

# Outdoors Mobile Augmented Reality for Coastal Erosion Visualization based on Geographical Data

Minas Katsiokalis

May 2020



Electrical And Computer Engineering,  
Technical University Of Crete

Thesis Committee

Associate Professor Katerina Mania (Thesis Supervisor)

Associate Professor Dionysia Kolokotsa

Associate Professor Vasileios Samoladas

# Acknowledgement

This thesis is a result of a collective effort. As a token of my appreciation I would like to mention some of the contributors.

First, I would like to thank my Supervisor, Associate Professor Katerina Mania for her supervision throughout this whole process. Next I would like to thank Dr. Lemonia Ragia for her invaluable knowledge on the field of coastal erosion and for providing the geographical data of this project. Her advice and guidance was of big importance.

Next, I would like to thank Eleftheria Gogonaki, for her infinite patience, assistance and support during the whole process. Without her the process would have become way more difficult in many aspects.

I also want to thank my friends and collaborators, for providing psychological support, ideas and useful instructions. Special thanks to my friend Maria Sgourou, a valuable partner during all these years of my studies providing invaluable help.

Big love to Danadas Squad (DnDs) for all those years of friendship and for having faith in me.

And last but not least, I would like to express my gratitude and infinite love to my family members: My father George, my mother Katerina and my little brother Demosthenes for all the psychological and economic support all these years despite the hard times that they have been through.

In memory of my grandfather Minas and my grandmother Kaiti, which I recently lost.

## Περίληψη

Σε αυτή τη διπλωματική εργασία παρουσιάζουμε μια εφαρμογή επαυξημένης πραγματικότητας σε κινητές πλατφόρμες για την οπτικοποίηση παράκτιας διάβρωσης βασιζόμενη σε γεωγραφικά δεδομένα που αφορούν την παραλία της Γεωργιούπολης στον νομό Χανίων. Ο κύριος στόχος αυτού του έργου είναι να παρέχει ένα μέσο για την τρισδιάστατη οπτικοποίηση της μελλοντικής κατάστασης της παραλίας και να αυξήσει την ευαισθητοποίηση του κοινού σχετικά με το φαινόμενο παράκτιας διάβρωσης που λαμβάνει χώρα σε πολλές παράκτιες περιοχές του νησιού της Κρήτης. Σε αυτήν την εφαρμογή έχουμε δύο μελλοντικά σενάρια σε τρεις διαφορετικές τοποθεσίες της παραλίας. Το πρώτο σενάριο δείχνει την παραλία με 3,6 μέτρα εισχώρηση στην ενδοχώρα και το δεύτερο σενάριο δείχνει την παραλία με 7,7 μέτρα εισχώρηση. Και τα δύο σενάρια αντιστοιχούν στην τάση της παραλίας μετά την ελάχιστη και τη μέγιστη προσδοκώμενη αύξηση της στάθμης της θάλασσας σε χρονικό διάστημα 80 ετών. Η βελτίωση της τεχνολογίας των κινητών τηλεφώνων έχει δώσει πρόσβαση σε εμπειρίες Επαυξημένης Πραγματικότητας στο ευρύ κοινό, χρησιμοποιώντας την κάμερα, το GPS και τους εσωτερικούς αισθητήρες που υπάρχουν στα σύγχρονα smartphones. Με την επίσκεψη στις τρεις τοποθεσίες, μια εικονική σκηνή αντιστοιχίζεται με την γεωγραφική θέση του χρήστη και μετά από μια σύντομη διαδικασία, ο χρήστης μπορεί να βιώσει την 3D αναπαράσταση. Η καταγραφή της θέσης του χρήστη πραγματοποιείται χρησιμοποιώντας το GPS του τηλεφώνου καθώς και με τους αλγόριθμους υπολογιστικής όρασης του επιλεγμένου εργαλείου AR. Σχεδιάστηκε και ενσωματώθηκε μια εφαρμογή βασιζόμενη στην τοποθεσία για να εξασφαλιστεί η φόρτωση του σωστού περιεχομένου σε κάθε τοποθεσία, αποφεύγοντας την άσκοπη φόρτωση γραφικών στοιχείων καθώς επίσης να διασφαλιστεί ότι ο χρήστης βρίσκεται στην περιοχή της Γεωργιούπολης. Η εφαρμογή παρέχει έναν χάρτη που μπορεί να χρησιμοποιηθεί για την ενημέρωση του χρήστη σχετικά με τις τοποθεσίες ενδιαφέροντος, καθώς και για τη λήψη οδηγιών και την επίσκεψη σε όποια από αυτές επιθυμεί. Η εφαρμογή κάνει χρήση σύγχρονων εργαλείων AR και μεθόδων rendering για ανάπτυξη εφαρμογών σε κινητές πλατφόρμες. Συνδυάζοντας την τεχνολογία AR με γεωχωρικά δεδομένα στοχεύουμε να βελτιώσουμε την κατανόηση των δεδομένων αυτών από τους χρήστες και να αυξήσουμε την επίγνωση του κοινού για κρίσιμα περιβαλλοντικά φαινόμενα όπως η διάβρωση των ακτών, εστιάζοντας σε μια περιοχή υψηλού κινδύνου όπως η Γεωργιούπολη.

# Abstract

In this thesis we present a mobile augmented reality application for coastal erosion visualization based on geographical data, at the beach of Georgiupoli in Chania. The main focus of this work is to provide a mean for the 3D on-site visualization of the future state of the beach and increase the awareness of the public audience about the coastal erosion effect that takes place in many coastal areas of the Crete island. In this application we feature two future scenarios in three different locations of the beach. The first scenario is showing the beach under 3.6 meters retreat inland and the second scenario showing the beach under 7.7 meters retreat inland. Both scenarios correspond to the tendency of the beach after the minimum and maximum sea level rise expectation (SLR) in a time interval of 80 years. Advances in mobile technology have brought Augmented Reality to the wider public by utilizing the camera, GPS and inertial sensors present in modern smartphones. Upon visiting these locations a virtual scene is matched to the user's position and after a short process the user can experience the visualization. Position tracking is performed by utilizing the phone's GPS and the computer vision capabilities of the chosen AR framework. A location aware experience was designed and integrated to ensure the loading of the right content at each location, avoiding unnecessary rendering of graphics and ensuring user is located in the area of Georgiupoli. The application provides a map which can be used to notify the user about the locations of interest as well as to get directions and visit whichever he/she desires. The application makes use of modern AR frameworks and rendering methods for mobile AR development. By combining AR technologies with geo-spatial data we aim to enhance user's understanding of those data and increase people's knowledge on crucial environmental phenomena like coastal erosion, focusing on a high risk area such as Georgiupoli.

# Contents

<b>1</b>	<b>Introduction</b>	<b>13</b>
1.1	Brief Description . . . . .	13
1.2	Thesis Structure . . . . .	15
<b>2</b>	<b>Augmented Reality (AR)</b>	<b>16</b>
2.1	Introduction to AR . . . . .	16
2.2	History of AR . . . . .	17
2.3	Current state of AR . . . . .	23
2.3.1	AR Head Mounted Displays (HMDs) . . . . .	24
2.3.2	Mobile Augmented Reality (MAR) . . . . .	30
2.4	Use of AR Nowadays . . . . .	32
2.4.1	Modern AR Apps . . . . .	32
2.4.2	AR For Environmental Purpose . . . . .	35
2.5	The Registration Problem . . . . .	41
2.5.1	Fiducial Marker Based Tracking . . . . .	42
2.5.2	Natural Feature Tracking . . . . .	42
2.5.3	Model Based Tracking . . . . .	43
2.5.4	Markerless Tracking . . . . .	44
2.5.5	Sensor Based Tracking . . . . .	45
2.5.6	Hybrid Tracking . . . . .	46
2.6	Developing Platforms & Software . . . . .	47
2.6.1	AR Software Development Kits (SDKs) . . . . .	47
2.6.2	Game Engines . . . . .	54
2.6.3	Platform of our choice . . . . .	56
2.7	Future of AR . . . . .	58
<b>3</b>	<b>Case Analysis on Coastal Erosion</b>	<b>60</b>
3.1	Introduction . . . . .	60
3.2	Our Case Analysis . . . . .	61
3.2.1	Economic Analysis . . . . .	61
3.2.2	Environmental Analysis . . . . .	64
3.3	Data Showcase . . . . .	65
3.3.1	Coastal Retreat Extraction . . . . .	65
3.3.2	Visualization of Data . . . . .	66
3.3.3	Tables of Data . . . . .	68
<b>4</b>	<b>Requirements Analysis</b>	<b>72</b>
4.1	Introduction . . . . .	72
4.2	Pre-Requirements . . . . .	72
4.2.1	General Requirements . . . . .	72
4.2.2	Augmented Reality Requirements . . . . .	73
4.2.3	Map and Navigation Requirements . . . . .	74
4.3	Use Case Scenarios . . . . .	75
4.3.1	User Enters the Application . . . . .	75
4.3.2	User is Located Outside of Georgioupoli . . . . .	76
4.3.3	User Enters the Map . . . . .	77
4.3.4	User Enters AR View . . . . .	79

4.3.5	User Experiences the AR Visualization . . . . .	81
4.4	Hardware Requirements . . . . .	83
<b>5</b>	<b>End User Experience</b>	<b>84</b>
5.1	Introduction to the App . . . . .	84
5.2	Initial/Welcome Screen . . . . .	85
5.3	Not-in-area Screen . . . . .	86
5.4	Map Screens . . . . .	87
5.4.1	Entering Map View . . . . .	87
5.4.2	Choosing Location . . . . .	88
5.4.3	Navigating to Selected Location . . . . .	89
5.4.4	Reaching Selected Location . . . . .	89
5.5	AR Screens . . . . .	90
5.5.1	Entering AR View . . . . .	90
5.5.2	Scanning the Environment . . . . .	91
5.5.3	Detecting Planes & Shoreline . . . . .	91
5.5.4	Placing Virtual Content . . . . .	92
5.5.5	AR visualization . . . . .	94
5.5.6	Beginning Visualization . . . . .	94
5.5.7	First AR Scenario . . . . .	95
5.5.8	Second AR Scenario . . . . .	96
5.5.9	Info Signs Enabled . . . . .	96
5.5.10	Information Panels . . . . .	97
5.6	Usage of the App . . . . .	99
5.6.1	Giving Access to Hardware . . . . .	99
5.6.2	Helpful Tips . . . . .	99
<b>6</b>	<b>Implementation</b>	<b>101</b>
6.1	Developing Platform . . . . .	101
6.1.1	Unity3D . . . . .	101
6.1.2	Android Studio . . . . .	101
6.1.3	Hardware . . . . .	102
6.2	Software & Packages . . . . .	102
6.2.1	ARCore . . . . .	102
6.2.2	Mapbox . . . . .	103
6.2.3	AR Foundation . . . . .	104
6.2.4	Lightweight Render Pipeline . . . . .	105
6.2.5	Shader Graph . . . . .	107
6.2.6	Rest of Packages . . . . .	108
6.3	Structure of the App . . . . .	109
6.4	Welcome Scene . . . . .	111
6.4.1	Welcome Scene Components . . . . .	111
6.4.2	Interactions . . . . .	112
6.5	Map Scene . . . . .	114
6.5.1	Map Scene Components . . . . .	114
6.5.2	Integrating Mapbox . . . . .	115
6.5.3	Self-Indicator Functionality . . . . .	117
6.5.4	Directions Implementation . . . . .	118
6.5.5	Checking Location & Distance . . . . .	121

6.6	AR Scene . . . . .	124
6.6.1	AR Scene Components . . . . .	124
6.6.2	Setting Up AR Foundation . . . . .	125
6.6.3	Creating Water Shader . . . . .	127
6.6.4	Designing Virtual Content . . . . .	131
6.6.5	Interactivity in AR . . . . .	133
<b>7</b>	<b>Conclusion</b>	<b>136</b>
7.1	Summary . . . . .	136
7.2	Evaluation . . . . .	136
7.2.1	Technical Characteristics . . . . .	136
7.2.2	Goal and Functionality . . . . .	137
7.3	Future Work . . . . .	138
7.3.1	Increase Environmental Understanding . . . . .	138
7.3.2	Improve Shoreline Detection Method . . . . .	139
7.3.3	Visual Upgrade & Optimization . . . . .	139
7.3.4	HMD implementation . . . . .	140
7.3.5	Adding More Scenarios . . . . .	140
<b>8</b>	<b>Bibliography</b>	<b>141</b>

