Sedláček, Ph.D.
Department of Computer Graphic and Interaction,
Karlovo náměstí 13,
Praha 2

Abstrakt

Tato práce popisuje vývoj multiplayerové mobilní hry postavené na výstupu provedené analýzy a výzkumu nové koncepce použitelnosti rozšířené reality, která využívá knihovnu vybranou ze srovnání nejvyspělejších vývojových nástrojů rozšířené reality. Nejlépe hodnocená z pohledu funkčního, technologického a kvality sledování je Vuforia, která svým obrazovým sledováním v reálném čase otevírá nové možnosti použitelnosti. To znamená, že náš demonstrační herní projekt sdružuje rozšířenou realitu, Unity editor, platformu Android a principy hry pro více hráčů zatímco při implementaci používáme vysokoúrovňový síťový přístup od společnosti Unity společně se sledováním rovinných targetů od Vuforie. Na konec bylo na realizované aplikaci provedeno testování pěti osob, z nichž byly získány faktické závěry a pozorování.

Hra byla zaměřena na systémové telefony Android s SDK větší než 24. Vývoj byl proveden v C# pomocí Unity Editoru 2017.4 a Vuforie 7.1

Klíčová slova: Rozšířená realita, Vuforia, Unity,C#, Mobilní aplikace, Hra více hráčů, Herní vývoj, Android

Překlad názvu: Víceuživatelská hra v rozšířené realitě na mobilní platformě

Contents

1 Introduction **1**
1.1 Motivation 1
1.2 Aim 1
1.3 Thesis structure 1

2 Augmented reality **3**
2.1 Definition 3
2.1.1 Mediated reality 4
2.1.2 Virtual reality 4
2.1.3 Mixed reality 4
2.1.4 Augmented reality 5
2.2 Function principle 6
2.2.1 Marker tracking 6
2.2.2 Markerless tracking 7
2.2.3 Further division 7
2.3 Problems 7
2.4 Usability 7
2.5 Applications 8

3 Analysis **11**
3.1 SDKs 11
3.1.1 ARToolKit 11
3.1.2 Vuforia 12
3.1.3 ARKit 13
3.1.4 ARCore 14
3.1.5 Wikitude 14
3.1.6 EasyAR 15
3.1.7 Maxst 16
3.2 Platforms or hybrid libraries . . . 17
3.3 Comparison of libraries 17
3.3.1 Functions 17
3.3.2 Usability 18
3.3.3 License 19
3.4 Quality of tracking 20
3.5 Conclusion 20

4 Design **21**
4.1 Objectives 21
4.1.1 Existing game concepts 21
4.1.2 Proposed game concept 21
4.2 Requirements 23
4.2.1 Functional requirements 23
4.2.2 Performance requirements . . . 23
4.2.3 Design requirements 23
4.2.4 Operational requirements . . . 24
4.2.5 Constrains 24
4.3 Background 25
4.3.1 Game predecessor 25
4.4 User interface 26
4.4.1 Trackable movement 27
4.4.2 Menu 27
4.4.3 HUD 27
4.5 Target tracking 27
4.5.1 Marker image selection 27
4.5.2 Target detection 28
4.5.3 Target synchronization 29

5 Implementation **31**
5.1 Development environments 31
5.1.1 Unity 31
5.1.2 Visual studio 2017 32
5.2 Development tools 32
5.2.1 Vuforia target manager 32
5.2.2 .NET tools 32
5.2.3 Unity tools 34
5.2.4 Code version control system . 35
5.3 Architecture 35
5.3.1 Logic distribution 35
5.3.2 Network management 36
5.3.3 World generation 37
5.3.4 Real-time navigation mesh . . 37
5.4 Game 37
5.4.1 Game loop 38
5.4.2 Gameplay 40
5.5 Third party libraries 41
5.6 Problems 41

6 Evaluation **43**
6.1 Requirement verification 43
6.1.1 Requirements tests 45
6.2 Usability test 47
6.2.1 Test plan 48
6.2.2 Test results 49
6.2.3 Observations 51
6.3 Future work 52
6.4 Known bugs 53
6.5 Known issues 53

7 Conclusion **55**

References **57**

A List of Shortcuts **61**

B Content of attached CD **63**

C Target images **65**

D Game screenshots **69**

E Project Specification **73**

Figures

2.1 Realities and their relationships . 3
2.2 Position of mixed reality in our perception [6] 5
2.3 Position of AR and AV at virtual continuum 5
2.4 Pokemon Go 9
2.5 World Lens demonstation 9
2.6 IKEA Place 10

3.1 AR toolkit logo 11
3.2 Vuforia logo 12
3.3 ARkit logo 13
3.4 ARCore logo 14
3.5 Wikitude logo 14
3.6 EasyAR logo 15
3.7 Maxst logo 16

4.1 Game control 22
4.2 Game layout 22
4.3 AR Tank v1.1 26
4.4 User interfaces 26
4.5 Target location raycasting 29
4.6 Target found/lost flowchart 30

5.1 Unity Editor 31
5.2 Vuforia target manager 32
5.3 Remote actions graph [31] 36
5.4 NavMesh Surface component . . . 38
5.5 Lobby 39
5.6 Gameplay 40
5.7 Vuforia error 42

D.1 Menu I 69
D.2 Menu II 70
D.3 Deathmatch I 70
D.4 Deathmatch II 71
D.5 Cooperative I 71
D.6 Cooperative II 72

Tables

3.1 Function comparison 18
3.2 Target platform comparsion 18
3.3 Development platform comparison 19
3.4 License comparsion 19

4.1 Functional requirements 23
4.2 Performance requirements 23
4.3 Design requirements 24
4.4 Operational requirements 24
4.5 Constrains 24
4.6 Suggested image attributes 27

6.1 Verification matrix 45
6.2 Test 1 46
6.3 Test 2 47
6.4 Test 3 47
6.5 Participants initial survey 48
6.6 Participants devices 48
6.7 After testing survey 51

Chapter 1

Introduction

1.1 Motivation

In recent years, the augmented reality (AR) technology has a made significant progress toward maturity with many commercial products available and its applications are a contemporary very fashionable topic. While in manufacturing and marketing business its appliance comes relatively slow the vast majority of today's utilisation which ordinary user come in touch with are in the entertainment industry. According to an article [1] , revenue generated by the augmented/virtual reality market is expected to reach $108 billion by 2021, with the share of AR likely to reach $83 billion (versus $25 billion for VR). Augmented reality is therefore a huge and rapidly growing market. [2] and is certainly capable of much more than catching Pokémon. That said, the new concepts of utilisation shall be introduced.

1.2 Aim

The primary focus of the thesis is to develop a multiplayer mobile game using augmented reality library which is to be determined as a major secondary intention. The third aim of this thesis is to introduce a new game concept build on augmented reality principles.

1.3 Thesis structure

The elementary division of the thesis is to analysis and development. Besides that first introduction to problematics of AR is presented then analysis of suitable SDKs is performed followed by the design of the application, its implementation and finally evaluation with completed tests and finally the observations.

Chapter 2

Augmented reality

This chapter explains terminology, problematic of usage and usability of the augmented reality. Lastly it shows some use-cases and presents successful applications.

2.1 Definition

For the correct understanding of this principle, the reader ought to recognise the differences between "other" types of realities. In recent years the shortcuts such as AR, VR, MR were often misunderstood and interchanged furthermore due to their similarities and general interconnections, the accessible public explanations are regularly vague, loose and without further context. Despite the ongoing broad debate about terms defining the [Fig. 2.1] gives a decent example of their relationships. I do my best to describe the problematics as clearly as possible.

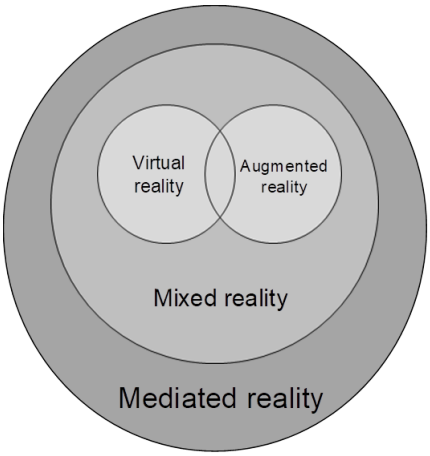


Figure 2.1: Realities and their relationships

2.1.1 Mediated reality

Computer-mediated reality is the kind of reality where the observer is watching the surrounding world through an electronic device which is functioning as an intermediary between observers eye and observable item. This term is not widely used, but for our purpose its adoption is essential. Such device can be considered a simple camera with a display or a mobile phone (when user watch screen is displaying cameras view of the real world). Let's imagine a situation where the observer cannot see the real world over some blockade (e.g. soldier in the tank is watching mediated outside world on the screen or cars driver is watching parking camera on a car screen). In more simple meaning, we can say that we first capture the real world with a camera then convert it to an electrical signal and accordingly visualise it.[3], [4]

2.1.2 Virtual reality

Virtual reality (VR) is an environment in which the participant-observer is entirely immersed and possibly can interact with a fully virtualised word. VR label is also frequently used in association with a variety of other environments, to which total immersion and complete synthesis do not necessarily pertain, but which fall somewhere along a "virtuality continuum"[1] which connects real environments to entirely virtual ones [5]. Furthermore, VR is often incorrectly interchanged even with Augmented reality and mixed reality which is to be described later in this work. Users perspective is through mediated observation entirely separated from surrounding real world and replaced with illusion. Nowadays it is characteristic to see the relation between VR technology and usage of specialised equipment such as VR glasses. Various kinds of VR glasses differ from each other, from the less complex to the very sophisticated (e.g. Oculus Rift, Sony PlayStation VR). They allow a user to perceive virtual reality from first person view and in 3D mode by cause of perspective given is transferred separately for left and right eye. Moreover, the transmitted picture can be changed accordingly to tilt of the glasses which results in user perceiving the virtual world in the way he sees the real one.

2.1.3 Mixed reality

Mixed reality (MR) sometimes called hybrid reality associates real and virtual world in the way that they can co-exist and possibly interact with each other. [Fig. 2.2] gives a nice idea of its relations with observer and computer.

[1] The virtuality continuum is a continuous scale ranging between the completely virtual, a virtuality, and the completely real, reality. See [Fig. 2.3]

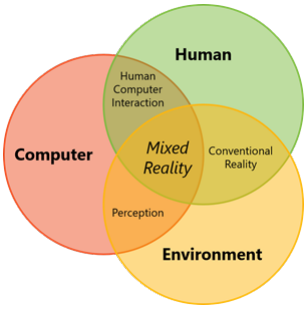


Figure 2.2: Position of mixed reality in our perception [6]

The Mixed reality term was first introduced in the year 1994 [5]. However, there are a few later different interpretations in slight contrary to each other. The summary of contemporary explanations are shown in the list below and they are defined as:

1. An outcome of blending the physical world with the digital one.[6]
2. An Abstract term relating AV and AR [5]

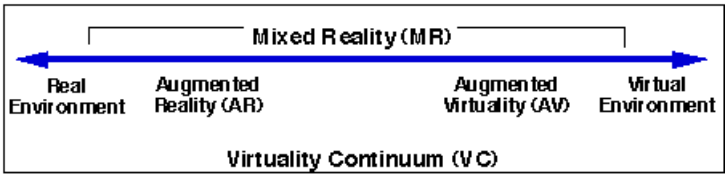


Figure 2.3: Position of AR and AV at virtual continuum

   classifying the perceived environment according to severity its virtualisation.

3. An abstract term relating AR and VR into one group
4. AR and MR are sometimes used as synonyms. Esspecialy in unprofessional and amateur cases.

2.1.4 Augmented reality

Augmented Reality (AR), refers to all cases in which otherwise real environment is mediated and augmented with non-real(virtualised) objects. Typical augmented functionality consists of displaying various pictures, texts or even 3D models over real-world perceived by the user. Latest applications use additional data inputs such as sounds, location and mainly video. The concept of augmented reality directly follows the principle of mediated reality and extends it thoroughly.

That can be further divided into:

- **Marker-Based** augmented reality mobile applications are based on image recognition. They use portable devices camera to detect certain patterns or markers [2]

- **Location based** AR apps don't need markers; instead, they use GPS and other position detectors (accelerometers and digital compasses) to establish your location and create augmented reality objects. [2]

Various usage of AR is to be described in section [2.4]

Augmented virtuality

Augmented virtuality (AV) is the real-time representation of the current state of real-world elements in mediated reality. It is also a converse case of AR on the virtuality continuum.

2.2 Function principle

Function principle varies from technology to technology and changes rapidly in time. To properly deliver AR technology the necessary components are:

1. Input sensors - serving as an intermediary between the real world and the computing devices capturing various data. That could be either Camera, GPS or even accelerometer or compass.
2. Computing device - handling calculations and merging varied data inputs
3. Display device - output device presenting augmented virtual objects over mediated real world

With some simplification, we can say that today's AR is working in two ways. Either with previously loaded and known target or targetless.