## List of Figures

1	(a) Present (Real) (b) Future (Real + Virtual) . . . . .	14
2	Milgram's Reality - Virtuality Continuum [1] . . . . .	16
3	Ivan Sutherland's first Augmented Reality System [2] . . . . .	18
4	Caudell and Mizell coining AR in 1992 [3] . . . . .	18
5	Touring Machine by Feiner et al. [4] . . . . .	20
6	ARToolKit for pose tracking in 6DOF [5] . . . . .	20
7	ARQuake by Thomas et al. [6] . . . . .	21
8	(a) Siemens SX1 AR game "Mozzies".[5] (b) Multi-user AR applica- tion for handheld devices "The Invisible Train".[7] . . . . .	22
9	(a) AR-Tennis by Henrysson et al.[8] (b) Wikitude AR Browser.[5] . .	23
10	Microsoft HoloLens HMD [37] . . . . .	25
11	Windows 10 on Microsoft HoloLens [38] . . . . .	25
12	Microsoft HoloLens 2 HMD [37] . . . . .	26
13	Microsoft HoloLens 2 on Healthcare [37] . . . . .	27
14	Magic Leap One HMD [39] . . . . .	27
15	Magic Leap One Gesture Recognition [40] . . . . .	28
16	(a) Magic Leap Lightpack (b) Magic Leap Control [39] . . . . .	28
17	(a) Nreal Light Glasses, Controller and mini PC (b) Nreal Light Glasses: look like sunglasses [41] . . . . .	29
18	AR Training in the Workplace [35] . . . . .	30
19	AR Navigation using Smartphone [36] . . . . .	31
20	IKEA Place: Virtual Couch in Living Room . . . . .	32
21	Toyota Hybrid AR: In-depth Information . . . . .	33
22	Google Translate: AR Sign Translation . . . . .	34
23	Night Sky: AR View of Virgo . . . . .	34
24	Unbelievable Bus Shelter: Tentacle Illusion . . . . .	35
25	Energy Source application workflow [9] . . . . .	36
26	Climate Change application workflow [9] . . . . .	37
27	(Left) Wind Power, (Right) Climate Deforestation [9] . . . . .	37
28	Eco-discovery AR-based learning model (EDALM) [10] . . . . .	38
29	(a) Screenshot of an interactive virtual plant silhouette (b) Learning content about specific plants [10] . . . . .	39
30	Browser/Authoring tool. AR tracking showing side menu, interaction annulus, and two triangulated points [11] . . . . .	39
31	Example point annotations [11] . . . . .	40
32	Flooding with building geometry on (Left), and geometry off (Right) [11] . . . . .	40
33	Augmented Reality on Magazine [42] . . . . .	42
34	Natural Feature Tracking . . . . .	43
35	Model Based Tracking of industrial objects by using CAD models [43] .	44
36	Plane Detection of Horizontal (blue) and Vertical (pink) Surfaces [44] .	44
37	Mobile coordinate system and orientation relative to the Earth's frame of reference . . . . .	45
38	ARCore Depth API, Occlusion off (left) and Occlusion On (right) [26] .	48
39	ARKit 3 People Occlusion, Occlusion off (left) and Occlusion On (right) [29] . . . . .	50



40	Vuforia Cylinder Targets [31]	52
41	Wikitude Object Tracking [30]	54
42	Functionality Provided by Game Engines	54
43	AR Foundation development compared to ARCore & ARKit development [32]	57
44	Location of Georgioupoli, Crete, Greece	61
45	Georgioupoli's Beach	62
46	Coastal zone changes in Georgioupoli [12]	63
47	Minimum and maximum retreats of Cretan beaches for sea level rises of (a) 0.82 m and (b) 1.86m estimated on the basis of the low and high mean of the model ensemble projections. Final widths values less than zero show beaches that will be entirely lost. 71 Beaches Showcase. [13]	65
48	Shoreline Retreat Scenarios for SLR 0.5m/1m [12]	67
49	Initial Use Case	75
50	User isn't in Georgioupoli Use Case	76
51	Map Use Case	77
52	Map Settings Use Case	78
53	AR View Use Case	79
54	AR Calibration Use Case	81
55	AR Visualization Use Case	82
56	App Basic Structure and Navigation	84
57	Welcome Screen	85
58	Not-in-area Screen (a) Light Map (b) Dark Map	86
59	Map Screen: Note	87
60	Map Screen: Humanoid/Light Map	87
61	Map Screen: Pin Arrow/Dark Map	88
62	Map Screen: Map Settings	88
63	Map Screen: (a) Select Location & Enable Directions (b) Hide Settings	89
64	Map Screen: AR Button	90
65	AR Screen: Note	90
66	AR Screen: Scanning Initiation	91
67	AR Screen: Ground Detected	92
68	AR Screen: Indicating Shoreline	92
69	AR Screen: AR Content on top of Real World	93
70	AR Screen: Calibration Menu	93
71	AR Screen: (a) Before (b) After	94
72	AR Screen: Current State (Close Distance)	95
73	AR Screen: 3.6m Retreat (Medium Distance)	95
74	AR Screen: 7.7m Retreat (Far Distance)	96
75	AR Screen: Info Signs	96
76	AR Screen: Signs, (a) Current Shoreline, (b) Retreat 3.6m, (c) Retreat 7.7m	97
77	AR Screen: Current Shoreline, Info Panel	97
78	AR Screen: Info Panels, (a) Retreat 3.6m, (b) Retreat 7.7m	98
79	Orientation: (a) Landscape (b) Portrait	100
80	Unity Version	101
81	Google Play Store: ARCore	103

82	Importing Packages in Unity [52]	104
83	Inserting Mapbox API key in Unity [52]	104
84	Essential Packages in Unity's Package Manager	105
85	LWRP Asset Assignment	106
86	LWRP Settings	107
87	Shader Graph	108
88	Imported Packages	109
89	Scenes of the Project	110
90	Structure of the Project	111
91	Welcome Scene Components in Unity Editor	111
92	Welcome Scene Manager Script	112
93	Toggles Handler Script	113
94	Start Button Script	113
95	Map Theme: Light (Up), Dark (Down)	114
96	Map Scene Components in Unity Editor	114
97	Location Provider	115
98	Target's Components	116
99	Map Components	116
100	Inserting POIs on Map (Left), Cone Prefab (Right)	117
101	Character Movement Component	117
102	Character Movement Script	118
103	Indicator Objects, Arrow (Left), Humanoid (Right)	118
104	Requesting Directions Script	119
105	Handling Directions Script	119
106	Direction Factory Component	119
107	Set to Location Component	120
108	Set to Location Script	121
109	Calculate Geo Distance Script	122
110	Get Distance Script	122
111	Enable Info/Alert Script	123
112	Enabling AR button Script	123
113	Load AR Scene Script	124
114	Scene Manager Components	124
115	AR Scene Components	125
116	AR Session Components	125
117	AR Session Origin Components	126
118	AR Camera Components	126
119	Custom Forward Renderer for AR	127
120	Our Water Shader on Shader Graph	128
121	Our Water Shader Public Variables	129
122	Essential Water Shader Properties on Shader Graph	129
123	Water Shader: (a) Alpha Blend, (b) Additive Blend	130
124	Water: (a) Default Unity Render, (b) Lightweight Render Pipeline	130
125	Designing 3D Content based on the Data	131
126	Layer Designing on 3D Content	132
127	Water Lerp Script	132
128	Check for Tackable Panes Script	134
129	Spawn Scene Script	134

130	Calibration Content Script . . . . .	135
131	Raycast Signs Script . . . . .	135
132	Toggle Trackables Script . . . . .	135

## List of Tables

1	Points of Shoreline - Current State [12]	69
2	Points of Shoreline - Retreat 3.6 meters inland [12]	70
3	Points of Shoreline - Retreat 7.7 meters inland [12]	71

# 1 Introduction

Littering behavior is a global issue affecting most countries, regardless of their development status. Mobile Augmented Reality (MAR) shows a promising contribution in different fields. However, despite the wider applications of MAR in different areas such as cultural heritage [14] and shopping [15], acceptance studies of mobile augmented reality applications with environmental awareness are still rare. In this thesis, the main purpose is to increase environmental awareness using Landscape Visualization in combination with Augmented Reality. [16]

Landscape visualization is often used for communicating complex information about the state of a landscape and how it might change. It can be particularly effective when communicating to community groups and policymakers. Representation of existing real world and potential alternatives is a powerful tool for public understanding. MAR can achieve that by giving to the public audience the ability to experience potential changes of the environment around them, as if they take place the exact same time, at the exact same place.

The phenomenon this thesis focuses on, is the erosion of coastal zone, and the changes in the coastline. Coastline is a physical line of the area where land meets sea. Nowadays, coastline extraction and tracking of its changes has become of high importance because of the climate change, global warming and rapid growth of human population. Coastal areas play a significant role for the economy of the entire region. Dramatic changes in coastline can result in disastrous outcomes for the region, therefore, the increase of public awareness is vital.

## 1.1 Brief Description

In this thesis, we present the development of a mobile application for coastal erosion visualization, using Augmented Reality. The location of our interest is Georgiupoli's Beach in Chania, Crete. The coastal erosion in the area is quite extensive and is expected to be increased even more on the following years.

The purpose of this thesis is to visualize the beach in its future state. The beach will be depicted in the near future from now, on the basis of mathematical models that show the tendency of the beach to erode. This prediction is based on the current state of the beach, and having as a given, that no further measures will be taken place (by human factor) to reduce the rate of erosion.

The final product, is an outdoors, on-site Augmented Reality application for mobile devices as mean of the visualization. Our implementation supports both Android and iOS devices. The user will be able to visit the area of interest and experience in real time the visualization with his/her smartphone device using our MAR application. The experience should give the user an understanding and clear view of the future changes that coastline will be undergone. Below in figure 1 an

example is showcased.



(a)



(b)

Figure 1: (a) Present (Real) (b) Future (Real + Virtual)

In more details, the application was developed in Unity3D game engine using its newly released package AR Foundation. AR Foundation simplifies the procedure of development on different mobile platforms such as Android and iOS. The application consists of two main phases. The first phase (non AR), offers a navigation in the area of Georgioupoli's Beach with useful instructions and options to the user. Once the user navigates himself/herself to one of the notified locations, the second phase takes place. In this phase the user can experience the visualization, immediately after a brief procedure that will be imposed. The locations of interest are three and the visualization shows two possible future scenarios of the coastline for sea level elevation by 0.5 meters and 1 meter, where coastline is estimated to penetrate 3.6 meters and 7.7 meters inland, respectively.

## 1.2 Thesis Structure

This thesis is organized to provide a continual narrative to be followed. There are seven chapters that are set out as follows.

After this Introduction, the 2nd chapter acts as a prologue and introduces the reader to the fields of Augmented Reality (AR), the history of AR, state of the art and the means of AR interaction, different AR applications that are well-known nowadays, the problems that faces AR technology at the moment and finally the progress and the future of Augmented Reality.

The 3rd chapter contains the explanation of the task, the case analysis and few words about the significance of coastal erosion and the possible threats of this phenomenon in the area of Georgioupoli. Last but not least, It includes the data that have been used in this thesis, and the forecast based on these.

In the 4th Chapter, requirement analysis takes place. We outline the requirements we set for our application before the development. The use cases scenarios are presented schematically. And finally, the hardware requirements for the smooth execution of the application.

In the 5th Chapter, there is a presentation of our application from the perspective of an End-user that executes the app. There is an introduction to the application showing the navigation structure between the different screens, followed by a showcase of the final screens which the user sees during the whole experience, step-by-step. Closing with some helpful tips and functionality directions.

The 6th Chapter includes the full Implementation process from the developers point of view and how every tool have been used and the part of each one in the structure of the application. There we analyze exactly what we did and describing the procedure of development in details.

Finally, in the 7th Chapter there is a summary of our thesis and our experience during this period of time. In addition, there is an evaluation of the final result and we list future improvements that can be made to solve some issues and improve the overall experience of the end user.

## 2 Augmented Reality (AR)

### 2.1 Introduction to AR

Augmented Reality (AR) allows the real time blending of the digital information processed by a computer, with information coming from the real world, by means of suitable computer interfaces. There is a clear difference between the concept of virtual reality and augmented reality that can be explained with the help of Paul Milgram and Fumio Kishino's Reality – Virtuality Continuum, as in figure 2 .

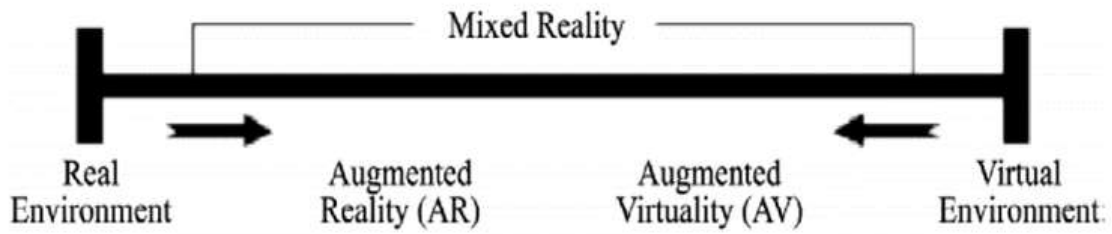


Figure 2: Milgram's Reality - Virtuality Continuum [1]

The real world and a totally virtual environment are at the two ends of this continuum. The middle region is called Mixed Reality. Augmented reality lies near the real-world end of the spectrum, with the predominate perception being the real world augmented by computer generated data. Augmented reality (AR) aims to enhance our view of the world by superimposing virtual objects on the real world in a way that persuades the viewer that the virtual object is part of the real environment.

AR system consists of three simple steps: Recognition, Tracking, and Mix. In recognition any image, object, face, a body or space is recognized on which virtual object will be superimposed. During tracking real-time localization in space of the image, object face, a body or space is performed and finally media in the form of video, 3D, 2D, text, etc. are superimposed over it, achieving an elegant way of mixing real and virtual world into one.



## 2.2 History of AR

While AR is most widely known for its modern applications, it has been around and experimented upon for approximately 20 years. Thus, various technologies have already been tested using a multitude of tools, especially when trying to align the virtual and real worlds. Older technologies consisted of Head-Mounted Displays (or HMDs), eyeglasses or contact lenses that showed virtual objects in front of the user's eyes. This posed a multitude of problems because the tolerance when tracking sudden movements of the user was low and the precision of the then available instruments could not match it, thus users experienced frequent nausea and disorientation.

Moving further into the future, AR applications moved from HMDs to handheld tablets and smartphones. As users were distanced from the virtual screens, the tolerance for accurate tracking rose making it simpler to test new ideas. In addition, with the rising popularity and demand for better smartphones and tablets, many complicated Head-Mounted sensors were integrated into smartphones and tablets, making them the ideal environment for developing new applications.