2.2.1 Marker tracking

On our computing device must be running an application which is searching for already loaded and a known target. That is achieved in three steps. Firstly, the significant points, static reference points or optical flow of the target are determined. Secondly, outputs are compared to the captured real-world footage separated to each frame. If the sufficient conformity is attained the co-ordinate real world system is determined as the last stage. **8** These processes are further studied in machine learning and computer vision fields, and its in-depth description is not an objective of this work. In a more simplified way, we can say that the application is in each frame trying to determine a position, tilt, and overall targets orientation.

In the background of the detection there are two main approaches:

- **Marker tracking** - they are special images designed prior to the tracking start which often holds an information and their recognition algorithms and processes are distinct. Typical types are QRCode or VUMark [7]
- **Natural features** - in short NF is a simple descriptor which is run over the captured image and then the significant features are filtered and the search for clusters is executed.

2.2.2 Markerless tracking

"Markerless AR" is a term used to denote an Augmented Reality application that does not need any pre-knowledge of a user's environment to overlay 3D content into a scene and hold it to a fixed point in space. [8] . The computing device is also running application, but before the virtual objects displaying; the application must perform a scan of the environment. That can be either performed real-time or had been done prior to application launch. After the application is run, any camera movement results in environment scanning and sufficiently scanned environment sets a reference point in the virtual scene which allows the virtual object placement.

2.2.3 Further division

With co-ordinate system information the application is then capable of filling the virtual environment with various object depending on the target location (e.g. display 3D object above target). The AR working principle can be further divided into :

- **Optical see-through** - systems combine computer-generated imagery with "through the glasses" image of the real world, usually through a slanted semi-transparent mirror. [9]
- **Video see-through** - presents mediated real world video feeds through a device display. [9] Typically mobile phone.

2.3 Problems

There are few obstacles for marker discovery. We are often not able to find the marker because of an insufficient camera resolution which does not allow the fine detail's determination of the marker on more substantial distances. Furthermore, overall readability is strongly influenced by surrounding light conditions and finally right choice of marker. All these things together create a basic model guidelines ensuring proper image recognition conditions we shall try to reproduce. Suitable and inappropriate markers will be intensely discussed later, but we can generally say that varied, heterogeneous, irregular shapes with sharp edges and deep contrast are a good choice. Fading, dull, and fuzzy are on the other hand not suitable.

Markerless tracking challenge a similar problem with uneven terrain or a diverse object in the real world that might result in insufficient reference point attachment. Light conditions, camera resolution together thus creates key preconditions to successful tracking.

2.4 Usability

Intensive technological development drove an expansion of mobile electronics devices in last two decades and therefore nowadays nearly every human in the

first worlds is equipped with a smartphone, and this state creates a vast group of people which can utilise AR application in everyday life. However, the AR was subject of interest before. In early times of AR technology (the 80s, 90s), few prototype systems had been developed, and these systems were aimed at a narrow group of, often top experts, in particular fields. Despite the fact that today AR technology is beneficial for a considerably small amount of people we can say that it is on the significant rise.

The boom of AR comes with few particular applications which recently have caused a wave of discussion and raise awareness of this issue.

2.5 Applications

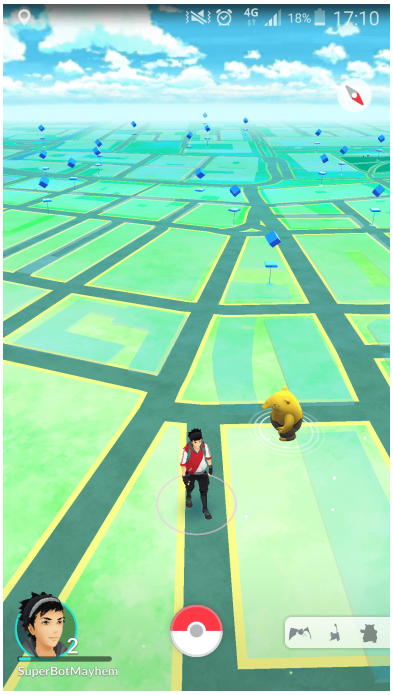
This section presents several present-day sleek applications using augmented reality which have recorded a great success and significantly raises awareness about AR.

Pokemon Go

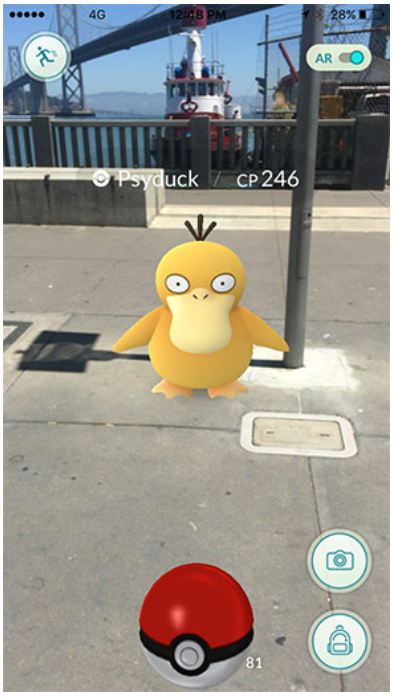
This mobile phone game (Android, iOS) is based on the AR principle, was launched in July 2016 and caused a huge popularity wave with up to 28 million users in one day. The popularity of the game as it has often happens receded over the 5 million users per day, which is still a very high number. As an AR input into the gaming environment, it uses GPS location and phones camera. The principle of the game is to capture a Pokemon and then use it to fight other players. By moving the real world [Fig. 2.4a], the user may encounter and interact with virtual objects (Pokémon) and interact with them [Fig.2.4].[10]

Word Lens (Google)

This is an AR application for translating scanned text into another language. It uses a mobile video capture camera that scans and searches for text. Once found, content is translated and transformed into similar fonts and styles as the original and projected back on the same background. The user perceives the translated text as well as the original only in another language. The technology was introduced in 2010, and after the first stable release in 2014, the company was bought by Google with the intention of integrating Word Lens into its Google Translator. [11]



(a) : Real world map



(b) : Real world augmented with a creature

Figure 2.4: Pokemon Go

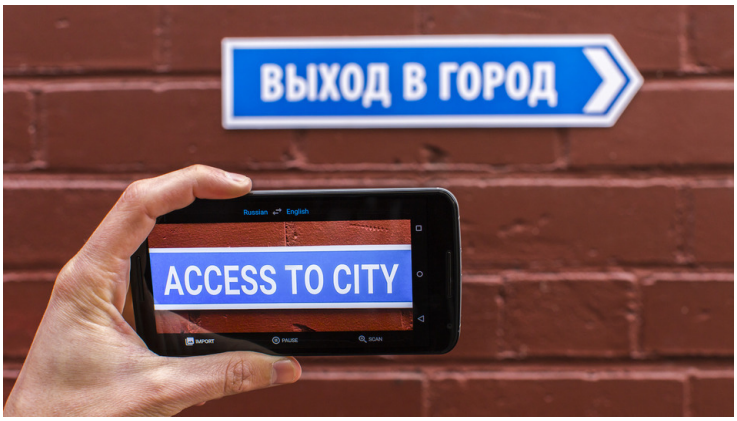


Figure 2.5: World Lens demonstation

Lenses (Snapchat)

This technology was first introduced in September 2015 within the Snapchat application used for photo exchange where sent photos are deleted after the previously set time. Lenses also allow the real-time addition of filters to the image by the camera which recently became very popular among young people. The application first detects the face and then adds the virtual object or edits your real face in some form of art (typical motives are dog nose and

ears). The technology has been enriched over time by various animations that are even more realistic. Similar technology has been used by Snapchat competitors such as Instagram and Facebook and for example, users at the beginning of the year 2017 sent 1.5x as much Instagram stories (250mil) than messages over Snapchat(166mil). Not all of the messages have of course contained augmented reality. [12], [13]

IKEA Place

IKEA Place lets you virtually 'place' IKEA products in your space. The app includes 3D and true-to-scale models of everything from sofas and armchairs to footstools and coffee tables and therefore allows users clearer picture of how the furniture would look in the real room. The first version of the application was used with a marker placed in the room, but temporary version 2 uses scanned floor space. [14]

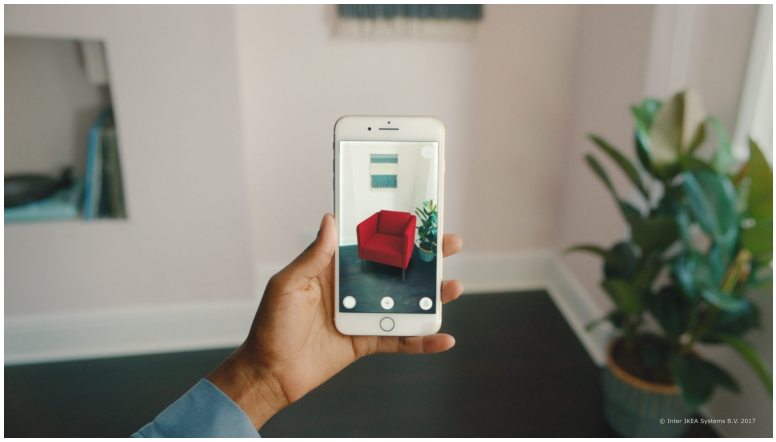


Figure 2.6: IKEA Place

Chapter 3

Analysis

This Chapter will introduce several state-of-the-art AR SDK; then its comparison will be performed followed by the conclusion which will result in the selected SDK to be used in upcoming development. Main analysis criteria are functions, usability and licenses whereas detailed computer vision algorithms and procedures are not subject to this work. Ultimately the chosen SDK shall allow us to develop a mobile game application in the Unity editor with moderately advanced tracking features and ability to track at least five images simultaneously.

3.1 SDKs

In the vast majority of applications developers using extended reality do not develop machine vision and marker recognition in the image itself. For these purposes, libraries that already offer prepared recognition technology are used. There is a whole range of them on the market, and their comparison will be the subject of the upcoming section.

3.1.1 ARToolKit

Figure 3.1: AR toolkit logo

This is an open source AR library shared on GitHub. Initially developed by an employee of Hirokazu Kato of the Nara Institute of Science and Technology in 1999, the last released version at the time of writing this thesis was from 2016. Offer decent functionality with lack of more advanced features.. [2], [15]

Latest version: Stable ARToolKit5, Beta ARToolKit 6

Functions:
- 2D marker tracking
- 1 or 2 camera tracking
- Javascript support

Platforms:
- Android
- iOS
- Linux
- Windows
- Mac OS
- Smart Glasses

License:
- Free, Open source, GNU Lesser GPL v3.0

Additional:
- Plugins for Unity and OpenSceneGraph

3.1.2 Vuforia

Figure 3.2: Vuforia logo

Vuforia is one of the most popular AR SDKs introduced in 2010 yet under the name of QCAR SDK. It is an SKD for the development of mobile applications with expanded reality. It offers excellent integration with XCode, Android Studio and Unity (from 2017.2). Due to its more commercial character, it does not offer free basics functionality but exceeds its competitors with functionality and tracking precision. [16]

Latest version: Vuforia 7.1

Functions:
- 2D marker tracking
- 3D object and model tracking
- Multi target tracking
- Text tracking
- VuMark - a combination of picture and QR-code
- Ground plane - automatic detection of horizontal surfaces from camera
- Vuforia Object Scanner - allows 3D marker creation by scanning
- Extended tracking - persist tracking despite lost of tracked marker
- Smart Terrain Technology - allows scan real world, reconstruct image/object

- Cloud databases - load targets from distanced server, allows great amount of markers

Platforms:
- Android
- iOS
- Universal Windows Platform
- Smart Glasses, Hololens

License:
- Free - for limited function and user amount
- One-time fee 499 $- standart functions
- Monthly 99 $- advanced functions

Additional:
- Very good documentation and large community

3.1.3 ARKit

Figure 3.3: ARkit logo

ARkit introduced by Apple at 2017 Apple Worldwide Developers Conference is an SDK targeted only for latest iOS devices (iPhone6s <) and exploited from 2 camera devices. [17], [18]

Latest version: ARKit 1.5

Functions:
- 2D marker tracking
- Markless tracking - with irregular shape detection
- SLAM - Simultaneous Localization and Mapping
- Light estimation image/object

Platforms:
- iOS (Swift, XCode)

License:
- Free
- App publish to App Store Developer account is neccesary (yearly 99 $)

3.1.4 ARCore

Figure 3.4: ARCore logo

ARCore introduced in February 2018 as an Android's world answer to ARKit offers an AR functionality for several latest mobile devices running Android 7.0 and higher. The new version of 1.2 introduces cloud anchor system allowing synchronisation of surrounding world among users. Therefore allowing all kids of multi-player behaviour. [19]

Latest version: ARCore 1.2 (8th May 2018)

Functions:
- Markless tracking
- SLAM - Simultaneous Localization and Mapping
- Large areas mapping
- Light estimation
- Cloud Anchors - colaborative AR experience image/object