Mobile Augmented Reality has largely evolved over the last decade, as well as the interpretation itself of what is Mobile Augmented Reality. The first instance of Mobile AR can certainly be associated with the development of wearable AR, in a sense of experiencing AR during locomotion (mobile as a motion). With the transformation and miniaturization of physical devices and displays, the concept of mobile AR evolved towards the notion of "mobile device", aka AR on a mobile device.

Below, there is a showcase of some honorable mentioned AR systems and applications using different approaches through the years.

### *The First Augmented Reality System*

1968: Ivan Sutherland creates the first augmented reality system, which is also the first virtual reality system. It uses an optical see-through head-mounted display that is tracked by one of two different 6DOF trackers: a mechanical tracker and an ultrasonic tracker. Due to the limited processing power of computers at that time, only very simple wire-frame drawings could be displayed in real time. [5]



Figure 3: Ivan Sutherland's first Augmented Reality System [2]

#### *First Use of "Augmented Reality" as a Term*

1992: Tom Caudell and David Mizell coin the term "augmented reality" to refer to overlaying computer-presented material on top of the real world. Caudell and Mizell discuss the advantages of Augmented Reality versus Virtual Reality such as requiring less processing power since less pixels have to be rendered. They also acknowledge the increased registration requirements in order to align real and virtual. [5]

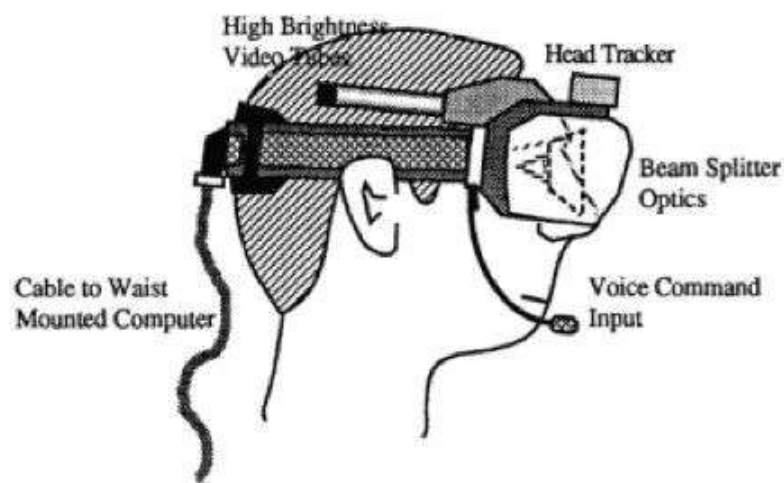


Figure 4: Caudell and Mizell coining AR in 1992 [3]

### *Reality-Virtuality Continuum*

1994: Paul Milgram and Fumio Kishino write their seminal paper "Taxonomy of Mixed Reality Visual Displays" [17] in which they define the Reality-Virtuality Continuum. Milgram and Kishino describe a continuum that spans from the real environment to the virtual environment (fig. 2). In between there are Augmented Reality, closer to the real environment and Augmented Virtuality, which is closer to the virtual environment.[5]

### *Definition of Augmented Reality*

1997: Ronald Azuma presents the first survey on Augmented Reality. In his publication [18], Azuma provides a widely acknowledged definition for AR, as identified by three characteristics:

- it combines real and virtual
- it is interactive in real time
- it is registered in 3D.

Today Milgram's Continuum and Azuma's definition are commonly accepted as defining Augmented Reality [5]

### *First Mobile AR System*

1997: Steve Feiner et al. present the Touring Machine (fig. 5), the first mobile augmented reality system (MARS) [4]. It uses a see-through head-worn display with integral orientation tracker; a backpack holding a computer, differential GPS, and digital radio for wireless web access; and a hand-held computer with stylus and touch-pad interface[5].

### *ARToolKit: Beginning of an Era*

1999: Hirokazu Kato and Mark Billinghurst present ARToolKit, a pose tracking library with six degrees of freedom, using square fiducials and a template-based approach for recognition [19]. ARToolKit uses a simpler marker-based approach, with additional support for Natural Feature Markers. ARToolKit tracks the markers while in view, calculating which markers are in view, their orientation relative to the camera as well as their depth from the camera. The most commonly used markers are 2D barcodes and 3D boxed pattern barcodes (fig. 6), which can often look out-of-place if they are not hidden correctly. ARToolKit is available as open source and is still very popular in the AR community.



Figure 5: Touring Machine by Feiner et al. [4]



Figure 6: ARToolKit for pose tracking in 6DOF [5]

### *ARQuake: First AR Game*

2000: Bruce Thomas et al. present AR-Quake, an extension to the popular desktop game Quake [6] (fig. 7). ARQuake is a first-person perspective application which is based on a 6DOF tracking system using GPS, a digital compass and visionbased tracking of fiducial markers. Users are equipped with a wearable computer system in a backpack, an HMD and a simple two-button input device. The game can be played in- or outdoors where the usual keyboard and mouse commands for movement and actions are performed by movements of the user in the real environment and using the simple input interface [5].



Figure 7: ARQuake by Thomas et al. [6]

### *Mozzies: AR Mobile Game*

2003: The Siemens SX1 is released, coming with the first commercial mobile phone AR camera game called Mozzies (also known as Mosquito Hunt) (fig. 8a). The mosquitoes are superimposed on the live video feed from the camera. Aiming is done by moving the phone around so that the cross hair points at the mosquitoes. Mozzies was awarded the title of best mobile game in 2003 [5].

### *The Invisible Train: Multiplayer AR Mobile Game*

2004: The Invisible Train, is shown at SIGGRAPH 2004 Emerging Technologies [5] (fig. 8b). The Invisible Train is the first real multi-user Augmented Reality application for handheld devices (PDAs). Unlike other projects, in which wearable devices were merely used as thin-clients, while powerful (PC-based) servers performed a majority of the computations (such as graphics rendering), the software



runs independently on off-the-shelf PDAs - eliminating the need for an expensive infrastructure [7].



(a)



(b)

Figure 8: (a) Siemens SX1 AR game "Mozzies".[5] (b) Multi-user AR application for handheld devices "The Invisible Train".[7]

#### *AR-Tennis: First Collaborative AR on Mobile Phone*

2005: Anders Henrysson ports ARToolKit to Symbian (fig. 9a). Based on this technology he presents the famous AR-Tennis game, the first collaborative AR application running on a mobile phone. ARTennis was awarded the Independent Mobile Gaming best game award for 2005, and the technical achievement award [5].

#### *Going out: Robust Model-based tracking for Outdoor AR*

2006: Reitmayr and Drummond present a model-based hybrid tracking system for outdoor augmented reality in urban environments enabling accurate, real-time overlays on a handheld device. The system combines an edge-based tracker for accurate localization, gyroscope measurements to deal with fast motions, measurements of gravity and magnetic field to avoid drift, and a back store of reference frames with online frame selection to re-initialize automatically after dynamic occlusions or failures [5].

#### *Wikitude World Browser*

2008: Mobilizy launches Wikitude, an application that combines GPS and compass data with Wikipedia entries. The Wikitude World Browser overlays information

on the real-time camera view of an Android smartphone[5]. In 2012, the company restructured its proposition by launching the Wikitude SDK, a development framework utilizing image recognition and tracking, and geolocation technologies. Wikitude SDK is well known for its capabilities until now in 2020.



Figure 9: (a) AR-Tennis by Henrysson et al.[8] (b) Wikitude AR Browser.[5]

#### *Google Glass: Mixed Reality gets Publicity*

2012: Google Glass (also known as Google Project Glass) is firstly presented to the public. Google Glass is an optical HMD that can be controlled with an integrated touch-sensitive sensor or natural language commands. After its public announcement Google Glass had a major impact on research but even more on the public perception of mixed reality technology[5].

### **2.3 Current state of AR**

As in early 2020, augmented reality is believed to have the biggest potential for mass consumption. The last few years have seen immense improvements in AR technologies. Not only are smartphones becoming equipped with high end sensors mandatory for AR, but also major companies have started prototyping new HMDs, commonly known nowadays as AR masks/glasses. AR masks tend to make standalone dedicated hardware specifically for AR, but unlike old HMDs, these will not be application specific but re-programmable AR hardware. On the other hand, there are multiple Software Development Kits, or SDKs for short, which can be used to develop smartphone applications.

Augmented reality is primarily experienced via a wearable glass device, head-mounted device, or through smartphone applications. Augmented reality overlays digital content on top of the real world. Therefore, AR enhances the user's experience in the real world rather than replacing it. In order this to be achieved, special

hardware is needed. For simplicity reasons, the hardware is separated into two main categories: 1) The head mounted equipment and 2) The handheld equipment. In the next two sections, some of the state-of-the-art equipment is listed and explained shortly.

### 2.3.1 AR Head Mounted Displays (HMDs)

A head-mounted display (HMD) is a display device, worn on the head or as part of a helmet, that has a small display optic in front of one (monocular HMD) or each eye (binocular HMD). An HMD has many uses including gaming, aviation, engineering, and medicine. There is also an optical head-mounted display (OHMD), which is a wearable display that can reflect projected images and allows a user to see through it. A typical HMD has one or two small displays, with lenses and semi-transparent mirrors embedded in eyeglasses (also termed data glasses), a visor, or a helmet. The display units are miniaturized and may include cathode ray tubes (CRT), liquid-crystal displays (LCDs), liquid crystal on silicon (LCos), or organic light-emitting diodes (OLED). Some vendors employ multiple micro-displays to increase total resolution and field of view.

HMDs differ in whether they can display only computer-generated imagery (CGI), or only live imagery from the physical world, or combination. Most HMDs can display only a computer-generated image, sometimes referred to as virtual image. Some HMDs can allow a CGI to be superimposed on real-world view. These are augmented reality (AR) or mixed reality (MR) HMDs, and on these we shall focus on.

AR HMDs are the epicenter of modern AR research. Using new technologies borrowed from the nowadays popular Virtual Reality masks and combining them with spatial scanning technologies such as Microsoft's Kinect promises to create a new standard for AR research. These new masks promise to have fully 3D visuals for all necessities, from industrial and academic usage to integrating standard computer functionality into a mask. As many major companies, such as Microsoft, are investing into developing their own AR HMD, this medium promises to be as big an evolution in technology as smartphones were 15 years ago. Although the new era of AR masks started back in 2016, up until 2018 their production and shipping were very limited. Still, AR masks are still a prototype idea starting to slowly take form. Major companies are competing to design the optimal User Environment, usually with completely different approaches into both the hardware as well as the software of these devices. As such, it is a new technology that still requires years of optimization and improvement until it is widely known and accepted. Below, there are a few AR HMDs/Glasses that is believed to be the best in the market at this moment and will have major influence in the years to come.

#### *Microsoft HoloLens*

Microsoft HoloLens: developed by Microsoft back in 2016 (fig. 10). One of the first AR HMDs to be announced and sell their prototypes, although in limited



regions. Backed up by Microsoft's name, Kinect's tracking technology and an ambition to fully integrate Windows in an AR environment, this mask has set very high expectations both for itself and competitors.



Figure 10: Microsoft HoloLens HMD [37]

When it was first announced, HoloLens not only promised to integrate traditional computer graphics in AR, but also full scale holograms, for example human holograms, that could move naturally or even recreate scenes like a full-scale soccer match from a recording.

Packed with the processing power of an average laptop and a multitude of sensors HoloLens aims for precision tracking and world scanning around the user. On the software side, it uses an optimized version of the already trained and tested Microsoft Kinect's Neural Network for tracking and environmental scanning. Microsoft highlights numerous industries for which the HoloLens is suitable, from manufacturing and retail to healthcare and education. This self-contained, holographic computer allows for increased productivity, collaboration, and 3D design. (fig. 11/13)



Figure 11: Windows 10 on Microsoft HoloLens [38]

### *Microsoft HoloLens 2*

Microsoft HoloLens 2 is the successor to the pioneering Microsoft HoloLens (fig. 12). On early 2019 the HoloLens 2 enterprise edition debuted as the first variant of the device, followed by a developer edition that was announced on May 2, 2019. It was subsequently released in limited numbers on November 7, 2019. As for now, the shipping is limited and only to authorised customers.



Figure 12: Microsoft HoloLens 2 HMD [37]

Some of the main improvements with the HoloLens 2 headset include: 1) Processing power: the HoloLens 2 is more powerful than its predecessor. 2) Field of View (FOV): at 52 degrees, the HoloLens 2's field of view is larger, offering a more immersive AR experience for the user. The original HoloLens only has an FOV of 30 degrees. 3) Battery life: the HoloLens 2 features a 3-hour battery life, while the HoloLens 1 has a 2.5-hour battery life. 4) Design and fit: according to Microsoft, the HoloLens 2 offers a lighter and more ergonomic fit than the original HoloLens. The HMD also now features a flip-up visor which allows users to enter/exit augmented reality more quickly.



Figure 13: Microsoft HoloLens 2 on Healthcare [37]

### *Magic Leap One*

The Magic Leap One is a popular standalone AR headset made by Magic Leap 14), Magic Leap One uses Machine vision to thoroughly scan the environment around the user and make virtual objects context sensitive to the world around them. In addition, virtual objects are not only visually immersive but also use spatial audio with increasing depending on the distance from virtual objects.



Figure 14: Magic Leap One HMD [39]

More features of this AR headset include advanced eye-tracking. Therefore, the MR device knows where the user is looking and tracks their gaze, with even blinks serving as a command function for the user. Furthermore, user input methods include head posing, gesture controls (fig. 15) and voice commands.

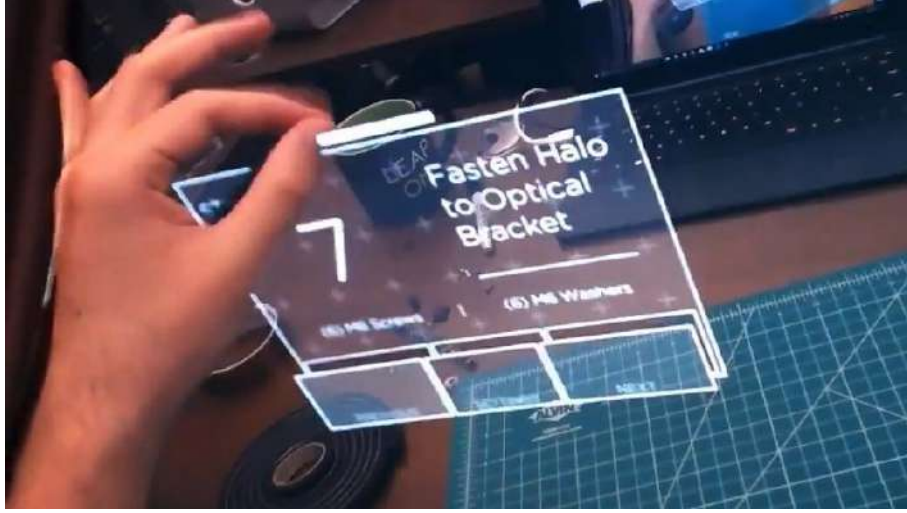


Figure 15: Magic Leap One Gesture Recognition [40]

Magic Leap's hardware consists of more than just the mask. The mask consists of the glasses and stereo headphones the user wears, but all the processing power resides in a tethered Lightpack (fig. 16a), which is a trademarked high-powered processing and graphics portable mini PC. It also comes bundled with a controller with 6-DoF (Degrees of freedom) of movement called Magic Leap Control (fig. 16b).



(a)



(b)

Figure 16: (a) Magic Leap Lightpack (b) Magic Leap Control [39]

## *Nreal Light*

The Nreal Light augmented reality glasses, publicly announced in early 2019 by Nreal, a tech startup based in China. Nreal Light works tethered to a mini PC (like the Magic Leap One) (fig. 17a). Light and simple, they almost look like “normal” sunglasses or eyeglasses, unlike most other AR products on the market that tend to be bulky (fig. 17b).



Figure 17: (a) Nreal Light Glasses, Controller and mini PC (b) Nreal Light Glasses: look like sunglasses [41]

The Light AR wearable features Simultaneous Localization and Mapping (SLAM) algorithms which allows for inside-out tracking, in addition to two on-board cameras which provide for a seamless AR experience. In addition, users also are provided with a controller (3DoF) called Oreo which offers haptic feedback. These AR glasses must be tethered to an external device, Nreal Light’s proprietary tethered external computing pack (Android-powered). The pack provides about 3 hours of usage. Users can also replace the lenses for prescription lenses. Furthermore, adjustable pads are available to allow for a more comfortable fit.

Other main features Nreal Light offers are: 1) Wide FOV: for an augmented reality device, the 53 degrees FOV is considered wide, thus providing users with a more engaging AR experience. 2) Voice-activated: users can give voice commands and utilize spatial audio with the Light. 3) Built-in speakers with spatial sound: 3D audio enables users to detect where sounds come from, offering a true AR experience. Users can also opt for Bluetooth speakers.



### 2.3.2 Mobile Augmented Reality (MAR)

Mobile Augmented Reality (MAR) systems provide the same services as augmented reality systems without constraining the individual's whereabouts to a specially equipped area. Mobile augmented reality is one of the fastest growing research areas in the augmented reality area, due to the emergence and widespread uptake of smart-phones that provide powerful platforms for supporting augmented reality on a mobile device.



Figure 18: AR Training in the Workplace [35]

Smartphones already have required technologies for hosting MAR applications, such as mobile processing, image recognition, object tracking, display technology and GPS location. They provide a suitable environment for hosting MAR application without paying any extra cost for special hardware. Many MAR applications were developed for the users of popular mobile operating systems: Android and IOS. The available MAR applications in both Play store and App store are developed based on two main approaches: MAR browsers based on geo referenced positioning and image-recognition-based MAR [20]. Mobile augmented reality technology can raise environmental awareness and help learning about the surrounding context. Chou and Chan Lin [21] state that AR can raise awareness of users' surroundings more than what they would gain by traditional methods such as radios, maps, and hand-held displays (fig. 19). Additionally, the user's learning interest can be improved by augmented reality achieving more well-educated audience.



Figure 19: AR Navigation using Smartphone [36]

AR apps utilize the sensors and cameras already present in modern computers or, more commonly, smartphones in order to gather information from the real world and allow virtual graphics to blend into the natural environment seen through a camera. In order to develop an AR application, developers frequently use a pre-built Software Development Kit, or SDK for short, which provides them with premade tools useful for AR. These tools vary from automating simple jobs, like setting up a new AR scene, to complicated algorithms like using Machine Vision to scan the environment and extract data like marker detection.

Augmented Reality SDKs facilitates many components within the AR application: AR recognition, AR tracking and AR content rendering. The recognition component works as the brain of the AR app. The tracking component can be stated as the eyes of the AR experience, and the content rendering is simply imaginative virtual objects and scenes on the real time information. An array of tools is provided to developers through SDK, required to recognize, track and render AR application in the most efficient manner. Augmented Reality SDKs can be organized in these broad categories: Geo-located AR Browsers, Marker based, Natural Feature Tracking. AR Browser SDKs allows users to create geo-located augmented reality applications, using the GPS and IMU available on today's mobile and wearable devices. Marker based SDKs employ special images, markers, to create augmented reality experiences. Natural Feature Tracking SDKs rely on the features that are actually present in the environment to perform the augmentation by tracking planar images, based on a SLAM (Simultaneous Location and Mapping) approach[22]. Some AR SDKs that seem promising and currently lead the way on mobile AR development are listed in subsection 2.6.1.

In this thesis, we follow the mobile AR approach, since the only equipment we shall need is a smartphone device. Also, HMDs are not so popular to the average user so we cannot expect that everyone owns an HMD. A smartphone device is enough to bring the visualization we desire with ease.

## 2.4 Use of AR Nowadays

While writing this thesis, AR has become quite popular to the public. More and more people understand the technology of AR and big brand names make use of this technology in a variety of applications. From gaming to advertising and from navigation to education, AR has to offer a wide range of options. AR has been used successfully in many cases, bringing to life experiences that, otherwise, would be unimaginable. Below, there is a small presentation of AR application that made the public audience feel the presence of AR technology, and brought new opportunities to AR developers.

### 2.4.1 Modern AR Apps

- **IKEA Place:** is an AR app for smartphones that's focused on home decor. With this augmented reality app The Swedish furniture retailer gives shoppers a chance to place virtual furniture from the IKEA catalogue directly to their room. A whole new interior design experience (fig. 20).

This app looks at the bigger picture, taking into account your home's entire floor plan to see which items will fit best where. Easy drag-and-drop functionality and the option to see different colors.



Figure 20: IKEA Place: Virtual Couch in Living Room



- *Toyota Hybrid AR*: The Toyota Hybrid AR application uses Augmented Reality technology and object recognition software to overlay graphics of the inner workings of the Hybrid drivetrain onto physical vehicles, helping customers to gain a better understanding of how the system works.

The 3D experience also features a number of ‘hotspots’ which, when clicked on, enable customers to get in-depth information on key features of the system, such as the motor, battery and fuel tank (fig. 21).

Brandwidth worked with Toyota to develop the Hybrid AR app as part of an ongoing drive to educate the brand’s customers on hybrid technology and the benefits it offers. The app currently supports the Toyota C-HR model, across all grades and colours.



Figure 21: Toyota Hybrid AR: In-depth Information

- *Google Translate*: Google’s Translate app is one of the most useful applications of augmented reality technology so far. It can translate text in an image from one language to another, allowing you to read signs, packaging, and even memes in other languages. Just open the app, capture the text, and wait for the translation (fig. 22). It also includes more pedestrian translation tools, which are the best free tools available.



Figure 22: Google Translate: AR Sign Translation

- *Night Sky/Star Walk*: Night Sky on iOS and Star Walk on Android both offer a compelling AR experience for stargazing and astronomy. These apps use the user's current location and phone's orientation to display a geographically accurate star map on the screen. In either app, this map can be layered on top of your surroundings through the augmented reality functionality (fig. 23). It's an awesome tool for learning about the astronomical, which is both entertaining and educational.



Figure 23: Night Sky: AR View of Virgo

- *Unbelievable Bus Shelter*: Pepsi decided to bring some surprises to a London bus shelter to make waiting a bit more interesting. For its ad campaign, Pepsi Max used augmented reality to turn a bus shelter's wall into a fake window that appeared to show flying saucers, an attacking robot, and a loose tiger — among other unlikely subjects — making their way down the street. Pepsi doesn't say exactly what it used to make the illusion happen, but it appears to have relied primarily on a camera outside the shelter that let it capture people and vehicles on the street.



Figure 24: Unbelievable Bus Shelter: Tentacle Illusion

#### 2.4.2 AR For Environmental Purpose

Despite the fact that AR has become quite popular, applications that focus on environmental understanding and education are, still, very rare or not so well-known. In this thesis, the increase of environmental awareness is of major importance. AR is believed and is able to bring a new way of eco-education, especially on younger ages. AR has more advantages compared to the traditional teaching methods. One of these advantages is that it activates many senses such as touch, hearing, and vision at the same time. Additionally, AR allows access to learning content in three-dimensional perspectives. 3D offers the possibility of ubiquitous learning and makes learners more cooperative. It gives users a sense of presence and immediacy with the object of exploration. It does something that is invisible to be visible. Below, there are some AR applications and studies, that use the AR technology for environmental purposes.

- *Augmented Reality as a mean of communicating environmental issues and boosting environmental education for school students* [9]: This paper deals with the implementation of augmented reality technology as a means of communicating environmental issues and boosting environmental education for 241 school students in the 4th, 5th, 6th classes of two primary schools in Athens, during the course of Computer Science. Specifically, an early version of two augmented reality applications for android mobile devices were designed and deployed. Two activities combining this technology were designed in order to address environmental learning goals concerning climate change concepts and fundamental aid in the understanding of renewable energy resources. The study assessed whether the students liked the applications and the rate of knowledge change, driven by pre-post questionnaires, which were given both at the start and at the end of the implementation. The results showed that the implementation of Augmented Reality applications for environmental educational concepts have a significant supplemental learning effect as a mobile-assisted learning tool. Finally, the paper concludes with future guidelines in the field of other environmental issues of great importance.

The focus of the script was the students. Students are divided into groups and learn to use AR on their own. Students should explore and achieve results through team effort not being benefited from the presence of the teacher in the classroom. The applications had a duration of about two minutes each.

For the first application were used two image targets (fig. 25). The first image target presents the renewable energy sources, such as solar power, geothermal power and wind power, while the second presents the fossil fuels (coal, oil and natural gas). The result is shown in Figure 27 (Left).

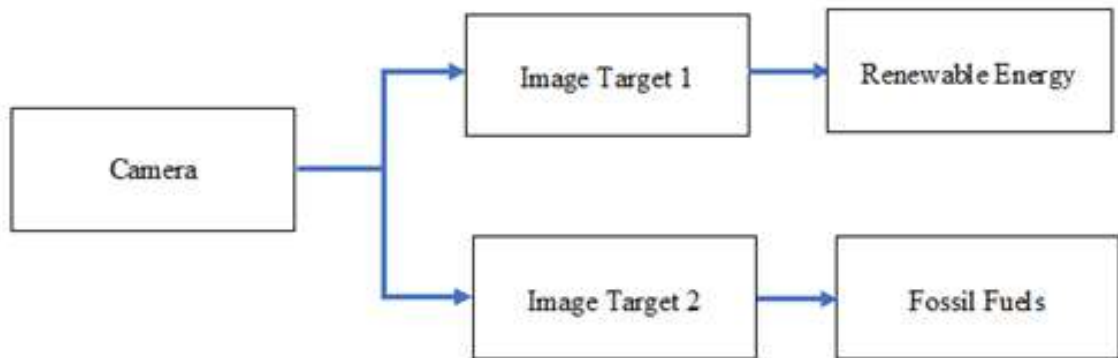


Figure 25: Energy Source application workflow [9]

For the second application were used only one image target (fig. 26). The second application deals with the phenomenon of climate change. At first, it's presented the causes of temperature increase, such as deforestation and burning fossil fuels that increase CO2 emissions, then it's shown the impact of the above, such as ice melting, sea level rise and desertification. Finally, there is a proposition of various practices to reduce the progress of climate change such as recycling, energy saving

and reduce car usage. A screenshot is shown in Figure 27 (Right).