Platforms:
- Android, Android NDK
- Unity
- Unreal

License:
- Free

3.1.5 Wikitude

Figure 3.5: Wikitude logo

Wikitude is a mobile platform SDK originating from WikitudeGlobH's Wikitude World Browser. As the first AR SDK, the JavaScript API has been enabled and supported. The current version 7.2 has brought many additional innovations. Due to its more commercial character, it does not offer free basics functionality, furthermore its prices starts at 2490 eur yearly. It however exceeds its competitors with very advanced functionality. [20]

Latest version: Wikitude SDK 7.2

Functions:
- 2D marker tracking

- 3D object tracking
- Multi target tracking
- Markless tracking
- Javascript support
- SLAM - Simultaneous Localization and Mapping
- SMART - integration with ARkit and ARcore
- Extended tracking - persist tracking despite lost of tracked marker
- Geo Data - facilitates integration with location information
- Cloud Recognition - allows you to save the tracking image to the cloud to increase the capacity and performance of the application

Platforms:
- Android
- iOS
- Unity
- Titanium
- Xamarin
- Smart glasses

License:
- Yearly from 2490 eur to 2990 eur - depending the functions
- Yearly 4490 eur - with Cloud Recognition

3.1.6 EasyAR

Figure 3.6: EasyAR logo

The latest version is divided into the Basic version and has only 2D image recognition and Pro version with advanced features. It is considered to be a small and light library with wide usage but with poor and insufficiently localised documentation as the core market and development is in China. [21]

Latest version: EasyAR SDK 2.2.0

Functions:
- 2D marker tracking
- SLAM - Simultaneous Localization and Mapping
- Multi target deteciton
- Cloud recognition
- Planar Image Tracking
- 3D object tracking(PRO)
- Screen recording (PRO)

Platforms:
- Android
- iOS
- Unity
- UWP
- Windows

License:
- Free - Basic
- One-time 499$- Pro version

3.1.7 Maxst

Figure 3.7: Maxst logo

Maxst offers two kinds of software to create augmented reality apps: Maxst AR SDK 2D and Maxst AR SDK 3D. As their names suggest, one tool can recognise only 2D images, while the other is far more powerful and can track 3D objects. [22]

Latest version: Maxst SDK 3.5

Functions:
- 2D marker tracking
- 3D object tracking
- SLAM - Simultaneous Localization and Mapping
- Mlutiple image recignition - up to 200

Platforms:
- Android
- iOS
- macOS
- Windows
- Smart glasses

License:
- Free - Basic
- One-time 499$
- Yearly 599$- with SDK updates

ocitovat knihovny

3.2 Platforms or hybrid libraries

Layar

This is a mobile browser created by the Danish company of the same name that enables content creation in enhanced reality and its subsequent display in the application. [23]

Augment

Augment is a SaaS platform suitable for e-commerce, a developer looking for a way to enable the customer to visualise the product better. Augment also uses technologies such as Vuforia and OpenGL. [24]

8thWall

This platform is used to deliver 6DoF (degrees of freedom) experience for mobile phone cross-platform and integrates seamlessly with ARKit and ARCore with support older devices with or without native AR libraries. [25]

3.3 Comparison of libraries

The comparison will be performed above SDKs listed in [Sec. 3.1]. Despite the fact that some platforms [Sec. 3.2] can extend SDKs capability I will focus on the three features listed below. This comparison shall output 1-3 suitable candidates for an SDK which will be further researched.

1. Functions
2. Usability
3. License

3.3.1 Functions

Functions are further divided into

1. 2D recognition - ability to recognise and track 2D images
2. 3D recognition - ability to track 3D objects
3. Cloud recognition - support of cloud target recognition
4. SLAM - simultaneous localization and surrounding world mapping or its clone
5. Multiple targets - ability to track multiple targets at once
6. Extended tracking

SDK name	2D recognition	3D recognition	Cloud recognition	SLAM	Multiple targets	Excended tracking
ARToolKit	+	-	-	-	-	-
Vuforia	+	+	+	+	+	+
ARKit	+	+	-	+	+	-
ARCore	-	-	-	+	-[1]	-
Wikitude	+	+	+	+	+	+
EasyAR	+	+	+	+	+	-
Maxst	+	+	-	+	+	+

Table 3.1: Function comparison

As can be seen in [Tab. 3.1] 2D and 3D recognition of multiple targets can be considered a standard AR SDK function except for ARCore which is aimed to deliver markless tracking and ARToolkit which merely lacks the functionality. It is to be said that 3D tracking often comes with higher and more professional purchased plan. The cloud recognition is noticeably rarer, whereas SLAM is an advanced feature which comes with almost all latest SDKs, exceded tracking is privileged to Wikitude, Vuforia and Maxst. It should also be mentioned that Vuforia contains more qualities such as Smart terrain and Object scanner whereas Wikitude contains Geo Data synchronisation. From the function perspective, there are three items which are leading from the others. **Vuforia**, **Wikitude** , **Maxst** and **EasyAR**.[3]

3.3.2 Usability

This section is divided into two parts. First it presents comparison of the platforms to which SDKs can be targeted [Tab. 3.2] and then the platforms on which the development can be performed [Tab. 3.3].

SDK name	Android	iOS	UWP	Windows	MacOS	Linux	Smart glasses
ARToolKit	+	+	-	-	+	+	-
Vuforia	+	+	+	+	-	-	+
ARKit	-	+	-	-	-	-	-
ARCore	+	-	-	-	-	-	-
Wikitude	+	+	-	-	-	-	+
EasyAR	+	+	-	+	+	-	-
Maxst	+	+	-	+	+	-	+

Table 3.2: Target platform comparsion

[1] Not applicable

[3] At the time of thesis writing, ARCore introduced a Cloud Anchor point system allowing users to synchronise scenes over the internet, being at the beginning of development the game concept could be adjusted to fit this feature.

SDK name	Java	Objective C	Unity	Unreal	C/C++	JavaScript	Other
ARToolKit	+	-	-	-	+	+	
Vuforia	+	+	+	-	+	-	
ARKit	-	+	-[1]	-	-	-	Xcode
ARCore	+	-	+	+	-	-	
Wikitude	+	+	+	-	-	+	Xamarin
EasyAR	+	+	+	-	+	-	
Maxst	+	-	+	-	+	-	

Table 3.3: Development platform comparison

The platform choice severely depends on the development tool available to the developer and the targeted platform. That said, we aim to use Unity platform and target Android devices which all platforms allow except ARToolKit and ARKit form Apple. Smart glasses are advanced features exclusively for Vuforia, Maxst and Wikitude whereas the only Wikitude supports Xamarin development.

However, there is an additional key feature regarding Unity development which only Vuforia and Maxst deliver. It is the ability to use AR features in the play mode of Unity editor which comes very handy especially during testing. Unfortunately, it is an infrequent feature. There are alternatives (e.g. ARKit Remote, ARInterface) but there is still a need to deploy built application to the smartphone.

From our previously raised preconditions we have a two most suitable SDKs: **Vuforia** and **Maxst**.

3.3.3 License

SDK name	Open source	Basic free	Free-Trial	One-time	Periodic
ARToolKit	+				
Vuforia	-	+	+[2]	499$	99$/m
ARKit	-	+	+	0$	0$
ARCore	-	+	+	0$	0$
Wikitude	-	-	+	1999-2499$	2499-4490$/y
EasyAR	-	+	+	499$	-
Maxst	-	+	+	499$	599$/y

Table 3.4: License comparsion

It is no surprise that the winner of this comparison is ARToolKit which is an open source and is free to use. In special category of platform targeted SDKs belong ARKit and ARCore which are both complementary. Moreover, we have a group of Vuforia, EasyAR and Maxst which all offer one time payment of 499$for reasonably function package development kit.

[1] Exists in experimental version as a plugin in Unity Assetstore [26]
[2] Very limited, only on user defined content

3.4 Quality of tracking

We have concluded several suitable libraries. They are Vuforia, Wikitude, EasyAR, Maxst. Now, we aimed to test a direct tracking accuracy of the multiple targets tracking. We downloaded all libraries, created the demo project and for each library performed research about project set-up. To be objective, the basics knowledges of Vuforia project setting have already been gained by ourselves. Therefore, it is somewhat a subjective view. Despite that, the setup of Maxst was easy, but EasyAR with Wikitude produced few obstacles. However, we were unable to run multitarget tracking on Maxst library for various reasons. For others, I have made a **video** [27] capturing its comparison for multi-target tracking. It has been published on YouTube and is also attached to the CD.

Observations from the test were following:

- EasyAR does not offer sufficient tracking precision. It is unable to track more than three targets which are tracked with already great difficulty.
- Vuforia was able to track four targets with ease and with surprising accuracy and responsiveness.
- Wikitude was able to track four targets with moderate accuracy, but when one target was lost its re-recognition was troublesome.

3.5 Conclusion

From the variety of SKD libraries and depending on planned features, the **Vuforia** library has been chosen. At the time of development start, it offered best feature pack with great support from the developer community including integration into Unity 2017.2. Despite complicated and not suitable license options which could later result in the inability to make free deployment of the application on Google play or other platforms Vuforia was evaluated as the best. At the early stage of development, It was also not yet determined if advanced features will have a fundamental role in the game. Therefore, to avoid being be limited by their absence the more equipped library should be chosen. Two most advanced SDKs were Vuforia and Wikitude which both offers excellent features and community support, but only Vuforia offered target platform of UWP allowing Unity AR play mode and it has also exceeded Wikitude in tracking precision. Had we decided to aim to free application deploy, the more accessible SDK library such as Maxst, EasyAR or ARToolkit would be chosen . With the future knowledge of game function demands and with the EasyAR expected and rumoured support of Extended tracking in future versions it would also be a good choice against Vuforia.

Chapter 4

Design

This chapter will focus on process of game design. Firstly we will introduce the game concept. Secondly the requirements will be declared and lastly the selected game mechanics solutions will be presented.

4.1 Objectives

As was declared in the introduction, main aim of this work is to develop a multiplayer mobile game using augmented reality principles with the innovative game-play. I have presented Vuforia in [Sec. 3.5] as an optimal AR SDK to use for our purpose. It offers excellent tools for real-time marker tracking which we will take advantage of in following game-concept subsection.

4.1.1 Existing game concepts

Nowadays the vast amount of AR multiplayer application is based on simultaneous tracking of the single target by each player and displaying synchronised game content over it. There is minimal interaction with surrounding area, and for the user interface, they use only touch gestures (buttons) on the mobile devices.

4.1.2 Proposed game concept

Each player shall have one unique trackable card and a mobile phone with the running game application. Firstly, players user interface shall consist of not only touchscreen input but also card target location tracking. Therefore game scene will change according to position and rotation of each marker on the table, more specifically the game object representing player (in this case tank) will change barrel orientation and position with marker movement see [Fig. 4.1 b), c)]. Secondly, rotation and position are captured at any time if at least one player is tracking the card target. A game build on top of this proposal must solve synchronization problem among all players and precise distance measurement from centre target to each player target. It is essential for the each running client application to be tracking central target or unexpected behaviour may occur. To improve game entertainment the

game shall implement tank shooting and overall tank interactions (collision, enviroments, etc.). Moreover it shall offer 2-3 game type:

1. **Deatmatch**- players fight against each other. The aim is to destroy the other players

2. **Team deatmatch** - playing team to team. The aim is to destroy the other team

3. **Cooperative** - players are playing together against AI (Turrets, bots)

(a) : Still **(b) :** Rotation **(c) :** Movement

Figure 4.1: Game control

At early stage of design I planned not to use the center target but orientate in the surroundings with a help of markerless tracking. The performed experiments showed acceptable precision for a singleplayer but there is not a universal anchor point for multiplayer game. The major reason why I decided to use center target was therefore impossibility of proper synchronization. In summary, players and cards layout can be seen at [Fig. 4.2].

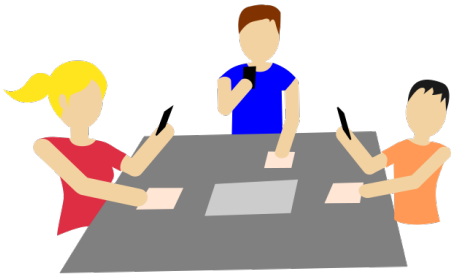


Figure 4.2: Game layout

The UI interface proposed in this paragraph is unique and remarkably differs from already existing games.

4.2 Requirements

4.2.1 Functional requirements

Functional Requirements define the functionality that the application needs to have or the tasks it needs to fulfil in order to achieve the objectives.

F1	The application shall augment captured camera view with virtual content on the display
F2	The application shall be run on a mobile phone
F3	The application shall implement multi-player game
F4	The game shall display game object according to the target position
F5	The game shall allow player movement
F6	The game shall allow interaction among particular game objects
F7	The application shall implement AI bot

Table 4.1: Functional requirements

4.2.2 Performance requirements

Performance requirements quantify to what level the functional requirements will be fulfilled.

P1	The scene changes shall be visible to all players in the real-time
P2	The camera view shall be augmented in real-time
P3	The application shall deliver 6DoF (Degree of freedom) camera behaviour
P4	The bot shall have an ability to move and to damage selected game entities
P5	The game shall be playable by up to four players

Table 4.2: Performance requirements

4.2.3 Design requirements

Aspects that the system needs to fulfil in order to achieve the objectives

D1	The application shall implement the Server-Client architecture
D2	The application shall ensure world scene synchronisation from server to clients
D3	The game shall update player position accordingly the target tracked position
D4	The clients shall update server about all currently tracked images
D5	Important updates shall be sent over reliable channel
D6	Less important or periodic updates shall be sent over unreliable channel
D7	The application shall create local world scene for each client
D8	The application shall be run on Android 7.0 or newer
D9	The application shall be developed in Unity editor and language C#
D10	The application shall spawn selected objects on the server and clone it on the clients
D11	The application shall keep a local world scene calibrated although the central target is lost
D12	The application shall have a synchronisation procedure at the start of the game

Table 4.3: Design requirements

4.2.4 Operational requirements

Operational requirements are requirements that the system has to fulfil to be handled and operated safely and reliably.

O1	The application shall by compliant with latest software
O2	The mobile devices shall be connected to the same network for the local game creation
O3	The mobile devices shall be connected to the internet for the internet game creation
O4	Any user UI input shall be tested
O5	The user text input shall be controlled for non-standard characters

Table 4.4: Operational requirements

4.2.5 Constrains

Constraints are those things which limit cost, schedule and implementation techniques available to the developer.

C1	The application shall be completed prior to the thesis handout

Table 4.5: Constrains

4.3 Background

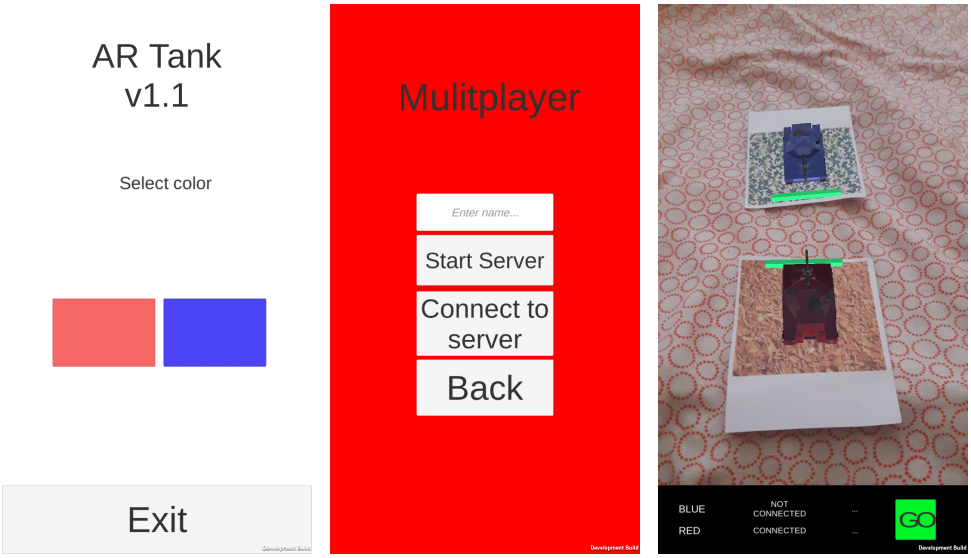
Prior to the game design, technology research and familiarising had been executed. My experiences with Unity were broadened in the development of the Furherous Housewife game in the course HRY, and outcome of the Projects subject was an application where I gained skills in Vuforia AR library.

4.3.1 Game predecessor

The game with working title "AR Tank" was a game developed in 2017 by me from which current game is directly derived. However, they share only game mechanism and background in Vuforia and Unity at best. The elementary principle was a game of 2 players playing against each other.

Each player is equipped with a unique paper card with symbols which is captured by both players phones camera. These symbols came from the internet, were colourised and adjusted the size to the circa 10x10cm which can be seen in [Fig. 4.3 c)]. The green trackable produced better results as Vuforia library works with black-and-white images and thus its better contrast image showed considerably better results in tracking, and consequently, the orange trackable image showed poorer results as it had a drab picture with a contrast absence. The tank 3D model is then displayed on the card and does move with it but only statically, therefore real environment is augmented, but the tank has no other control that the card location. The tank is periodically shooting a projectile, and its collision with the environment is resolved locally. The reason for that was imprecision of the network communication which had been established by own communication protocol build on top of the LLAPI (low-level API) integrated into Unity. That proved not to be a right decision as there were many special cases in which my protocol was unreliable and overall unstable. This approach is to be changed in future development to more advanced HLAPI (high-level API).

The game principle was relatively simple and not much entertaining as it had only limited functionality, but it was a good place where to test AR capabilities of Vuforia SDK and generally improve development skills. There were also several issues risen regarding the future game evolution. Firstly it was mainly the problem of proper synchronisation among all players which showed to be relatively ambitious on LLAPI. Secondly indisputable need of better trackable images.



(a) : Main menu **(b) :** Connecting page **(c) :** Game screen

Figure 4.3: AR Tank v1.1

4.4 User interface

The user interface, according to the proposed game concepts [Sec. 4.1.2], consists of two parts. Firstly the printed image trackable movement and secondly the standard touchscreen input which is to be further divided into the standard menu and HUD (head-up-display). The [Fig. 4.4] demonstrates the user input and game control capabilities. [28]

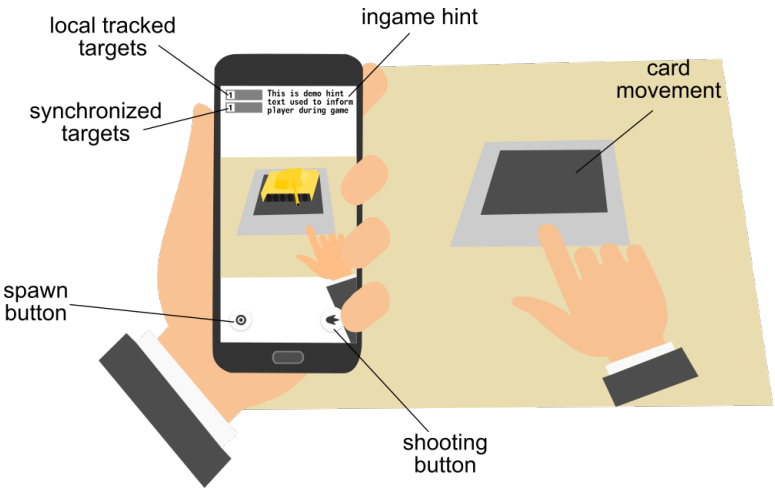


Figure 4.4: User interfaces

4.4.1 Trackable movement

Movement of the player object follows the same pattern as the [Fig. 4.2] describes. It shall detect a change of position, change of rotation and overall visibility.

4.4.2 Menu

The main goal of the menu is to allow player choose game type, map, and establish the connection among players. The minor intention is to offer an other useful functions or informations such as Game rules, About game, Options Menu. The game shall, therefore, consists of 7 self-reliant screens (unique display overlay): StartMenu, NetworkMenu, StartGame, ServerList, Lobby, Options, About and four dependent, not full screen overlaying panels: Info, Countdown, TopPanel and InGamePanel. Part of the menu is to be included in the project from the third-party source [29] (Unity) in the form of prefab and then severely edited.

4.4.3 HUD

The HUD is in the form of upper, and bottom part See [Fig. 4.4] which will allow the user to see the in-game hint, currently tracked targets and offers various button occurrence depending on the game type chosen. The player's and enemy's health is displayed above each game-object with health script in the form of a health-bar and is not included in HUD.

4.5 Target tracking

4.5.1 Marker image selection

One of the goals of target tracking in AR game development is to provide an AR SDK with suitable target images. To be easily trackable, images must match several criteria. The [Table 4.6] introduces several rules which should be adhered to.

Attribute	**Example**
Rich in detail	Street-scene, group of people, collages and mixtures of items, or sport scenes
Good contrast	Has both bright and dark regions, is well lit, and not dull in brightness or color
No repetitive patterns	Grassy field, the front of a modern house with identical windows, and other regular grids and patterns

Table 4.6: Suggested image attributes

In total the game use five single-image targets [Appendix C]. They are called

Center, **1**,**2**,**3**,**4** and the last four were downloaded from [30] where they had been published under CC0 (Creative Common zero) license and later greatly edited to follow trackable guidelines in [Tab. 4.6] which resulted in adding contrast, changing color to black-and-white, adjusting brightness, cropping and rescaling of the picture. In addition to that, the number representing the id of player has been placed to each trackable image. The first trackable called Center is however original picture provided by Vuforia.

4.5.2 Target detection

The Vuforia tracks targets in the way that it sets an image game objects transform according to the recognised marker in the currently searched frame. As can be seen in the [Fig. 4.5] tracked target are "floating" in the scene to service the illusion which tries to adjust virtually augmented objects scale, position and rotation into the real world. Distances between these floating game objects do not represent real-world ratio and therefore we need to introduce a system which would allow us to determine a precise measurement of card layout because all the paper cards position can be represented in the two-dimensional space. First of all, to proper initialise the scene we need to determine the centre of the world. That can be done in the AR Camera (prefab from Vuforia) configuration of World Center Mode, we will use SPECIFIT-TARGET which will be set on previously mentioned centre marker. Besides the centre target, we also have a four trackable player targets but for current demonstration and explanation purposes lets introduce only one [Appendix C].[Fig. 4.5] shows AR camera capturing and positioning floating game objects of **center target** and **trackable** into the game scene. Our approach is to shoot a ray from the camera in the direction and through the trackable game object and determine its collision location with reference ground which is described as a grey pad underneath. In the figure we can see 2 rays (red - to player target, green - center target).

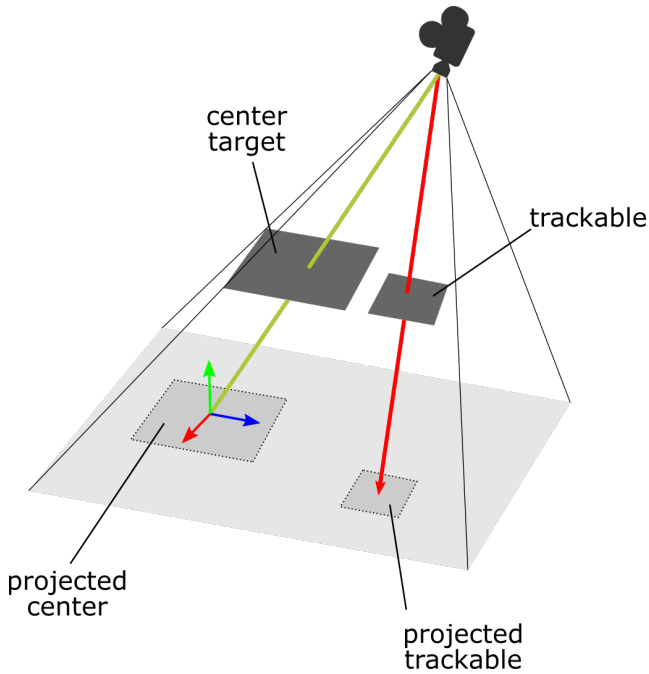


Figure 4.5: Target location raycasting

4.5.3 Target synchronization

As can be seen in [Fig. 4.6] trackable target synchronisation is designed to hold an information locally on the clients and remotely on the server. Client triggers communication chain when the state change is triggered internally by *TrackableEventHandler.OnTrackableStateChanged()* public void. Then the message about newly tracked or lost target shall be transmitted to the server using network **[Command]** attribute of NetworkBehaviour in HLAPI which serves clients as a tool for triggering functions which will be executed only on the server (any function arguments must be serializable). The opposite way of communication (Server to Client) is achieved with **[ClientRpc]** call which can be triggered only on Server and executes the function on every connected client. [31]

Server is furthermore responsible for the client's tracked lists management and updating of synchronized list which is set to be a SyncVar. **[SyncVar]** is an attribute which allows labelled variable to be automatically synchronized between Server and Clients and updated on any change. All mentioned attributes will be widely used all across the project as they provide a simple and reliable way of network communication.

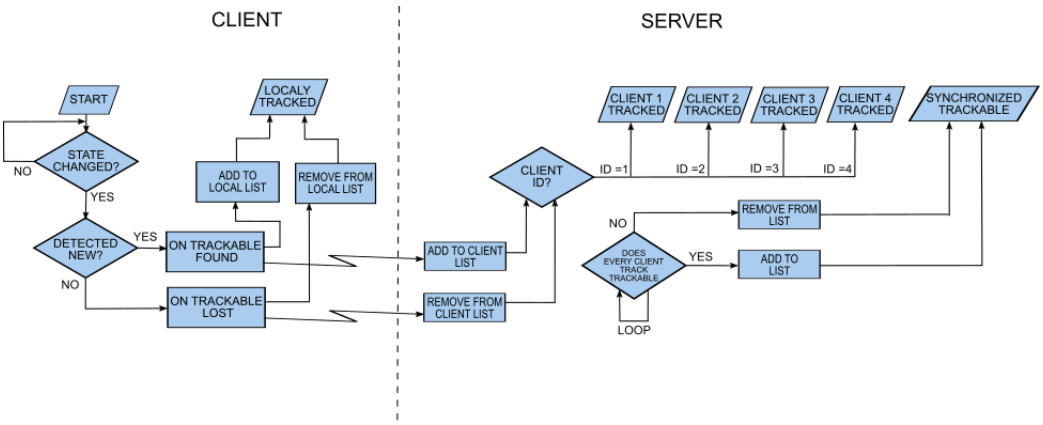


Figure 4.6: Target found/lost flowchart

Besides, the problem of target discovery and losing our game must implement a periodical update system which will provide the current position of target despite possible tracking interruption in some clients. That is achieved by PlayerNetwork script which sends an update in the form of Command in the previously set period of time. All received positions are written to the SynVar variables on the server and distributed back to all clients. Despite the fact that it might be costly in the form of network traffic it is a reliable approach which in case tracking interruption keeps updating the client with the position of the targets. There are undoubtedly space for improvements and optimisation with particular steps to be suggested in later chapters.