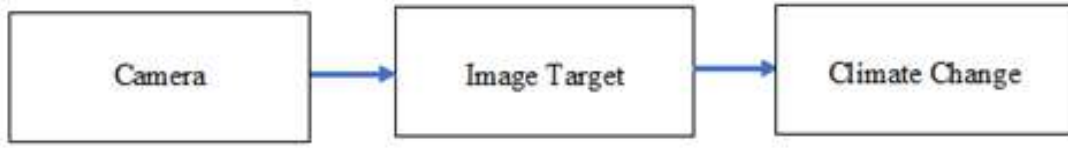


Figure 26: Climate Change application workflow [9]

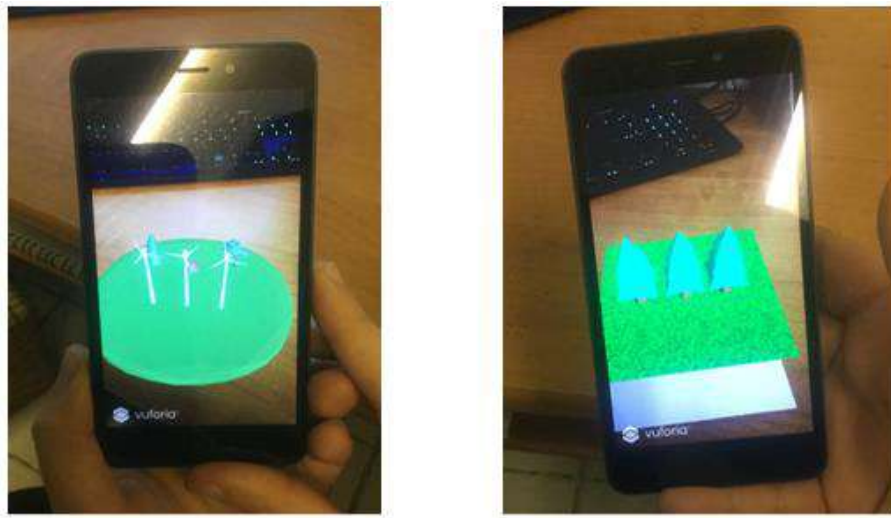


Figure 27: (Left) Wind Power, (Right) Climate Deforestation [9]

• *Animating eco-education: To see, feel, and discover in an augmented reality-based experiential learning environment* [10]: the current study develops an eco-discovery AR-based learning model (EDALM) (fig. 28) which is implemented in an eco-discovery AR-based learning system (EDALS). In a field experiment at a botanical garden, 21 middle school students constitute three groups participated in a learning activity using different learning types and media. Quantitative results indicate that, compared to the human-guidance-only model, EDALS successfully stimulates positive emotions and improved learning outcomes among learners. In post-activity interviews, students indicated they found the exploration mode provided by the proposed system to be more interesting and helpful to their learning in school. The use of attractive technologies increase students' willingness not only to learn more about the environment, but also to develop a more positive emotional attachment to it.

To understand the impact of the system on emotion and learning effect in the ecological education context, this study conducts three post-activity assessments, and conducted a whole day learning experiment at the National Museum of Natural Science-Botanical Garden, Taichung, Taiwan. The subjects were 21 middle school



students randomly assigned to three groups (7 in each group) with approximately equal numbers of boys and girls in each group. The three groups were: (1) Group A used the AR system for self-learning; (2) Group B used AR plus commentator, where learning is accomplished via AR system instruction and commentator guidance; and (3) Group C was a control group using traditional experiential learning, in which students followed the commentator through the gardens and listened to commentator guidance

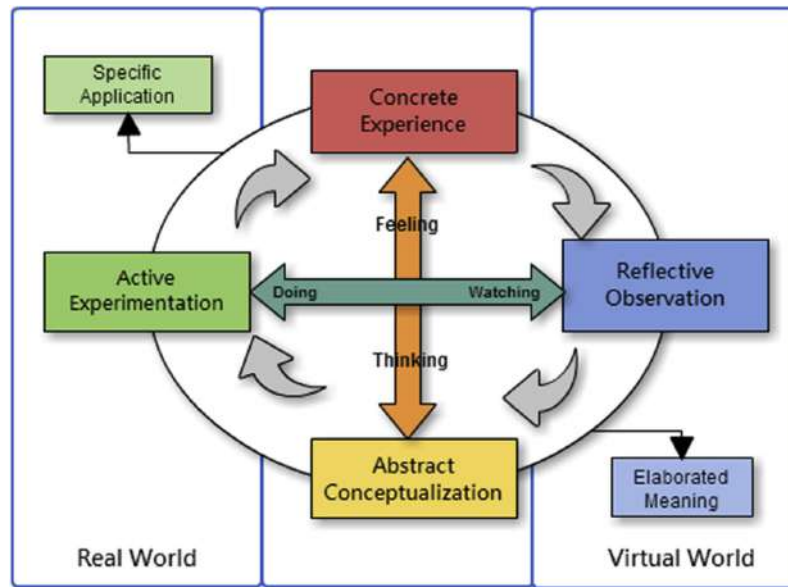


Figure 28: Eco-discovery AR-based learning model (EDALM) [10]

To avoid interference from situational factors, all three groups followed an identical learning route in the garden: 1. Taitung Cycads - 2. Monsoon Rain Forest - 3. Central Lowlands - 4. Tropical Rainforest Greenhouse. The groups all toured the garden on the same day, but with staggered starting times to ensure no overlap.

Specifically, experimental procedures involved the three groups first taking an indoor 15-min paper-based pre-test to assess their basic knowledge of and emotional attachment to plants. All participants were required to complete a series of experiential tasks, including indicating their pre-task emotional state, completing the learning tasks, indicating their post-task emotional state, and answering questions. Group C finished all tasks using pencil and paper (without EDALM), while each member of groups A and B were provided with a tablet computer with which to use EDALM. To ensure system familiarity, groups A and B received additional guidance on EDALS on their tablets before entering the garden. After indicating their pre-task feelings, they read learning themes related to the eco-area. The AR feature in the operation system presents an interactive virtual plant silhouette (fig. 29a) and subjects were prompted to learn about related plants (fig. 29b), guiding them to experience and explore the surrounding environment. Subjects were then prompted to indicate their post-task emotions, and answer 4 to 6 achievement assessment questions to complete the learning theme before moving on to the next theme. After all

four learning themes are complete, the subjects were asked to complete a questionnaire on the experiential learning activity. Six subjects were randomly selected for interviews to learn more about their learning conditions using the AR system.



Figure 29: (a) Screenshot of an interactive virtual plant silhouette (b) Learning content about specific plants [10]

- *Mobile Augmented Reality for Flood Visualisation* [11]: Presentation of a real time immersive prototype Mobile Augmented Reality (MAR) app for on site content authoring and flood visualisation combining available technologies to reduce implementation complexity. Networked access to live sensor readings provides rich real time annotations. The main goal was to develop a novel MAR app to complement existing flood risk management (FRM) tools and to understand how it is judged by water experts. Going beyond the presented work, the flexibility of the app permits a broad range of applications in planning, design and environmental management.



Figure 30: Browser/Authoring tool. AR tracking showing side menu, interaction annulus, and two triangulated points [11]

The app is formed of three distinct activities for (i) main menu, (ii) project in-

formation and options, and (iii) authoring and browser (fig. 30). The former two enable the user to create new projects, find, select and view existing project information and options, whereas the latter activity is where authoring and/or browsing (i.e. visualisation) occurs. Return to previous activities is achieved by pressing the device back button. Authoring/browser activity interaction occurs via a retractable side menu. A typical authoring use case would see the user select the triangulate menu option to triangulate a point by focusing a central annulus on a desired point and tapping the touch screen three times from three different viewpoints, repeating this process to triangulate further points. Then, selecting to add geometry from the menu allows the user to attach, or “hang”, geometry to these triangulated points. Model parameters may be adjusted via the menu to adapt the model to the existing natural features. Additionally, textual annotations may be attached to triangulated points, such as sensor readings, which appear as spinning information cubes, to be selected during browsing (fig. 31). Lastly, a flood plane may be turned on, and the building geometry turned off, revealing a flood plain obstructed by the invisible building geometry (fig. 32). This flood plane may be moved up and down via the touch screen and low/high water levels set. As the user moves the flood plane up and down these flood level extremities are automatically interpolated to give the user a feel for flood depth.

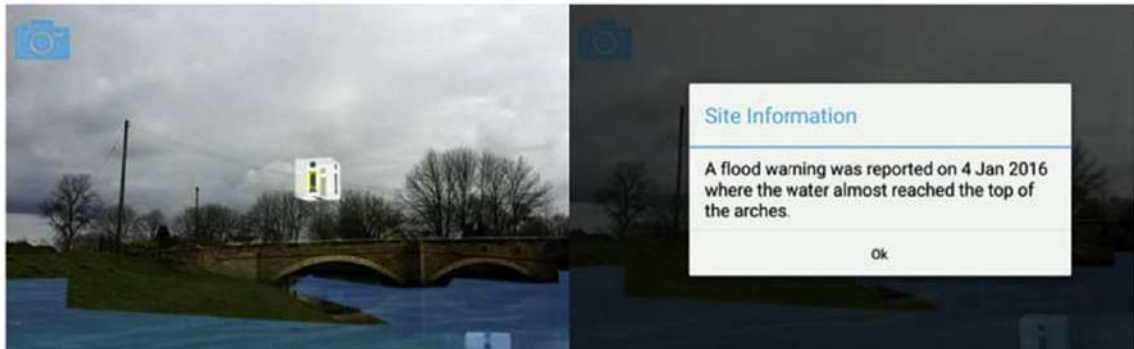


Figure 31: Example point annotations [11]



Figure 32: Flooding with building geometry on (Left), and geometry off (Right) [11]



## 2.5 The Registration Problem

Registration in an AR system is the degree in which the virtual information is accurately presented with the real environment. The objects in the real and virtual worlds need to be properly aligned with respect to each other, or the illusion that the two co-exist will be compromised (Azuma 1997). In contrast to Virtual Reality (VR) where such errors result in visual-kinesthetic conflicts, in AR such conflicts are visual and easier to detect. Take for example a user wearing a VR headset that raises his/her arm to see a virtual one. If the virtual arm is off by a few centimeters, it may not be detected because the conflict is between the “sensed” position of the real arm, and the “seen” position of the virtual one. In the corresponding AR application the virtual arm should completely overlap the real one, so such an error would be easily detectable.

Registration errors are divided to dynamic and static. Static errors are the ones that affect the AR scene even when both user and environment are in stasis. Sources of such errors can be bad calibration of mechanical parts, incorrect tracker-to-eye ratio, field-of-view parameters, optical lens distortions, etc. Static errors depend mostly on mechanical parts and the correct initial calibration of the system and can be accounted for to a very satisfying degree. Dynamic errors on the other hand are the ones that take effect if either the viewpoint (user) or the annotated object, begin moving. For MAR this kind of errors are by far the largest contributors to the Registration problem and vary depending on the implementation.

In early AR systems the single most important factor for dynamic errors was end-to-end system delays. A tracker reported user movements to the system and the system should then update the digital artifacts on the screen. This computation and its delivery should precede changes in the user’s pose which proved at the time to be a very difficult task. With today’s hardware, system delays have been minimized and the main source for errors in registration is Pose estimation (Position and Orientation tracking). Tracking in an AR scene has proven a complicated task with no single best solution.

In order to register virtual content in the real world, the pose (position and orientation) of the viewer with respect to some “anchor” in the real world must be determined. Depending on the application and technologies used, the real world anchor may be a physical object such as a magnetic tracker source or paper image marker, or may be a defined location in space, determined using GPS or dead-reckoning from inertial tracking.

In order for an AR system to overlay the world with digital information it needs to track its position with 6 DoF (Degrees of Freedom). That means three variables for position and three for orientation. There are many factors that have enabled modern smartphone’s to track their position. Inertial sensors, GPS positioning and optical sensors can provide all the necessary data for such computations. Although there are many approaches to pose estimation for AR systems, we will focus on implementations for consumer grade smartphones. Each tracking approach has its advantages depending on the use case. The most popular of which will be explained

below. These include Marker-based Tracking, Natural Feature Tracking, Sensor-based Tracking, Model Based Tracking and the newer Markerless AR.

### 2.5.1 Fiducial Marker Based Tracking

It is the most often used technique to achieve AR. In AR markers are used for easy recognition in the field of view and typically have high contrast. By using them we can not only relate to point in space, but also calculate distance and the angle which we are looking at. Typical markers used in AR are black and white squares with geometric figures. Using of black and white gives high contrast compared to background environment and can thus be quickly recognized. One of the obvious downfalls in fiducial markers technology is that they always have to be visible and cannot be obscured by other objects during the augmentation. This problem can be partially alleviated by remembering marker position and refreshing accordingly its position with device movement [23].



Figure 33: Augmented Reality on Magazine [42]

In modern AR SDKs (that mentioned before), the role of a marker can be taken by any (almost) picture (fig. 33). The picture should have a pattern with high-contrasted elements and be clearly visible. This method is optimal for AR books, newspapers and magazines, bringing to life any image or information you might see or read on a piece of paper.

### 2.5.2 Natural Feature Tracking

This technique for achieving AR allows using objects in real world as markers by recognizing their natural characteristics. These characteristics act as “interesting features” of the image. Such characteristics are edges, flat surfaces, patterns and other highly distinguishable features. Feature descriptor of given image is saved for further recognition. Based on this feature set, the recognition of the same image

can be achieved from different distances, orientation and illumination levels, even with some occlusion, as the descriptor is invariant to those changes [23].



Figure 34: Natural Feature Tracking

Natural feature tracking is currently the most commonly used tracking approach as it removes the intrusiveness of the AR markers and it allows for robust tracking and registration. However this approach relies heavily on the features present in the real world scene and the ability to identify them. This means that a little number of features as well as occlusions and diverse lighting conditions in a scene, can greatly diminish the system's tracking capability.

### 2.5.3 Model Based Tracking

A model based approach uses prior knowledge of 3D objects in the environment along with their appearance. Using geometrical representation of 3D objects we can manipulate their position and orientation, matching them to their counterparts in the field of view. Model approach works using edge detection for construction of 3D models. In some cases the model is provided to track resemblance in relation to its object in the environment, e.g. tracking a moving car on a street. On the downside, this approach usually requires much more processing power[23].

Although not as popular as the Fiducial and Natural feature tracking these tracking techniques use known 3D structures, like a CAD model (fig. 35), to track real world objects. Edge filters are used to extract structure information about the scene which is then matched to primitive structure types, like lines, cubes, cylinders and circles, to provide pose estimations. Combining these tracking techniques with natural feature detection allowed for the inclusion of textures in the models which provided greater robustness in complex and variable environments.