Chapter 5

Implementation

In this chapter, the main explanation of the multiplayer mobile game development is presented. Firstly, used developer tools will be demonstrated. The architecture and technical description will follow. At the end of this chapter, the game loop together with the third-party libraries and the encountered problems will be described.

5.1 Development environments

We have implemented the development instrument in C# language with using the Unity API in the form of Unity project.

5.1.1 Unity

Unity is a cross-platform game engine developed by Unity Technologies, which is primarily used to develop both three-dimensional and two-dimensional video games and simulations for computers, consoles, and mobile devices. Since the beginning of the development, I have gradually upgraded from version 2017.1 to 2017.4.1f1. Some upgrades were more severe others less, but the important milestone was the integration of Vuforia SDK into Unity 2017.2. [32], [33]

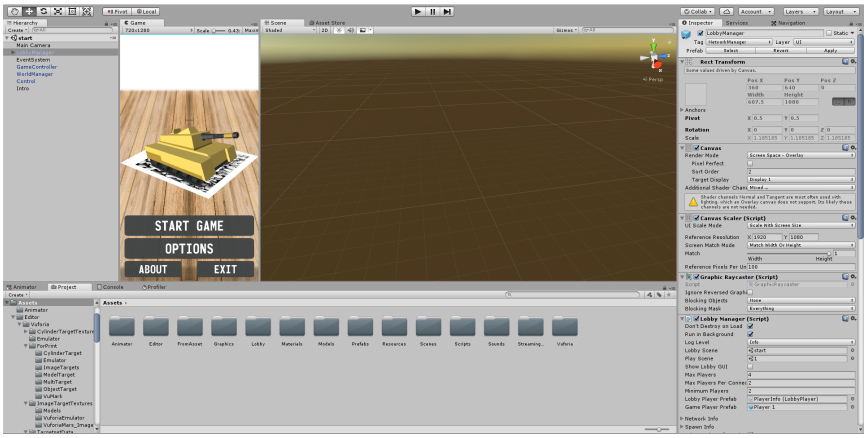


Figure 5.1: Unity Editor

5.1.2 Visual studio 2017

Visual studio 2017 in an IDE editor developed by Microsoft used in the version 15.1 under Community school license. It was used for code writing and editing as well as for debugging using its integrated debbuger.

5.2 Development tools

This section will demonstrate using important, advanced or non-standard development tools. Development tools allow developers to implement, test, and debug their code. During the development following tools were used:

5.2.1 Vuforia target manager

Vuforia target manager is an online development tool allowing uploading and management of the image markers. It requires registration which also provides App Licence Key necessary to run an application as well as the necessity to have enabled Vuforia in Unity. It is also used for target quality evaluation from which the trackable images in form of prefab can be downloaded and loaded into Unity Editor.

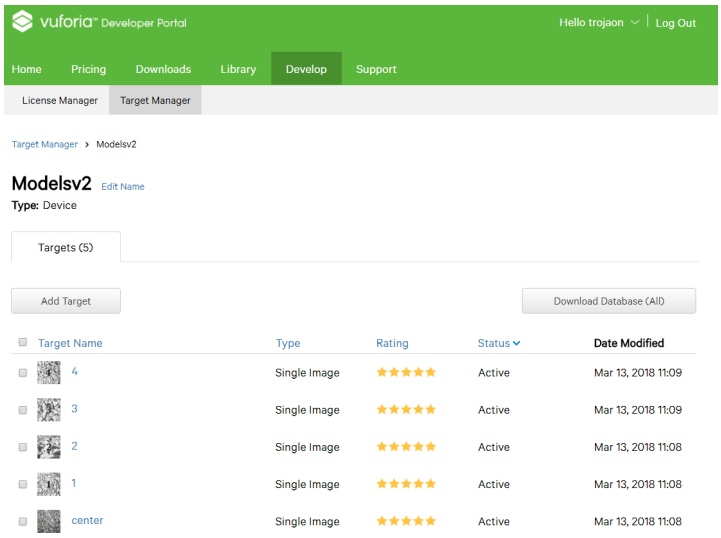


Figure 5.2: Vuforia target manager

5.2.2 .NET tools

Linq

Linq - Language Integrated Query is a language integrated into .NET Framework from version C# 3.0. What is unique about this neat tool is the ability to query various data types such as List, Enumerable, Dictionaries but also on Arrays in the same intuitive way as you would use in SQL commands.

Following list shows and explains few most essential queries and the Listing [Code 5.1] supports the presentation with a demonstration of usage. [34]

- Select - is an operator which performs a projection on the collection to select interesting aspects of the elements.
- Where - The Where operator allows the definition of a set of predicate rules that are evaluated for each object in the collection, while objects that do not match the rule are filtered away.
- Order By - The OrderBy operator is used to specify the primary sort ordering of the elements in a collection according to the key.
- Single - The Single operator takes a predicate and returns the element that matches the predicate.

Listing 5.1: LINQ demonstration

```
List<Item> testList = new List<Item>() {
            new Item(1, 2),
            new Item(1, 3),
            new Item(2, 4)};
//Returns objects compliant with the predicate
IEnumerable<Item> whereTest = testList.Where(x => x.item1 ==
    1);
//Orders the list according to the key selector
IEnumerable<Item> orderTest = testList.OrderBy(x => x.item2);
//Returns the selector for each item in list
IEnumerable<int> selectTest = testList.Select(x=> x.item2);
//Returns the object compliant with the predicate
Item singleTest = testList.Single(x => x.item1 == 2);
\label{code:linq}
```

Delegates

The delegate is a reference type variable that holds an reference to a method. They are often used for implementing events and the call-back methods and work in the way that their method can be overwritten. [odkaz na Microsoft] The following code chunk [Listing 5.2] shows the suggested practice in the LobbyManager which I have fully adopted and have used ever since. Public function GoBackButton() is triggered on back button click and it consequently triggers the backDelegate() on the line 5. The core concept is that the content of the BackButtonDelegate on line 2 can be changed accordingly to the currently active screen. [35]

,

Listing 5.2: Delegate demonstration

```
1 public delegate void BackButtonDelegate();
2 public BackButtonDelegate backDelegate;
3 public void GoBackButton()
4 {
```

```
  backDelegate();
}
\label{code:deleg}
```

5.2.3 Unity tools

Unity framework

Unity overall offers many tools and mechanics, but it is essential to understand how the basic object lifecycle works. The more detailed explanation in the reference [36] par First of all, any Gameobject shall extend either class Monobehaviour or NetworkBehaviour (For networking functionality). These classes give any object-capability to be initialised and updated in the game loop. Therefore when the game object is instantiated following methods are executed: **Awake** - this function is always called before any Start function, **OnEnable** - this function is called just after the object is enabled and **Start** - is called before the first frame update only when the script instance is enabled. Then **Fixed Update** - is often called more frequently than Update. It can be called multiple times per frame, **Update** - is called once per frame. It is the main workhorse function for frame updates, **Late Update** - LateUpdate is called once per frame, after Update has finished. [33]

Coroutine

Coroutine is a function which can pause its execution (yield) and wait for some conditions to fulfil. It is convenient when using procedural animations or any events with runtime longer than one Update cycle. Standard coroutine updates are run after the Update function returns. The coroutine is started as: ,

Listing 5.3: Coroutine start

```
1 StartCoroutine(RespawnLocalCountdown( 1f, done =>
2 {
3   //Some action (Respawn)
4 }));
```

When the coroutine is finished the pre-set System.Action<bool> is triggered in this case respawn on the line 3. The [Listing 5.4] shows countdown coroutine used to respawn player and which updates the countdown text visible on the screen until time is up. Notice that line 12 marks the end of the coroutine.

Listing 5.4: Local respawn croutine

```
1 public IEnumerator RespawnLocalCountdown(float time, System.
      Action<bool> done)
2     {
3         float remainingTime = time;
4         int floorTime = Mathf.FloorToInt(remainingTime);
5
```

```
        while (remainingTime > 0)
        {
            yield return null;
            remainingTime -= Time.deltaTime;
            if (remainingTime <= 0)
            {
                done(true);
            }
            int newFloorTime = Mathf.FloorToInt(remainingTime
                );
            if (newFloorTime != floorTime)
            {
                floorTime = newFloorTime;
                UpdateCountdown(floorTime);
            }
        }
        UpdateCountdown(0);
    }
```

5.2.4 Code version control system

Version control is a system that records changes made during the development. There are three main approaches in version controlling:

1. Local Version Control Systems - all changes are stored solely on the single device
2. Centralized Version Control Systems - versions are stored on the remote server and all clients access it over network (e.g., Subversion [SVN], CVS)
3. Distributed Version Control Systems - versions are stored on the remote server, but all clients have local copy of the repository and are able to make changes independently (e.g., Git or Mercurial)

The application development was undertaken using Apache Subversion and TortoiseSVN software tool.

Thesis was written in latex and versioned with all graphs, images and drawings using GIT on university web GitLab.

5.3 Architecture

The game consists of two scenes (Lobby scene and Game scene). In the first one, there are presented the menu and Network manager which is used for network communication and connection establishment. The game scene contains tracked targets prefabs provided by Vuforia SDK and HUD. Overall **5500** lines of code were written and **44** original scripts created.

5.3.1 Logic distribution

With a desire to make project as aspect oriented as possible there are several logic distributions to make code clean and safe. However despite my higher

effort not every method is well located. The list below show assorted classes responsible for particular objective:

1. **TargetController** - is used for trackable target management and game startup/restart settings.
2. **WorldController** - takes care of proper world instantiation, initialization and spawning from server to clients.
3. **NetworkController** - is used for network connection establishment and management.
4. **PlayerController** - script which is on the Player game object prefab takes care of the game objects position, color and rotation manegement.
5. **Control** - static game object holding current instances of several managers and controllers.

5.3.2 Network management

As was suggested in the game-predecessor AR Tank game, its follower shall use HLAPI [37] which is a set of networking commands used to efficiently implement multiplayer game see [Fig. 5.3] as a Server-Client communication. The core of this approach is the Network Manager which is equipped with functions and tools for connection establishment and maintenance. In our case, we have used the LobbyManager which is a specialised type of Network-Manager that provides a multiplayer lobby before entering the main play scene of the game. It is developed by Unity Technologies and available on Asset store for free. Consequently, game objects which used network functionality must have extended NetworkBehaviour instead of MonoBehaviour and its game object must have contained script Network Identity representing the object which is known in the network among all users. Furthermore, all players contain players game object, in my case Tank model which contains script PlayerNetwork which periodically sends updates of currently tracked targets in the form of Command with parameters (int ConnectionId,Vector3 position, Quaternion rotation) and size of (4B+12B+16B) = **32**Bytes.

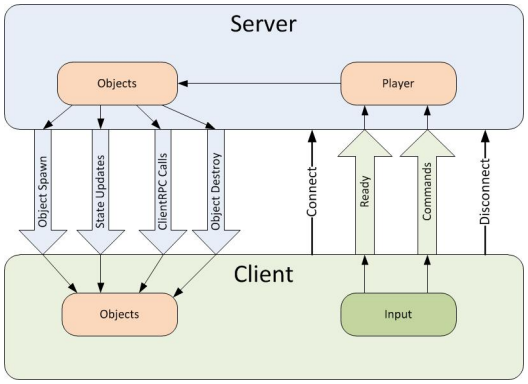


Figure 5.3: Remote actions graph [31]

Any object which is to be spawned (instantiated on clients and maintained by the server) must be registered as a spawnable prefab in Network manager either by the inspector or by a script.

Network traffic

The game has three main interfaces through which the majority of traffic goes. Firstly it is Command update tracked position which is sent from each client-tracked card to the server. Secondly, the SyncVar update from the server to all clients about the currently tracked card location and Finally the Network transform updates which quantity depends on the number of networked objects (Bots, Projectiles, Turrets). Majority of networked scripts use default update interval of 0.1 seconds. Some less critical only interval of 1 second.

From bandwidth use perspective the server side varies from **24** to **48** Kbps during two-player game and client's side from **16** to **24** Kbps.