Figure 35: Model Based Tracking of industrial objects by using CAD models [43]

#### 2.5.4 Markerless Tracking

With the rapid development of new technologies when it comes to machine vision, Markerless scene tracking has become possible in real time. With high resolution, high framerate cameras becoming widely available and cheap, we can extract highly detailed information about the surrounding environment, analyze the structure of the world and update virtual objects to blend in, all in realtime.

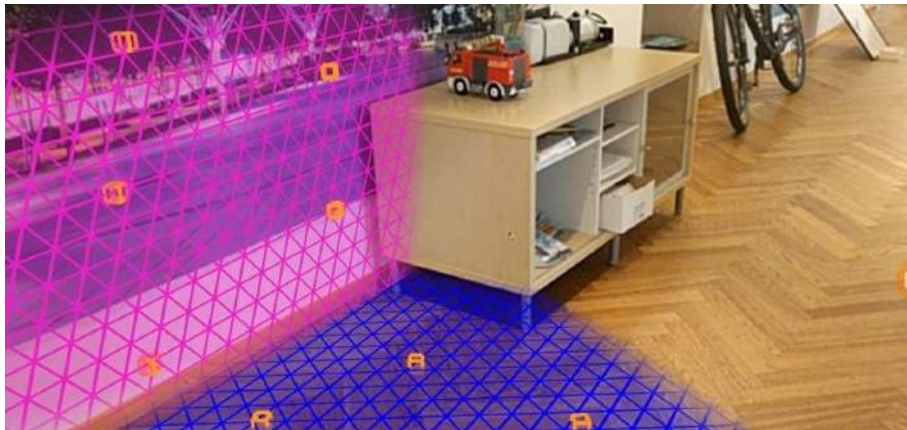


Figure 36: Plane Detection of Horizontal (blue) and Vertical (pink) Surfaces [44]

Extracting structure information about the real world has lead AR to adopt the well-known in robotics SLAM concept (Simultaneous Localization and Mapping) which allows for simultaneously create and update a map of the real environment while localizing the system's position within it. These approaches elevated

the AR system's capabilities from tracking planar surfaces (fig. 36) to more complex geometries and 3D structure. The motivation for SLAM together with further optimizations for AR has led to a process named PTAM (Parallel Tracking and Mapping) where the tracking of the camera and mapping of the environment components were separated which improved the overall performance. Furthermore, some newer smartphones as well as new HMDs further enhance markerless detection using a collection of cameras. By using multiple cameras or Infrared sensors we can also detect the depth of objects relative to the user allowing for better precision.

### 2.5.5 Sensor Based Tracking

Inertial tracking uses long range sensors like accelerometers, magnetometers and gyroscopes to calculate orientation, and combined with positions acquired from the GPS the system can calculate its pose relative to the Earth's frame. Inertial sensors allow for orientation tracking with 3-degrees of freedom by using a 6-axis accelerometer for orientation relative to the center of the Earth, and a magnetometer for measurements relative to the North. Gyroscopes can then be employed to detect changes between relative movements. By combining this information Easting can be calculated and 3 DOF orientation estimate is available (fig. 37).

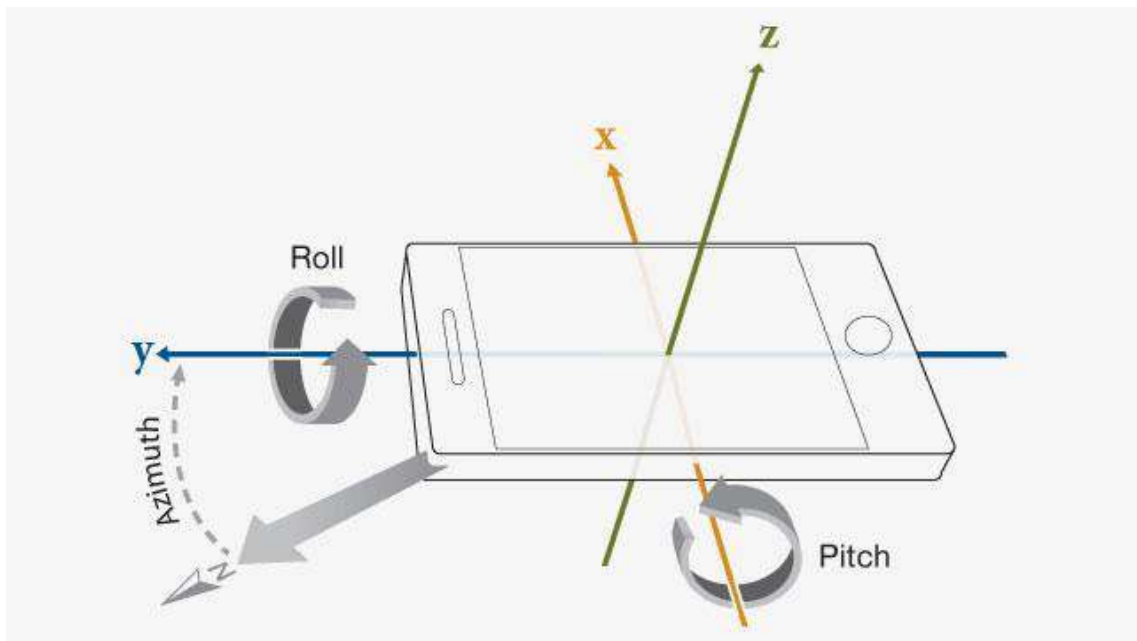


Figure 37: Mobile coordinate system and orientation relative to the Earth's frame of reference

Although position tracking is available solely with the use of the same sensors, they are very susceptible to drift over time, especially for such estimates which can only be derived from velocity. Due to this issue positioning with inertial sensors

should only be conducted with additional trackers capable of providing measurements for drift correction. For this reason sensor based systems employ other methods for position tracking. The GPS sensor provides position tracking in outdoor environments with an average accuracy of 3 meters. Locations acquired by the GPS include latitude, longitude and altitude (where available) information in the world coordinate system. Combined with the orientation estimate from the inertial sensors, 6 DOF tracking is possible relative to the Earth's Frame.

### 2.5.6 Hybrid Tracking

Hybrid tracking systems fuse data from multiple sensors to add additional degrees of freedom, enhance the accuracy of the individual sensors, or overcome weaknesses of certain tracking techniques. As mentioned above, sensor based implementations are susceptible to drift and latency due to filtering, while optical implementations have line of sight requirements and short range. As all the above registration methods are not mutually exclusive and detect different things, most developers tend to use multiple at once. Usually, Sensor-based tracking is combined with visual detection, as having information about the users' position and movement can be used to improve precision. Sensor-based tracking is also much faster relative to other registration methods, so using it is more beneficial than its computational cost.

In addition, visual registration methods are also combined. Since fully immersive methods are usually computationally expensive, they are also combined with Natural Feature Tracking and Marker-based AR to reduce the computational cost, or to improve accuracy by detecting key points. This allowed the systems to take advantage of the low jitter and drift of the optical approaches while extending the range of the AR system through the inertial sensors that have no line of sight requirements and high update rates that ensure responsive graphical updates.

Most commonly markerless tracking is combined with inertial sensor tracking. Motion estimates of the system are calculated from the inertial sensor, fusing data from accelerometers and gyroscopes, while the optical approaches provide measurements for drift correction and map the real world scene. Combining such techniques with additional depth information has opened up new possibilities for AR.



## 2.6 Developing Platforms & Software

A major part in the AR application development, is choosing the right tools based on the type of the Mobile AR application. After the requirements analysis we had about our MAR application, we had to find the right platforms and software to support our idea in the development phase. In this section, we give a brief description of the available AR SDKs that seemed to fit in our requirements and the game engines that act as AR development platforms nowadays. Closing, we discuss about the platform of our choice and the technical hardware requirements a device should have, in order to execute successfully the application.

### 2.6.1 AR Software Development Kits (SDKs)

While there exist plenty of AR SDKs they usually each have a specific focus, and it is quite frequent for companies to stop supporting and killing dated SDKs and newer companies publishing brand new SDKs. Luckily, there are still a few older SDKs still in existence, even with less support from their developers like Vuforia and Wikitude. Finally, the newest SDKs available are ARCore and ARKit that unlike previous ones are supported by Google and Apple directly. Below we will analyze these aforementioned SDKs. There are many more AR SDKs, providing similar capabilities with the ones mentioned before. More focus has been given to those four, as they seem to dominate the AR development scene and provide great documentation and support.



- *ARCore* is Google's platform for building augmented reality experiences [28]. Using different APIs, ARCore enables your phone to sense its environment, understand the world and interact with information. Some of the APIs are available across Android and iOS to enable shared AR experiences.

ARCore uses three key capabilities to integrate virtual content with the real world as seen through your phone's camera:

- 1) **Motion tracking** allows the phone to understand and track its position relative to the world. As your phone moves through the world, ARCore uses a process called concurrent odometry and mapping, or COM, to understand where the phone is relative to the world around it. ARCore detects visually distinct features in the captured camera image called feature points and uses these points to compute its



change in location. The visual information is combined with inertial measurements from the device's IMU to estimate the pose (position and orientation) of the camera relative to the world over time.

2) **Environmental understanding** allows the phone to detect the size and location of all type of surfaces: horizontal, vertical and angled surfaces like the ground, a coffee table or walls. ARCore looks for clusters of feature points that appear to lie on common horizontal or vertical surfaces, like tables or walls, and makes these surfaces available to your app as planes. ARCore can also determine each plane's boundary and make that information available to your app. You can use this information to place virtual objects resting on flat surfaces.

3) **Light estimation** allows the phone to estimate the environment's current lighting conditions. ARCore can detect information about the lighting of its environment and provide you with the average intensity and color correction of a given camera image. This information lets you light your virtual objects under the same conditions as the environment around them, increasing the sense of realism.

Fundamentally, ARCore is doing two things: tracking the position of the mobile device as it moves, and building its own understanding of the real world. ARCore's motion tracking technology uses the phone's camera to identify interesting points, called features, and tracks how those points move over time. With a combination of the movement of these points and readings from the phone's inertial sensors, ARCore determines both the position and orientation of the phone as it moves through space. In addition to identifying key points, ARCore can detect flat surfaces, like a table or the floor, and can also estimate the average lighting in the area around it. These capabilities combine to enable ARCore to build its own understanding of the world around it. Google recently announced Depth API, which allows developers to use depth-from-motion algorithms to create a depth map using a single RGB camera (fig. 38).



Figure 38: ARCore Depth API, Occlusion off (left) and Occlusion On (right) [26]

ARCore's understanding of the real world lets you place objects, annotations, or other information in a way that integrates seamlessly with the real world. Motion tracking means that you can move around and view these objects from any angle, and even if you turn around and leave the room, when you come back, the objects or annotation will be right where you left it.

ARCore supports developing in native platforms such as Android Studio (Android) and Xcode (iOS) and third party integration with Unity 3D and Unreal 4 game engines.

-ARCore Supported devices are listed in [\[49\]](#)



- *ARKit* is Apple's augmented reality (AR) development platform for iOS mobile devices [\[29\]](#). ARKit is an augmented reality framework included in Xcode that is compatible with iPhones and iPads. ARKit lets developers place digital objects in the real world by blending the camera on the screen with virtual objects, allowing users to interact with these objects in a real space.

ARKit uses a technology called Visual Inertial Odometry in order to track the world around the iPad or iPhone. Using the iOS device's camera, accelerometers, gyroscope and context awareness, ARKit performs environment mapping as the device is moved. Sensor fusion of the inertial sensor data with the data from the camera allows for highly accurate location awareness and mapping. The software picks out visual features in the environment such as planes and tracks motion in conjunction with information from the inertial sensors. The camera is also used to determine light sources by which AR objects are lit. Apple's solution to the increased detail and therefore memory usage is a sliding map where old data disappears for new. Users can place anchors to mark creations they want to save. This enables the iOS device to sense how it moves in a room. The user doesn't have to do any calibration, that's a breakthrough in this space.

ARKit-enabled devices are any iPhone or iPad running iOS 11 or later that have the Apple A9, A10, A11, or later processor; these devices are required to run the advanced Metal graphics.

ARKit 2: as part of iOS 12, Apple revealed a second-generation version of ARKit. ARKit 2 brought a new vivid augmented reality experience to apps that allowed them to interact with the real world in new ways. With ARKit 2, multiplayer AR

games were possible, as well as tracking 2D images, and detecting known 3D objects like sculptures, toys, and furniture.

**ARKit 3:** 2019's ARKit 3 includes Motion Capture so developers can integrate people's movement into their app and People Occlusion (fig. 39). Both these technologies mean AR content will show up naturally in front of or behind people to enable more immersive apps. There's support for both the front and rear cameras now, while the front camera can also track up to three faces. However, these features all need Apple's latest devices with A12/A12X Bionic chips, Apple Neural Engine and TrueDepth Camera.



Figure 39: ARKit 3 People Occlusion, Occlusion off (left) and Occlusion On (right) [29]

ARKit development is enabled in Native iOS environment, SceneKit integration and integrate with third-party tools such as Unity and Unreal Engine. Apple has also a new app for developers - Reality Composer that enables developers to prototype and produce AR experiences with no prior 3D experience. There's a drag-and-drop interface and a library of pre-existing 3D objects and animations.



- *Vuforia* is an augmented reality software development kit (SDK) for mobile devices that enables the creation of augmented reality applications [31]. Originally

developed by Qualcomm, it has been acquired by PTC Inc. in November 2015. It uses computer vision technology to recognize and track planar images and 3D objects in real time. This image registration capability enables developers to position and orient virtual objects, such as 3D models and other media, in relation to real world objects when they are viewed through the camera of a mobile device. The virtual object then tracks the position and orientation of the image in real-time so that the viewer's perspective on the object corresponds with the perspective on the target. It thus appears that the virtual object is a part of the real-world scene.