5.3.3 World generation

In total, the game contains five maps (3 - deathmatch, 2 - cooperative). The game has introduced a system for world game object spawning which differs from the standard unity scene system. Each map is represented as a prefab and is spawned as any other network object with a network identity. The reason for that was a desire for a dynamic world changing the setup. In the way that the world could change its shape, models position and practically enlarge itself dynamically. The secondary reason for that was a possibility to spawn more worlds in one scene. Cons of this approach are slight "uncertainty" of worlds generation in case any error appears in its process and possible worse performance with high-poly larger world game objects.

World generation is done in the World Manager

5.3.4 Real-time navigation mesh

The absence of a static scene with obstacle game objects and pre-baked navigation meshes consequently led me to a more dynamic approach of mesh generation. The suitable choice was the NavMesh building component [38] which is not included in Unity Editor but needs to be obtained from Unity GitHub page. It allows the user to be baking meshes during run-time and connect NavMesh surfaces together.

5.4 Game

This section describes important steps in game loop from connection start, game start, gameplay to the end of selected game type.

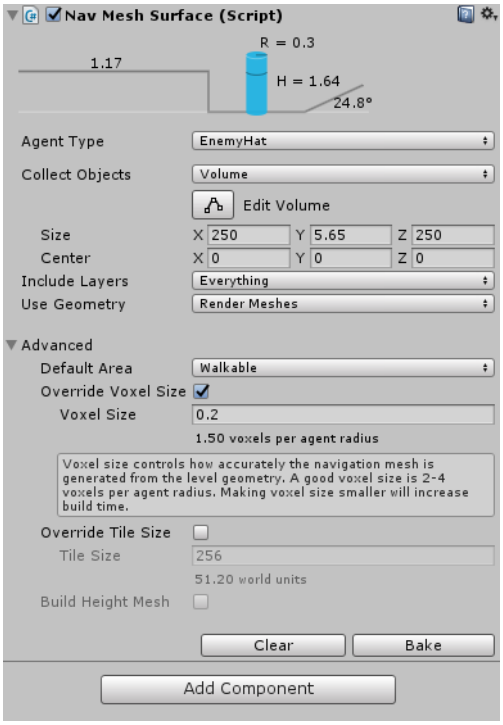


Figure 5.4: NavMesh Surface component

5.4.1 Game loop

Lobby

Lobby describes several interconnected menus which serve as a graphical tool for the player for connection establishment, game setting and pre-game interaction. The player enters the main lobby [Fig. 5.5a] and proceed either as a client or a server. Then player continue to next lobby [Fig. 5.5b] where he shall set its player's colour and card number which directly represents physical target markers showed in [Appendix C]. When is ready to enter the game he clicks to the JOIN button and after all players have confirmed that they are ready the game begins.

Start game

When the game starts the *ServerChangeScene(string sceneName)* is triggered by the server and results in parallel loading on each client. Depending on the speed and overall performance of the player's device one can load the scene faster than the other and therefore has a slight advantage in the game start. For that reason, initial centre tracking calibration has been implemented which creates synchronised countdown see [Fig. 5.6a] after all players are tracking centre target (can be disabled in Options). Regardless the scene change after proper initialisation selected world instantiation is executed. Likewise, it is triggered by the server, and the whole world is spawned at the

(a) : Main Lobby

(b) : Pre-game lobby

Figure 5.5: Lobby

very end of scene loading which is detected in *SceneManager()* which further execute *OnSceneLoad()* method handling the game scene startup including the world generation. For that, there is a particular class *WorldManager* which also sets the game rules (Cooperative/Deathmatch) and hold an instance of the specific world prefab. Players game object (Tank) is spawned above the pre-set spawn point which belongs to the World prefab more specifically to the *PlaySceneInitialize* script. Their location can vary but the direction should math the marked player number on the Center target [Appendix C].

Respawn

If the player dies depending on the game type, the in-game menu appears see [Fig. 5.6b] and offers the respawn which is implemented as a coroutine in [Listing 5.4] and is either local or synchronised. Respawn has pre-set waiting time which makes the player wait and in some way punish him for the death. After the time is up, respawn button is set as intractable, and the player can return to the game. In addition to that player's tank location was reset to the start spawn point set in the world object.

End

When the game is set to be ended depending on the game type the server shows an in-game menu offering going back to the lobby or stay in the scene.

(a) : Start countdown

(b) : Kill screen

Figure 5.6: Gameplay

5.4.2 Gameplay

According to the game proposal in [Sec 4.1.2] game implements two types of the gameplay

Cooperative

In this scenario, players are not fighting against each other but on the contrary, collaborate in various target destroying. If we take the demonstrative world of the green bunker, it contains bunker model as a passive environment and a couple of turrets which autonomously find and shoot at the player. The player shall follow the game hint which describes currently active task. To properly end the cooperative game player needs first to destroy all turrets and then destroy the bunker. That results in the game end triggering and offers an in-game menu with the choice of going back to lobby. The cooperative game running screenshots are in Appendix [D.5, D.6]

Deatmatch

Players are playing against each other, and the main goal is to destroy the opponent in various environments. In contrary to the cooperative gameplay, players can spawn a bot which targets nearest attackable entity (enemy player or enemy bot). The deathmatch game running screenshots are in Appendix [D.3, D.4]

5.5 Third party libraries

During the development of the application several the third party libraries were used. The following sections contain names of the libraries and their purpose in the project

- **Area 730** - contains model of tank and other various military models
- **Free LowPoly Desert Pack** - library from which additional models were obtained
- **LowPoly Enviroment Pack** - low poly models
- **Offices Supplies Low Poly** - models
- **Simple FX** - library with particle system prefabs
- **Standart Asset** - standard Unity downloadable asset
- **Toony Tiny People** - demo pack of humanoid models with animations
- **Unity UI Extensions** - extension of standard Unity UI system
- **Open-Sans** - font
- **Lobby Manager** - extension of Network Manager from Unity technologies
- **Nav Mesh Component** - unity library for advanced and dynamic navigation mesh generation

5.6 Problems

Despite the fact that Unity is very advanced and neat Game engine, several bugs were encountered during the development. Most of them were fixed during new Unity Editor version release but some still persists. List of the most severe bugs is:

- Gameobjects transform are not being disabled when the player is stopped
- Can not send network message receiver can not keep up with the amount of data sent when deep-profiling
- Applying prefab changes changes recttransform properties
- Network Discovery thrown errors (to be reported)
- NetworkManager Error: Server client disconnect error
- Vuforia on Android is not compatible with .NET 4.6

Moreover, an additional bugs with Vuforia were experienced. When the game was launched in the editor, it has regularly crashed and threw the error screen of [Fig. 5.7], and console log *Vuforia cannot be started before it is initialised.* This behaviour has been observed on all versions of Unity from 2017.1 to 2017.4, and its frequency increased over time. The only solution for this bug was to shut-down Unity editor and relaunch it. The bug was reported.

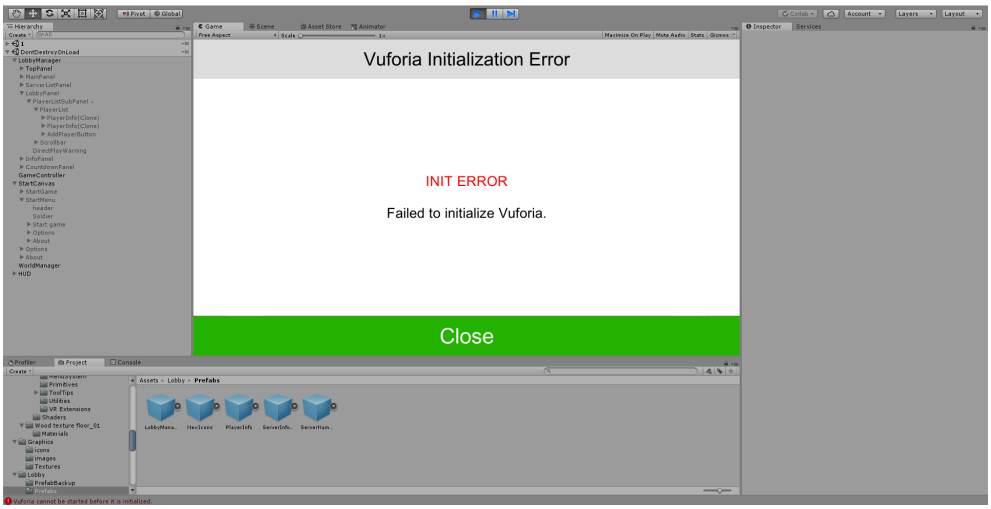


Figure 5.7: Vuforia error

Ultimately, the screen's orientation behaves oddly. The game launches as landscape, and when the scene is switched to the play-scene, it orientates to portrait and keeps there. This bug came with the update from Unity 2017.3 to 2017.4 where Vuforia changed from 7.0 to 7.1.

Chapter 6

Evaluation

6.1 Requirement verification

The requirements raised in the section [4.2] are to be validated and verified to check that application fulfils its intended purpose. To briefly point-out the work objectives, my task was to develop a multi-player game using augmented reality on a mobile phone.

To reliably check the requirements we will use a system of the verification matrix with four following validation methods:

- Verification by test **(T)** - Verification by test is performed by subjecting the application to a physical test
- Verification by inspection **(I)** - Verification by inspection is performed by simply inspecting/looking at the application.
- Verification by analysis or similarity **(A)**. Verification by analysis is performed by a simulation on some parts of the application. Verification by similarity is performed by stating that a part of the application is similar to a part that has already marked as tested
- Verification by review-of-design **(R)**. Verification by review-of-design uses documentations to show that the application or its component will perform as expected.

Id	Requirement	Verification	Status	Test No.
F1	The application shall augment captured camera view with virtual content on display	T, I, R	Done	Test 1
F2	The application shall be run on a mobile phone	T, I, R	Done	Test 1
F3	The application shall implement multi-player game	T, I	Done	Test 1
F4	The game shall display game object according to the target position	T, I	Done	Test 1
F5	The game shall allow player movement	T, I	Done	Test 1
F6	The game shall allow interaction among particular game objects	T, I	Done	Test 1
F7	The application shall implement AI bot	T, I	Done	Test 1
P1	The scene changes shall be visible to all players in real-time	T, I, R	Done	Test 1
P2	The camera view shall be augmented in real-time	T, I, R	Done	Test 1
P3	The application shall deliver 6DoF (Degree of freedom) camera behaviour	T, I	Done	Test 1
P4	The bot shall have the ability to move and to damage selected game entities	T, I	Done	Test 1
P5	The game shall be playable up to four players	T, I	Done	Test 2
D1	The application shall implement the Server-Client architecture	T, I, R	Done	Test 1
D2	The application shall ensure world scene synchronisation from server to clients	T, I, R, A	Done	Test 1
D3	The game shall update player position accordingly the target tracked position	T, I, A	Done	Test 1
D4	The clients shall update server about all currently tracked images	T, I, A	Done	Test 1
D5	Important updates shall be sent over reliable channel	I, A	Done	
D6	Less important or periodic updates shall be sent over unreliable channel	I, A	Done	

D7	The application shall create local world scene for each client	T, I	Done	Test 1
D8	The application be run on Android 6.0 or newer	T, I	Done	Test 1
D9	The application be developed in Unity editor and language C#	I	Done	
D10	The application shall spawn selected objects on the server and clone it on the clients	T, I	Done	Test 1
D11	The application shall keep local world scene calibrated although the central target is lost	T, I	Done	Test 1
D12	The application shall have a synchronisation procedure at the start of the game	T, I	Done	Test 1
O1	The application shall by compliant with latest software	T, R, A	Not tested	Test 3
O2	The mobile devices shall be connected to the same network for the local game creation	T, A	Done	Test 1, Test 3
O3	The mobile devices shall be connected to the internet for the internet game creation	T, A	Done	Test 3
O4	Any user UI input shall be tested	T, A	Done	Test 3
O5	The user text input shall be controlled for non-standard characters	T, A	Done	Test 3

Table 6.1: Verification matrix

6.1.1 Requirements tests

This part presents 3 test validating all requirements marked as verifiable by test **(T)** from verification matrix [6.1]

Participants	2
Test procedure	1. Two players shall connect to the game over a local network on two Android 7.0+ smartphones 2. Participants begin the game with calibration procedure and load scenes 3. Then check if content is augmented over the targets in the real-time. 4. Check augmentation position and rotation change when phone camera moves and rotates around the target 5. Players check that synchronised target tracking (upper left) panel (lower) are the same across both players 6. Players check synchronised tracking on real player object (only one player tracks the specific target, the other shall received its location with respect to the center target) 7. Player intentionally loose centre target tracking and point camera of the centre, then check if the scene holds its aspects 8. Then each player tries shoot the projectile to the ground 9. Player tries to shoot a projectile on the other player 10. Both targets spawn bots, then awaits for their mutual destruction 11. The first target spawns the bot; the second wait with no action till the bot destroy it
Verification of	F1, F2, F3, F4, F5, F6, F7, D1, D2, D3, D4, D7 ,D8 ,D10, D11, D12, P1, P2, P3, P4, O2
Test completed	Done (Successful)
Results	All tasks have been completed and verified. The sudden crash of the application was experienced. Probably due to the low battery of the phone. The battery save-mode was on.

Table 6.2: Test 1

Participants	4
Test procedure	1. Four players shall create the game the game and repeat the procedure in Test 1
Verification of	P5
Test 2.1 completed	Done (Unsuccessful)
Test 2.2 completed	Done (Successful)
Results of Test 2.1	The game of 4 player was unable to be created. The reason for that was maximum player set-up in Lobby Manager of 3. The test will be repeated
Results of Test 2.2	The game was tried to run with 4 players (1 computer and 3 mobiles). All task in Test 1 were accomplished.

Table 6.3: Test 2

Participants	2
Test procedure	1. Game is to be run on the android 8.0 Oreo 2. The player creates the game over a local network, tests basics functions and disconnects 3. The player creates the game over the internet connection, tests basics functions and disconnects 4. Player tries to insert non-standard characters (# $& ! *) to the player name text input field
Verification of	O1, O2, O3, O4
Test completed	Done (Partially successful)
Results	The game was tested on Android 7.0 for the reason that I have not had possession of the device with the newest Android 8.0. Game start over internet and local network was successful. Characters are rejected on player name, the IP input field allows them but throws an error

Table 6.4: Test 3

6.2 Usability test

This section will focus on group testing regarding applications game-play, design, control and user interface. For that reason 3 scenarios will be introduced which shall evaluate objectives raised in [Sec. 4.1.2].

Participants

	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
Name	Eva	Julian	Gerit	Kasia	Francesco
Age	23	27	24	23	23
Nationality	Czech	German	German	Polish	Italian
Field of study	IT	Political sciences	Urban planning	Pharmacy	Literature
Do you play mobile games?	Yes	Yes, regular computer mostly	Not a lot	Computer and phone	Rarely
What type of games do you play?	Logical, arcade, candy crush type	Strategical (Heardstone)	Playstation (FIFA)	"Click and Go" type	Playstation
How many hours do you spend playing/day	20 min	30 min	5 min	2 hours	3 hours but rarely
Do you have experience with AR	No	Have an idea, similar to VR	No	No	No

Table 6.5: Participants initial survey

Used equipment

Each participant had a possession of a mobile phone with an Android system. Their comparison are mentioned in [Tab. 6.6]

Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
Xiaomi Redmi 4X, Android 7.1.2	Samsung Galaxy A3 2016, Android 7.0	Samsung Galaxy A5, Android 7.0	Xiaomi Redmi 4X, Android 7.1.2	Ulefone Power 3S

Table 6.6: Participants devices

6.2.1 Test plan

At first participants were asked their first name, age, nationality, university and field of study. To have a better profiled participants few basics question will be asked and then additional explanation about the game, proposed game concept, UI and overall control will be given. They will also be notified about existing bugs and issues mentioned in [6.4].

The basic scenario is to make a connection to the game:

1. The first objective will be to start the application.
2. One of the participants must act as a server to whom others connect.
3. The one creates a local game (server) and select game type and select map.

4. Other (client) players will press "find local game" button and wait for the lobby to appear.
5. Paper cards will be distributed according to the player's location in respect to the centre target.
6. Players shall select its card number in lobby according to the number they were given.
7. Players can choose a colour freely. When are ready they shall press the ready button and wait for others to finish.
8. When the scene is loaded calibration sequence is initialised therefore all players must first track centre target to start the game.

Following the basics test above. 3 game scenarios will be tested on the participants.

- **Map gameplay** - Players shall perform a fight on the chosen map in which they can use shooting and bot spawn. The fight is not time-limited, the game has no end. Expected playtime is 5-15 minutes and players can freely disconnect from the game at any time. Meanwhile they are left to try and play the game.
- **Cooperative gameplay** - In this scenario, players shall cooperate in turrets and fort destroying. The game ends when the fort is destroyed. Players are left free to experiment and win the game.
- **UI testing** - Participants are asked to discover a menu of the game.

When all scenarios had been finished the after testing survey were given. They full answers are in the [Tab. 6.7] below.

6.2.2 Test results

Some of the collected verbal answers were reformulated and transcribed into the [Tab. 6.7]

	Participant 1	Participant 2	Participant 3	Participant 4	Participant 5
Comments on deatmatch?	I have succeeded in connecting, game creation. Map (Algebra) was nice. Prefer not to kill the own bots.	Cool but without point, boring, death does not matter, bots no purpose	When 3 players are playing, Unfair for the middle one	A bit chaotic	-
Comments on cooperative?	I enjoyed this more	No real opponent, just turret no challenge	Cooperative is better	Cooperative is better	-
UI observation	The rules shall inform the user about the necessity of tracking their own and centre target simultaneously.	The game should mention the right angle for tracking	-	No leaderboard	-
Did you have any difficulties during the task completing?	No	Starting process hard, you need to aim from specific angel	Agree with participant 2	No	-
Did the application behaved as you have expected?	Yes	Sometimes it was difficult, slippery, form of holding, not steady hand, shaking image, aiming hard	The moving of tank is difficult, maybe easier way of moving	Few problems, no movement, tank slow	No
Have you experienced severe bugs or errors?	Yes, tested map was still under development, fort was not destroyable	No	No	One crash	Severe, unable to play
Have you experienced unexpected loss of target tracking?	No major loss	No	When moving through	No	-
Have you experienced any advantages/disadvantages due to the loss of tracking?	No	-	no	No I was still at the same position	-
Did you have difficulties to hit other players and bots?	Normal difficulties	Process of aiming is not reliable and is hard	Sometimes when you press the shooting button and wait it shoot on short distance	No problem with aiming	-

How do you like the game design?	Yes 4/5	Yes	Yes	Yes	Yes
Did you like the application?	Yes, it is interesting but not my cup of tea	Yes	Yes	Yes	No
Which game type did you like the most?	Cooperative	Both advantages and disadvantages	Cooperative	Cooperative	-
Would you download this game?	No (Interesting, but I do not like games of this type)	Cannot objectively answer	Depends on description, logo and advertising	Agree with 2. participant, tank is a bad thing	-
Overall impression?	I think the game concept is nice and interesting, and have potential but it is not a game for me (tanks, shootings), the map with algebra was interesting. Tanks could be better distinguishable, tank's whole body in chosen colour.	Lot of potential, still raw, lot of glitches, need to improve experience	The fluent movement would make the game better, control is difficult, the idea is nice, 3D animation is nice, the idea of bots is good but not well made	I want score and leaderboard	-

Table 6.7: After testing survey

Comment on the testing process: The test was undertaken at 22:00 pm under artificial lighting conditions. The game for four players was unachievable due to the bug which limited maximum players to three. That is to be fixed, and the new test will be performed. Others three players successfully created the game both on the local network and over the internet. Players went through all three scenarios and played three maps (Office, Fort green and Fort dust). One application crash was experienced on the Xiaomi device.

6.2.3 Observations

Participants showed enthusiasm and interest for the game and liked the displayed augmented reality objects which was something new for them. Only three players played the game because of the bug. And the fourth player showed regret that he cannot participate during the gameplay. Overall, players liked the idea and game concept and preferred the Cooperative game type. However, according to some, the game presented itself in a bit raw view due to the control difficulties, especially at the beginning. Firstly, often mistake the players have made was pointing the camera from the wrong angle. The wrong angle together with bad lighting sometimes made the tracking difficult. The wrong angle consequently led to lousy tracking, and that led to not accurate behaviour. Secondly, some players had difficulties with shooting

and spawning. The buttons change colour when the minimal timeout for fire expire which participants did not notice from the beginning and tended to press it more frequently. After a while of practice players adopted the principle as was intended. Furthermore, some felt the game to be a bit hard during deathmatch and not challenging during the cooperative. The dead did not matter to the participants as the player can respawn in few seconds. They would appreciate more severe punishment.

The conclusions have been made according to the answer survey which can be seen in the [Tab. 6.7]. Overall, they have agreed on following points:

- Game has a potential
- Control needs some improvement
- The game should point out the need for tracking targets simultaneously under right angle.
- Shooting sometimes felt shifty
- AI Bots interesting, needs some adjustment
- Tanks are slow and not attractive for girls

6.3 Future work

Concerning the above testing and its outputs, some changes and adjustments of control shall be made. More specifically the tank movement shall be more fluent and projectile shooting easy to use and more accurate. Furthermore, the tank enlarging could be beneficial for the overall better scene orientation and could solve problems of difficult aiming and complicated handling.

From the functionality perspective, some participants asked for the leaderboard and score system which will be generally easy to do as well as the change of tank model which is decidedly losing with the girl audience. The necessity of the right angle and camera position shall be emphasised and presented more clearly and distinctly. Better button state change (clickable and unclickable) shall be determined.

Lastly depending the audience reaction, the application could be published on application distribution server (PlayStore) and distributed to end users. The primary and most notable obstacle is a need of Vuforia license purchasing which starts at 499 $as a one-time fee. Moreover, the full versions of other assets 5.5 would need to be obtained. That said and with zero monetisation plan, the application consequently needs to be unprofitable and lossy. To overcome this obstacle, we would need to gather the sufficient amount of initial funds, create a business plan with a deep income/expense description and proceed with the risk of active project maintenance. The second variant (if licenses allow) would be to distribute this project as an open-source on Github. Lastly, the game could be utilised for university and department presentation and advertisement purpose.

6.4 Known bugs

- When the player tries to delete its player in the lobby, application freezes
- The options reset its value when the user returns from game to menu

6.5 Known issues

When the application was developed in the library of University of Cagliari in Italy, odd behaviour had been encountered when the devices were connected to local Eduroam network. In addition to that, no ping between devices could be received. The conclusion from this uncomfortable event was except more challenging development a warning from misconfigured routers or firewalls.

Moreover, the local network communication does not work on some hotspot wifi network.

Chapter 7

Conclusion

The primary focus of this work was to design [Chap. 4] and develop [Chap. 5] a functional multiplayer game using augmented reality library which was determined as a second task in the [Chap. 3]. Development in Unity was unquestionably the most comfortable experience available with no competitor capable of such fluent and reliable tool. Despite the fact that several severe bugs have been encountered [Sec. 5.6] , the development was generally smooth and straightforward. The most challenging task was surprisingly the multi-player part with obstacles such as improper synchronisation or connection/disconnection problems. The use of Unity's HLAPI was undoubtedly more efficient than LLAPI. However, the use of Lobby Manager as an asset was an unfortunate solution because its usage was insufficiently documented and with poor guidelines. Luckily the Unity technologies released the full source code for inspection. Therefore, it slightly compensated the lousy documentation.

Augmented reality experience done with Vuforia target tracking was on the other hand generally easy to achieve and use. The Vuforia documentation is well made and is generally reliable. Its community is growing and likewise very helpful. The precision of target tracking is at this point sufficient although heavily dependent on the phone selection. With future expected technological advancement the tracking could become more and more reliable which could result in better game experience. At the time of work finish, the ARCore introduced new Cloud Anchor Point feature which allows users to synchronise scene over the network markerless and could massively improve the game experience.

The complications for the game release were mentioned in the [Sec. 6.3]. In addition to that according to the testing results, some adjustments need to be done. However, most importantly the player ought to focus on pointing the camera in the way he tracks centre target and his selected number target simultaneously. That is a crucial aspect of the game and needs to be better communicated to the user. Had I been more focusing on the game-play design, The game could have been more entertaining and challenging for its player. Meanwhile, the only two game-types serve as an appropriate demonstration of new UI interface proposed in [Sec. 4.1.2].

References

[1] (2017). After mixed year, mobile ar to drive $108 billion vr/ar market by 2021, [Online]. Available: www.digi-capital.com/news/2017/01/after-mixed-year-mobile-ar-to-drive-108-billion-vrar-market-by-2021/ (visited on 05/03/2018).

[2] B. Gleb and V. Mykola. (c2011-2018). Best tools for building augmented reality mobile apps, [Online]. Available: https://rubygarage.org/blog/best-tools-for-building-augmented-reality-mobile-apps (visited on 05/03/2018).

[3] S. Mann. (1999). Mediated reality: University of toronto rwm project, [Online]. Available: http://www.linuxjournal.com/article/3265? (visited on 03/29/2018).

[4] S. Mann. (1998). An historical account of the 'wearcomp' and 'wearcam' inventions developed for applications in 'personal imaging', Computer mediated reality, [Online]. Available: http://www.wearcam.org/historical/node12.html (visited on 03/29/2018).

[5] P. Milgram and F. Kishino. (). A taxonomy of mixed reality visual displays, [Online]. Available: http://etclab.mie.utoronto.ca/people/paul_dir/IEICE94/ieice.html (visited on 03/29/2018).

[6] B. Bray and M. Zeler. (). What is mixed reality?, [Online]. Available: https://docs.microsoft.com/en-us/windows/mixed-reality/mixed-reality (visited on 03/29/2018).

[7] (2015). Detection of aruco markers, [Online]. Available: https://docs.opencv.org/3.1.0/d5/dae/tutorial_aruco_detection.html (visited on 05/03/2018).