The Vuforia SDK supports a variety of 2D and 3D target types including 'markerless' Image Targets, 3D Model Target, and a form of addressable Fiducial Marker, known as a VuMark. Additional features of the SDK include 6 degrees of freedom device localization in space, localized Occlusion Detection using 'Virtual Buttons', runtime image target selection, and the ability to create and reconfigure target sets programmatically at runtime. In more details the core functionality that offers is:

- Model Targets recognize objects by shape using digital 3D models. Place AR content on multiple objects from multiple views on a wide variety of items like industrial equipment, vehicles, and toys.

- Area Targets utilize 3D scans of a location and are designed for experiences in large, indoor spaces. From retail stores to factory floors - place content anywhere in your environment.

- Image Targets are the easiest way to put AR content on flat objects such as magazine pages, trading cards and photographs.

- Multi Targets are for objects with flat surfaces and multiple sides, or that contain multiple images. Product packaging, posters and murals all make great Multi Targets.

- Cylinder Targets enable you to place AR content on objects with cylindrical and conical shapes. Soda cans, bottles and tubes with printed designs are great candidates for Cylinder Targets(fig. 40).

- Object Targets are created by scanning an object. They are a good option for toys and other products with rich surface details and a consistent shape.

- VuMarks allow you to identify and add content to series of objects. They're a great way to add information and content to product lines, inventory and machinery.

- Ground Plane detection as part of Smart Terrain enables digital content to be placed on horizontal surfaces in your environment, such as floors and tabletops.

It supports the detection and tracking of horizontal surfaces, and also enables you to place content in mid-air using Anchor Points (only for ARCore/ARKit enabled devices [49]).



Figure 40: Vuforia Cylinder Targets [31]

Vuforia SDK supports both native development for iOS and Android while it also enables the development of AR applications in Unity that are easily portable to both platforms.



## wikitude

- *Wikitude* [30] is a mobile augmented reality (AR) technology provider based in Salzburg, Austria. Founded in 2008, Wikitude initially focused on providing location-based augmented reality experiences through the Wikitude World Browser App (see 2.2). In 2012, the company restructured its proposition by launching the **Wikitude SDK**, a development framework utilizing image recognition and tracking, and geolocation technologies.

The Wikitude SDK is the company's core product. First launched in October 2008, the SDK includes image recognition & tracking, 3D model rendering, video

overlay, location based AR. In 2017 Wikitude launched its SLAM technology (Simultaneous Localization And Mapping) which enables object recognition and tracking, as well as markerless instant tracking.

For location-based augmented reality, the position of objects on the screen of the mobile device is calculated using the user's position (by GPS or Wifi), the direction in which the user is facing (by using the compass) and accelerometer. Augmentations can be placed at specific points of interest and afterwards viewed through the devices' screen or lenses.

Since August 2012, Wikitude also features image recognition technologies that allow for tracker images to trigger augmented reality technology within the app. The software identifies relevant feature points of the target image (also known as marker). This allows to overlay and stick augmentations in specific position on top or around the image.

In 2017 Wikitude launched its SLAM technology. Instant Tracking, the first feature using SLAM, allows developers to easily map environments and display augmented reality content without the need for target images (markers). Object Recognition is the latest addition based on SLAM, with the launch of SDK 7. The idea behind Object Recognition and Tracking is very similar to Image Tracking, but instead of recognizing images and planar surfaces, the Object Tracker can work with three-dimensional structures and objects (tools, toys, machinery etc.). Object Tracking enables real-time 360 augmented reality experiences around physical objects.

In 2018 with SDK 8 comes Scene Tracking and Extended Tracking. Scene Tracking makes it possible to recognize, track and augment feature-rich rooms, scenes, and larger objects. Extended Tracking allows digital augmentations, attached to objects, scenes, or images, to persist in the user's field of view even when the initial target is no longer in the frame. Once the image or object target is tracked, users can continue the AR experience by freely moving their device. This feature incorporates the SLAM algorithm at the base of Wikitude's Instant Tracking technology.

Also, one of the features is Seamless AR Tracking (SMART). SMART switches to Google's ARCore or Apple's ARKit, when the devices support it. When not, Wikitude's technology is launched, making it possible to work with Instant Tracking on a much wider range of devices.

The main downside is that it is not free, in contradiction with the previous SDKs that mentioned.

The cross platform SDK is available for Android, iOS and Windows operating systems, being optimized as well for several smart eyewear devices. Developers can also build in JavaScript, Unity, Xamarin, PhoneGap, Cordova, and Flutter.





Figure 41: Wikitude Object Tracking [30]

### 2.6.2 Game Engines

A game engine is a software framework designed for the creation and development of video games [50]. Developers use them to create games for consoles, mobile devices and personal computers. The core functionality typically provided by a game engine includes a rendering engine (renderer) for 2D or 3D graphics, a physics engine, a collision detection (and collision response) system, sound, scripting, animation, artificial intelligence, networking, streaming, memory management, threading, localization support, scene graph, and may include video support for cinematics (fig. 42).

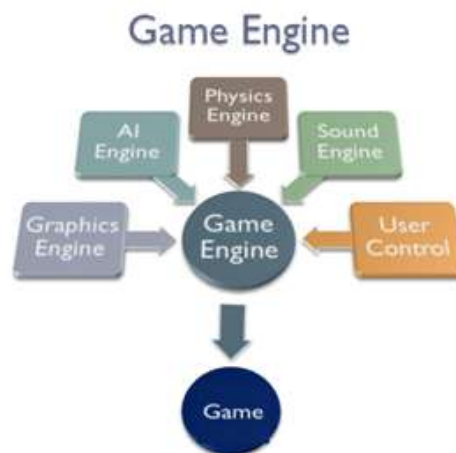


Figure 42: Functionality Provided by Game Engines



The process of game development is often economized, in large part, by adapting the same game engine to create different games, or to make it easier to "port" games to multiple platforms. The beauty and power of game engines, is that they speed-up the development process, by providing a suite of visual development tools, reusable software components and simplification of frequently used tools, elements and processes.

Game Engines are usually built upon one or multiple rendering application programming interfaces (APIs), such as Direct3D or OpenGL which provide a software abstraction of the graphics processing unit (GPU). These APIs are commonly used to interact and communicate with the GPU, to achieve hardware-accelerated rendering. Nowadays they often feature dozens of finely tuned systems interacting to ensure a precisely controlled user experience. The continued evolution of game engines has created a strong separation between rendering, scripting, artwork, and level design. It is now common, for example, for a typical game development team to have several times as many artists as actual programmers.

Furthermore, due to the constant growth of the smartphone application market and increasing competition, popular high-end Game Engines are proving to be a precious tool for developers worldwide, to bring their ideas and games to life, in as many platforms as possible. Below are 2 of the most well-known game engines for mobile development.



- *Unity 3D* [32], initially released on 2005, is a flexible and powerful development platform for creating high quality 2D and 3D games. Emphasizing on portability, Unity currently supports over 20 platforms, including PCs, consoles, mobile devices (iOS and Android) and websites. Additionally, many settings can be configured for each platform. As a result, Unity can detect the best variant of graphic settings for the hardware or platform the game is running, thus optimizing performance and sacrificing visual quality if necessary. Apart from its next-generation graphical capabilities, Unity also comes with an integrated physics engine. Unity offers developers an Asset Store to buy re-usable content and assets for use in their project. To sum up, Unity's primary goal may be the development of 3D video games, however, it is also suitable to create other kinds of interactive content, such as animations, simulations or 3D visualizations. Unity is a fully integrated development engine that provides functionality to create interactive 3D content. Due to its ability to efficiently target multiple platform at once and user-friendly environment, this game engine is an ideal choice for a large portion of developers.

Also, many of the most powerful and promising AR SDKs we mentioned, offer plug-ins integration with Unity3D, making AR application development more user friendly, letting the developer focus on the content of his/her application. For AR development, the world is the scene, so a 3D scene as an environment of development is suitable for our task.



- *Unreal Engine* [51] is a complete suite of development tools for anyone working with real-time technology. From design visualizations and cinematic experiences to high-quality games across PC, console, mobile, VR, and AR. Unreal Engine (UE), initially released on 1998, is a complete suite of game development tools, powering hundreds of games, simulations and visualizations. It is one of the most advanced engines to date, delivering top quality visuals while providing users with a large variety of tools to work with everything they need. Due to its capabilities, efficient design and ease of use it is well-appreciated engine from hobbyists to development studios. It is also available for free. Developers can also port their projects to mobile devices, both iOS and Android. Finally, UE also gives access to its users to the marketplace, to buy re-usable content and add to their project, speeding the development process.

Unfortunately, some of the AR SDKs, don't support development in Unreal's Environment. But, the most powerful AR SDKs for mobile development (ARCore and ARKit) have integrated plug-ins for Development in Unreal Engine.

### 2.6.3 Platform of our choice

All the examined SDKs are great and have advanced technologies and computer vision algorithms. Each one, though, specializes in specific fields and tracking methods. Wikitude SDK has very good for geo-location services and web services, while it provides SLAM capabilities and Object/Scene tracking. Vuforia is probably the best SDK for marker-based AR and model tracking while ARKit and ARCore are great for markerless AR and probably the best SLAM algorithms of them all. All SDKs provide great support and documentation for the developers and each one has a large community, providing guides, tips and solutions in many problems.

Our task, is not to choose the best AR SDK outthere but the best SDK for our needs. Wikitude SDK seemed very promising especially with the scene recognition. Wikitude gives a free student license for a year with some limitations , taking advantage of this, we had a hands on testing. The results of scene and object recognition weren't satisfying as expected. Our beach scenery isn't the best to implement this technology. Other than this, the fact that Wikitude is a licensed SDK, kept us away from further testing since we didn't need any of the other features that it offers. Vuforia SDK is a great tool for mobile AR development and we had great success so far in the development. Since Unity version 2017.1, Vuforia SDK is included as implementation in Unity3D, making the development using Vuforia easier than ever, providing the great support that Unity's community offers. Despite all of these, Vuforia's capabilities couldn't help us in this task since we cannot implement a marker based experience as we mentioned before. Vuforia's plane detection requires ARCore/ARKit supported devices so we decided to work directly with ARCore/ARKit. ARCore and ARKit share the same basic capabilities and provide the best (we have tested) SLAM algorithms, tracking of surfaces (angled, horizontal and vertical) and feature points.

In 2018, Unity Developers published the beta version of AR Foundation package to add high-level functionality for working with augmented reality. Unity 2018.1 includes built-in multi-platform support for AR. AR Foundation provides a platform-agnostic scripting API, making ARCore and ARKit apps that use core functionality shared between both platforms. This enables the development of an app once and the deployment to both devices without any changes, as shown in Figure 43. AR Foundation is still in development and is updated frequently, adding more functionalities. In 2019, with the new version of Unity 2019.1, AR Foundation has been launched officially, bringing even more capabilities for AR development. Focusing on handheld devices, it enables the creation of stunning environments, using smart lighting and lightweight rendering.

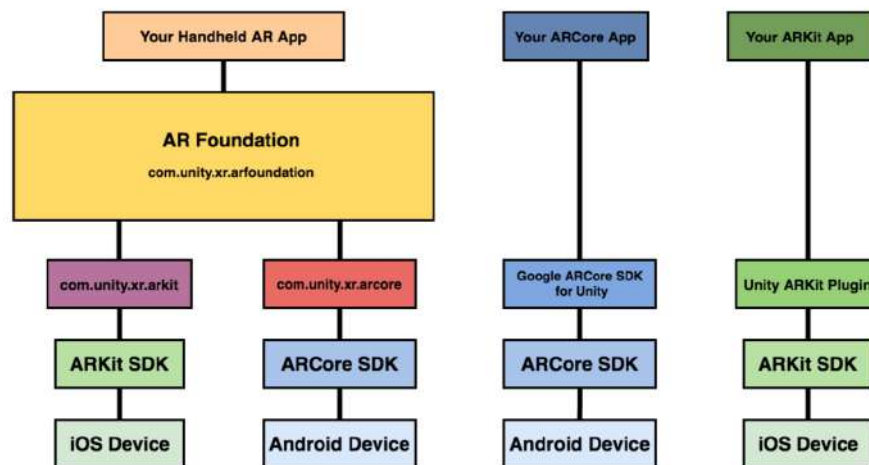


Figure 43: AR Foundation development compared to ARCore & ARKit development [32]

Considering all these, we chose Unity as a development platform in combination with the AR Foundation. ARCore and ARKit support only high end smart phone devices (see the hardware requirements below). A big impact in our choice was the personal knowledge and experience we had on Unity compared with Unreal. Also, the development via AR Foundation, is faster and brings satisfying results using the SLAM capabilities of ARCore/ARKit programming once for two platforms, both Android and iOS. This is the main reason we chose this software for development and we would like to provide the best experience we can offer with the given tools.

## 2.7 Future of AR

Since AR is a new technology, very little data is available so far to back its usefulness in marketing. Aside from AR's economic uncertainties, the technology must clear several technological hurdles before it can be fully realized. In his 2016 paper, "The Most Important Challenge Facing Augmented Reality" [24] Intel principal engineer Ronald T. Azuma lists several areas of improvement for AR:

- AR must have more precise tracking across large environments, indoors and outdoors, day or night.
- A wide field-of-view optical display (glasses or goggles) must be used to fully integrate AR into daily life.
- The AR platform must incorporate an innovative hands-free interface.
- AR object recognition and incorporation of real-world objects into the AR digital space must be improved for the augmented reality experience to be seamless.