[8] *What is markerless augmented reality?*, St. Petersburg, FL, c2018. [Online]. Available: https://www.marxentlabs.com/what-is-markerless-augmented-reality-dead-reckoning/ (visited on 05/04/2018).

[9] (2008). Optical see-through vs. video see-through, [Online]. Available: http://sensics.com/optical-see-through-vs-video-see-through/ (visited on 05/03/2018).

[10] (2001-). Pokémon go, [Online]. Available: https://en.wikipedia.org/wiki/Pok%C3%A9mon_Go (visited on 05/07/2018).

[11] (2001-). Word lens, [Online]. Available: https://en.wikipedia.org/wiki/Word_Lens (visited on 05/07/2018).

[12] (2001-). Snapchat, [Online]. Available: https://en.wikipedia.org/wiki/Snapchat (visited on 04/04/2018).

[13] (). Instagram stories vs snapchat stories, 2017 statistics, [Online]. Available: https://www.socialreport.com/insights/article/115005343286-Instagram-Stories-Vs-Snapchat-Stories-2017-Statistics (visited on 04/04/2018).

[14] (c2018). Ikea place (arkit app), [Online]. Available: https://augmented-reality-apps.com/resource/ikea-place/ (visited on 05/07/2018).

[15] (c2018). Artoolkit, [Online]. Available: https://github.com/artoolkit (visited on 05/07/2018).

[16] (c2018). Vuforia, [Online]. Available: www.vuforia.com (visited on 05/07/2018).

[17] (). What's new with arkit in ios 11.3, [Online]. Available: https://www.imore.com/arkit-updates (visited on 04/05/2018).

[18] (c2018). Arkit, [Online]. Available: developer.apple.com/arkit/ (visited on 05/07/2018).

[19] (2018). Arcore, [Online]. Available: developers.google.com/ar/ (visited on 05/07/2018).

[20] (c2018). Wikitude, [Online]. Available: www.wikitude.com (visited on 05/07/2018).

[21] (c2012-2018). Easyar, [Online]. Available: www.easyar.com (visited on 05/07/2018).

[22] (c2010-2018). Maxst, [Online]. Available: http://maxst.com/ (visited on 05/07/2018).

[23] (). Layar, [Online]. Available: www.layar.com (visited on 05/07/2018).

[24] (). Augment, [Online]. Available: http://www.augment.com (visited on 05/07/2018).

[25] (). 8th wall, [Online]. Available: https://www.8thwall.com/ (visited on 05/07/2018).

[26] (2017). Unity arkit plugin, [Online]. Available: https://assetstore.unity.com/packages/essentials/tutorial-projects/unity-arkit-plugin-92515 (visited on 05/02/2018).

[27] O. Trojan. (). Multitarget comparison, easyar, wikitude, vuforia, [Online]. Available: https://www.youtube.com/watch?v=DQ1YJ4OSuus (visited on 05/04/2018).

[28] M. Lanham, *Augmented Reality Game Development*. Packt Publishing, 2017, ISBN: 1787122883 9781787122888.

[29] (2017). Unity arkit plugin, [Online]. Available: https://assetstore.unity.com/packages/essentials/network-lobby-41836 (visited on 05/02/2018).

[30] (). Photo public domain, [Online]. Available: http://www.photos-public-domain.com/ (visited on 05/07/2018).

[31] (c2018). Remote actions, [Online]. Available: https://docs.unity3d.com/Manual/UNetActions.html (visited on 05/07/2018).

[32] (c2011-2018). Getting started with vuforia in unity, [Online]. Available: https://library.vuforia.com/articles/Training/getting-started-with-vuforia-in-unity.html (visited on 05/12/2018).

[33] A. Watkins, *Creating games with Unity and Maya, how to develop fun and marketable 3D games*. Burlington, MA: Focal Press, c2011, ISBN: 978-024-0818-818.

[34] (c2018). Language integrated query (linq), [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/ (visited on 05/12/2018).

[35] (c2018). Introduction to delegates, [Online]. Available: https://docs.microsoft.com/en-us/dotnet/csharp/delegates-overview (visited on 05/12/2018).

[36] (c2018). Execution order of event functions, [Online]. Available: https://docs.unity3d.com/Manual/ExecutionOrder.html (visited on 05/12/2018).

[37] (c2018). The multiplayer high level api, [Online]. Available: https://docs.unity3d.com/Manual/UNetUsingHLAPI.html (visited on 05/12/2018).

[38] (c2018). Navmesh building components, [Online]. Available: https://docs.unity3d.com/Manual/NavMesh-BuildingComponents.html (visited on 05/07/2018).

Appendix A

List of Shortcuts

Shortcut	Meaning
UI	User Interface
AR	Augment Reality
MR	Mixed Reality
VR	Virtual Reality
SDK	Software Development Kit
API	Application Programming Interface
UWP	Universal Widows Platform
GPS	Global Positioning System
HUD	Head-Up Display
LLAPI	Low-Level Application Programming Interface
HLAPI	High-Level Application Programming Interface
IT	Information Technologies

Appendix B

Content of attached CD

Folder	Content
Source	Folder contains source code
Targets	Folder contains 2 pdf files with target pictures
ThesisTEX	Folder with TEX version of thesis
Videos	Videos of performed tracking quality tests and finished game demonstration
BCv1.0.apk	Last build of game apk
Thesis.pdf	Thesis in form of pdf

Appendix C

Target images


1


2


3


4

Image Target:

Vuforia™

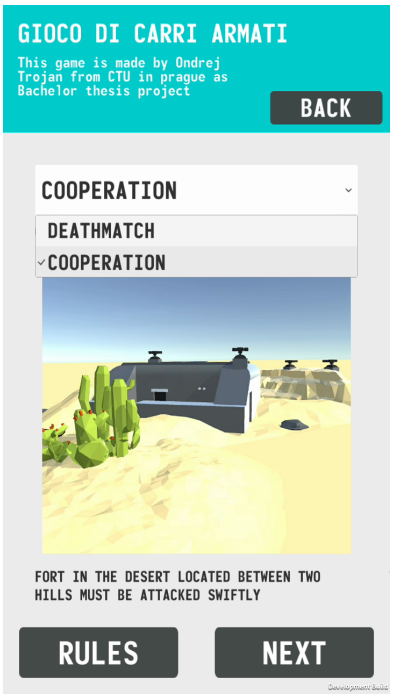
Appendix D

Game screenshots

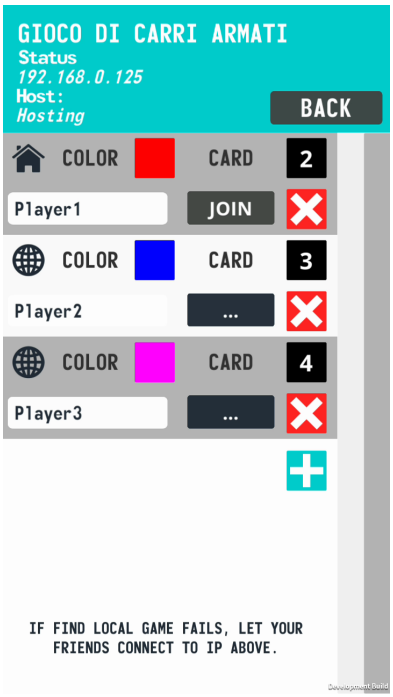
(a) : Start menu

(b) : Main lobby

Figure D.1: Menu I



(a) : Game type selection



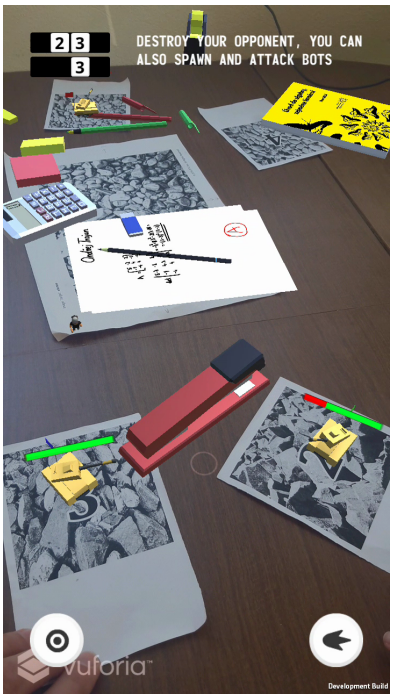
(b) : Pre-game lobby

Figure D.2: Menu II

(a) : Map Dust

(b) : AI Bots spawning

Figure D.3: Deathmatch I

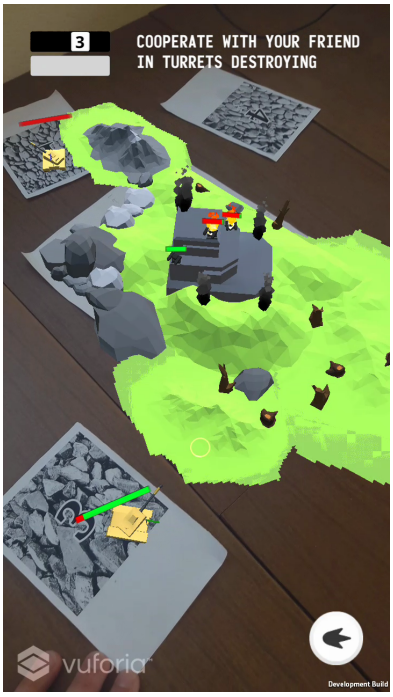


(a) : Map Study

(b) : Kill menu

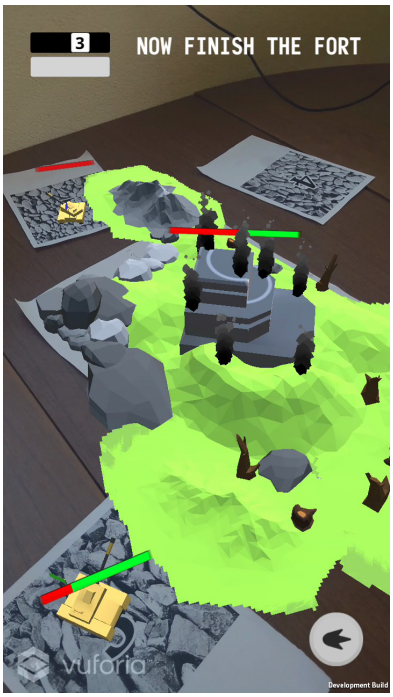
Figure D.4: Deathmatch II

(a) : Fort turrets hull health

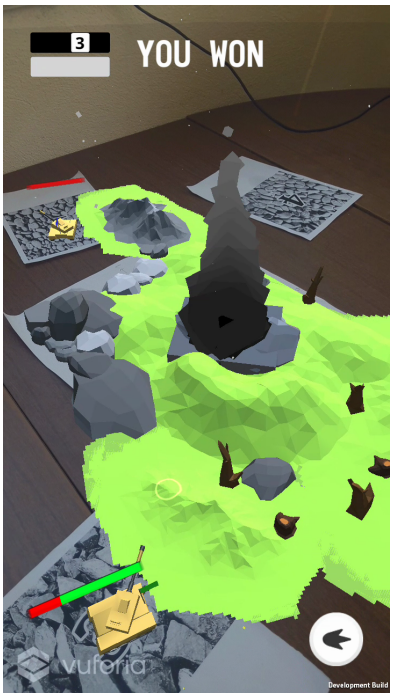


(b) : Fort turrets damaged

Figure D.5: Cooperative I



(a) : Fort attacking



(b) : Fort destroyed

Figure D.6: Cooperative II

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Trojan** Jméno: **Ondřej** Osobní číslo: **435005**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra řídicí techniky**

Studijní program: **Kybernetika a robotika**

Studijní obor: **Systémy a řízení**

II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

Víceuživatelská hra v rozšířené realitě na mobilní platformě

Název bakalářské práce anglicky:

Multiplayer mobile game development using augmented reality

Pokyny pro vypracování:

Prostudujte problematiku vývoje rozšířené reality (AR) na mobilní platformě a porovnejte dostupné knihovny. Vyberte vhodnou AR knihovnu pro vývoj hry na mobilní platformě a předveďte její základní vlastnosti. Navrhněte hru pro více hráčů, jejíž hlavní mechanikou (herním konceptem) bude právě rozšířená realita. Hru implementujte v prostředí Unity s vámi vybranou AR knihovnou. Proveďte testování s uživateli a vyhodnoťte pozorování.

Seznam doporučené literatury:

[1] Augmented Reality Game Development, Micheal Lanham, Packt Publishing, 2017
[2] Creating Games with Unity and Maya, Adam Watkins, Focal Press, 2011

Jméno a pracoviště vedoucí(ho) bakalářské práce:

Ing. David Sedláček, Ph.D., Katedra počítačové grafiky a interakce

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

Datum zadání bakalářské práce: **16.01.2018** Termín odevzdání bakalářské práce: **25.05.2018**

Platnost zadání bakalářské práce: **30.09.2019**

Ing. David Sedláček, Ph.D. podpis vedoucí(ho) práce	prof. Ing. Michael Šebek, DrSc. podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Student bere na vědomí, že je povinen vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

Datum převzetí zadání	Podpis studenta