Improvement in all of these areas will take time, but the world seems to be driving toward a future where augmented reality is a part of our daily lives. While still in its infancy, AR can easily be seen as a major part of our future society.

The initial demand for such AR systems is likely to be in the enterprise, for professional applications, but consumer applications will eventually drive most of the market. AR displays will enable natural interactions with virtual content that is integrated with the surrounding real world, while the users remain engaged with and aware of the real world. Compact, stylish, and portable wide field-of-view head-worn AR displays have the potential to supplant desktop, laptop, tablet, and even smartphone displays. As more of the real world is instrumented with the Internet of Things, a "physical web" will become established where information is tied to tangible objects and locations, and AR will provide the natural interface to that data [24].

Before AR technology can reach its full potential, it must become more than an afterthought on mobile devices. "For AR to become truly useful, somebody will have to make a platform for it that could host a variety of apps and services," claims

tech industry consultant Tim Bjarin in his 2017 Time article. “Why This Futuristic Tech Will Be The Future Of Computing” [45]. “It’s most likely this platform will exist first in smartphones,” he says, “then, years later, extend to some type of glasses or goggles, like a more fully realized Google Glass.”

Once AR finds a compelling, full-featured platform and it becomes clear that a vast number of consumers are becoming AR proficient, the potential of AR will begin to be fully realized. Every industry from architecture to education, sports, military training, and retail commerce will benefit by embracing AR.

The various industries that will see increased AR activity in the near future are[46]:

- E-Commerce – Many companies will be integrating AR into their websites and mobile apps. In retail, this will result in applications that seamlessly “clothe” a user in sunglasses, jackets, footwear, and jewelry via the camera in the person’s smartphone.

- Digital Marketing – AR technologies will continue to improve the way customers engage with brands. Marketing AR will likely be seen in packaging, on street signs, through gaming apps, and through interactions with other products.

- Geolocation – The ability of mobile devices to inform us of our surroundings be greatly improved over time. AR could benefit everything from real-time travel advisories to restaurant suggestions.

- Educational Resources – Researchers are already attempting to find new and beneficial ways to use AR in training situations. The military and healthcare industries, in particular, are developing powerful AR training simulations.

In conclusion, the future of AR seems very promising since many colossal companies like: Microsoft, Apple, Google, Facebook etc. have invested on this technology. Also, the research community have invested in AR, bringing solutions and algorithms to overcome the problems that, currently, the AR faces. With these two major investors by its side, Augmented Reality seems to be the next “big thing” in the field of technology.

## 3 Case Analysis on Coastal Erosion

### 3.1 Introduction

Today's coastal regions face intense problems caused by the rapid urbanization, coastal erosion, sea level rise, global warming and climate change. These factors have a huge impact on coastal communities. Especially in Greece with a coastline length of 17400km and with many cities and residential areas at the coastal regions the above factors play an important role. The biggest island of Greece is Crete which is predominantly based on services on tourism and agriculture. Since 1970 Crete became a popular tourist attraction, it has more than 2.000.000 tourists every year and this number is increasing.

Greece had about 32 million tourists in 2018 in comparison to 15 million in 2010. It is the seventh fastest growing major travel destinations in the world. Crete is the fifth largest island in the Mediterranean and only the region of Crete contributes 5% of the Gross Domestic Product (GDP) of the country. In the period between 2000 and 2009 the GDP of Crete increased 57.5%. Coastal areas face different problems due to the over-exploitation. Tourism facilities are yearly constructed along the coastal areas to bring more tourists and infrastructure had to be constructed to bring these people to their destinations. Coastal areas suffer from overbuilding and the area seems to have more concrete than vegetation. Coastal erosion is becoming a serious problem more than ever before. It is definitely a physical process but it is influenced by man-made constructions [12].

The massive influxes of tourists have pressed the coastal regions with nice beach to create big tourist developments. Hotels, marinas, roads, restaurants, facilities for recreation and sport activities are some of them. These results in great pressure mainly on resources and on the marine ecosystems. Natural habitats like of the sea-grass meadow have been removed to create open beach, other tourist developments have been built directly next and on the beaches. Careless constructions and resorts have destroyed the beauty of the environment. Tourism is a crucial aspect of the Greek economy given the pleasant climate and sea conditions which contribute to Greece's overall popularity as a tourist destination. In the Greek coastal zone, there are major conflicts between the demand for tourism on the one hand, and coastal preservation on the other. The Greek coastal zones face problems with delineation and definition of public land causes significant uncertainties among land owners regarding where the public domain ends. For the protection of the coastal areas it is important for Greece to conduct an evaluation of planning and legislative tools in relation to these zones.



## 3.2 Our Case Analysis

The area under examination is Georgioupoli, located approximately within 35 15 51N to 35 21 12N and 24 15 43E to 24 18 04E (fig. 44). Its population is about 2800 people and covers an area of 51km<sup>2</sup>. Georgioupoli is situated at the Northeaster part of the Crete in Greece. Georgioupoli is between Rethymnon and Chania 20km from Rethymnon and 38km from Chania. It has evolved to a major tourist attraction the last 30 years, because of the sandy beach and the beautiful surroundings [12].



Figure 44: Location of Georgioupoli, Crete, Greece

### 3.2.1 Economic Analysis

As stated in "Land Use Planning for Sustainable Development of Coastal Regions" [25]: One of the most significant problems is the coastal erosion. It takes place through strong winds and high waves and storm conditions and results in loss of land and beach. One can observe at the satellite images (fig. 46) the coastal erosion at Georgioupoli. The first image is taken in 2003 and the second one in 2016. The sandy coast almost disappeared in the second one. As one can see there are more buildings in the second image and a road is constructed between the residential part of the region and the coast (fig. 45). Using Geographic Information System (GIS) technologies it is found that the length of the beach is 400m and the average width 40m which makes 16.000 square meters loss of sandy beach. It is estimated a loss of economic value 10 euros per square meter per day in Greek beaches that means 160.000 euros per day for such a small village like Georgioupoli in Crete.





Figure 45: Georgioupoli's Beach

In "Monitoring the changes of the Coastal Areas using Remote Sensing Data and Geographic Information Systems" [12] The total lost surface of the sandy beach in the 13 years period (2003-2016) (fig. 46) is calculated to be 20862.40 square meters. This means that the region had an economical loss of 208624 euro per day due to erosion. Also, the analysis showed that there was significant change in the land use and the results validated that the area without vegetation, due to construction, increased from 83.219,4 square meters in 2003 to 106.805,34 square meters in 2016, an increase of 20%. Moreover, the vegetation area decreased from 292.367,52 square meters to 268.780,05 square meters, a reduction of 8%.

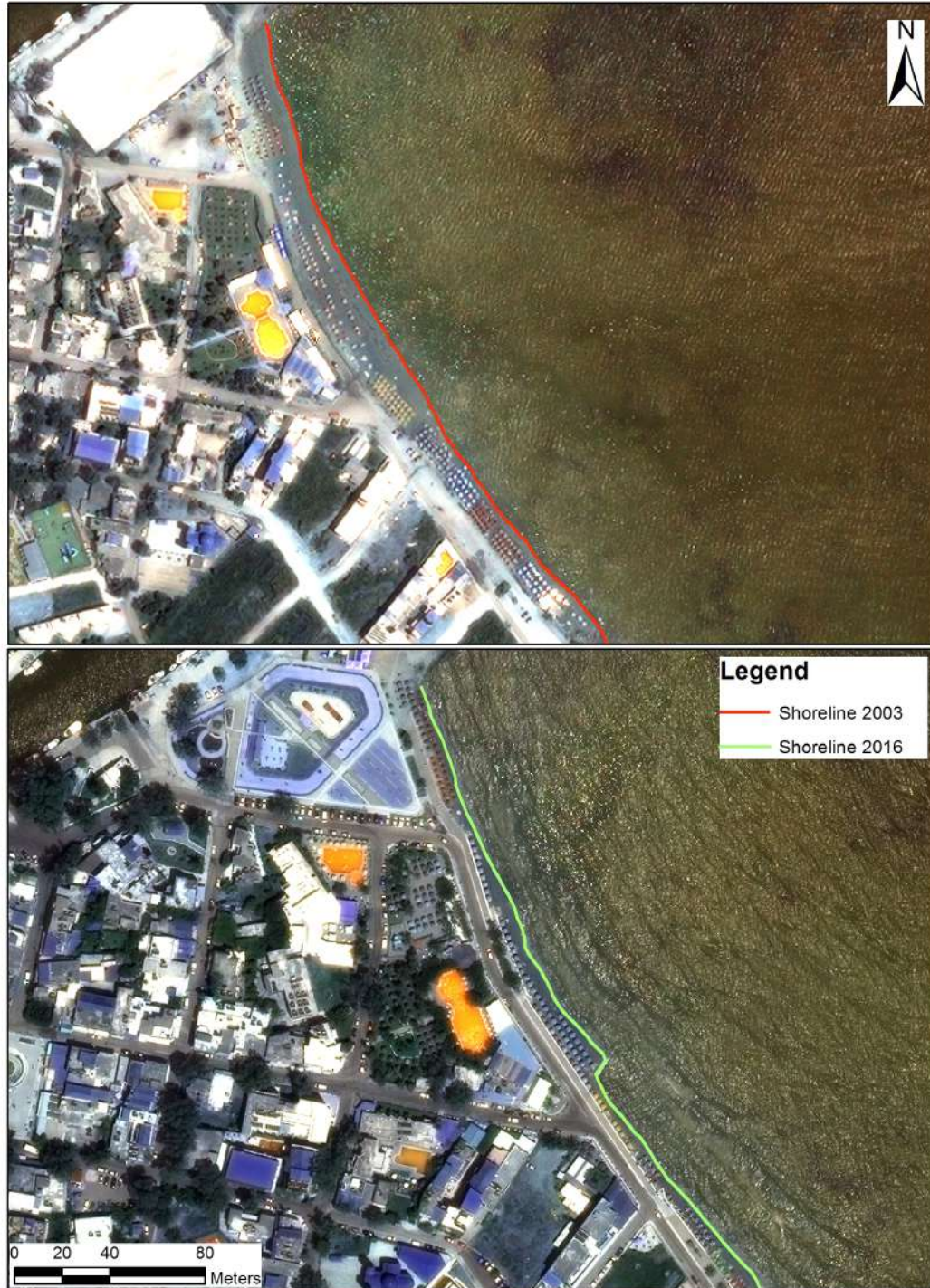


Figure 46: Coastal zone changes in Georgioupoli [12]

Human intervention is a major cause for coastal erosion. The construction of different types of hard structures including seawalls, breakwaters and roads are a major factor for coastal erosion and beach loss [25].

The current thesis is focusing on the coastal zone of Georgioupoli and its vulnerability as a result of the lack of spatial planning. The area is selected because it concentrates the characteristics of a typical coastal touristic zone, which faces



rapid intense not always well planned touristic expansion. The examined zone has been diachronically influenced by the liberalization of construction regulations, an unqualified private sector emerged, hastily developing construction mostly without government oversight and without building permits[25].

There are some main laws for constructions at Greek beaches: a) The leasing of seashores and beaches is allowed for works related to trade, industry, land and sea transportation, or “other purposes serving the public good”, b) beach zone 50 m wide, c) Access roads to the beach of minimum width 10 m., means: expropriations of land properties, d) fences are prohibited in a zone of 500 m from the beach in areas not covered by urban plan. Exceptions: when agricultural fields have to be protected, and e) “Light”, non permanent constructions are allowed in the seashore zone, meant to serve public recreation (tents, open bars etc.)[25]. However there is a need for more detailed and strict regulations for building a construction. The current thesis will focus on the problems of coastal erosion and beach loss.

### 3.2.2 Environmental Analysis

Beaches are considered both valuable economic resources for the Mediterranean countries and vulnerable to climate changes. Beach erosion has significant impacts, since coastal populations, activities, infrastructure and assets are exposed to damages/flooding. It is particularly alarming for the Cretan beaches because (i) of their limited size and sediment supply and (ii) beaches represent the most valuable natural resource of Greece, as they are the main focus of the “sun and beach” tourism (MAP, 2005)[13].

Sea level rise (SLR) (long-term and short-term) represents, probably, one of the most significant beach threats since beaches respond with retreat. IPCC (Intergovernmental Panel on Climate Change), 2013 suggests that mean SLR, in 2100, will be between 0.26 and 0.82 m higher than that of period 1986-2005. Other studies, suggest much higher rises for the same period (e.g. 0.87-1.8 m by Mori et al [? ]).

The Intergovernmental Panel on Climate Change (IPCC) is an intergovernmental body of the United Nations that is dedicated to providing the world with objective, scientific information relevant to understanding the scientific basis of the risk of human-induced climate change, its natural, political, and economic impacts and risks, and possible response options.

For a 0.82m sea level rise (the high estimate of IPCC for 2100), the effects will be devastating since all the beaches will lose more 20% of their width, 93% more than 50% and 41% ( 40% of North, 56% of East and 38% of South Crete) will be entirely lost/inundated (see negative values of beach widths, in fig. 47(a)). The negative values of beach widths suggest that not only the beaches will be entirely lost, but also that human infrastructure/activities will be likely seriously affected. In the case of a sea level rise of 0.87 m (low estimate for 2100 by Mori et al [? ].) the effects will be similar with the previous examined scenario. The East and South beaches are the most vulnerable since they will lose more than 50% of their width,

while 50-56% will be entirely lost/inundated. The losses of the North sector are not less important, since 80% of the beaches will lose more than 50% of their width and 40% will be entirely lost/inundated [13]. The beach of Georgioupoli belongs on the north side of Crete (fig. 44)

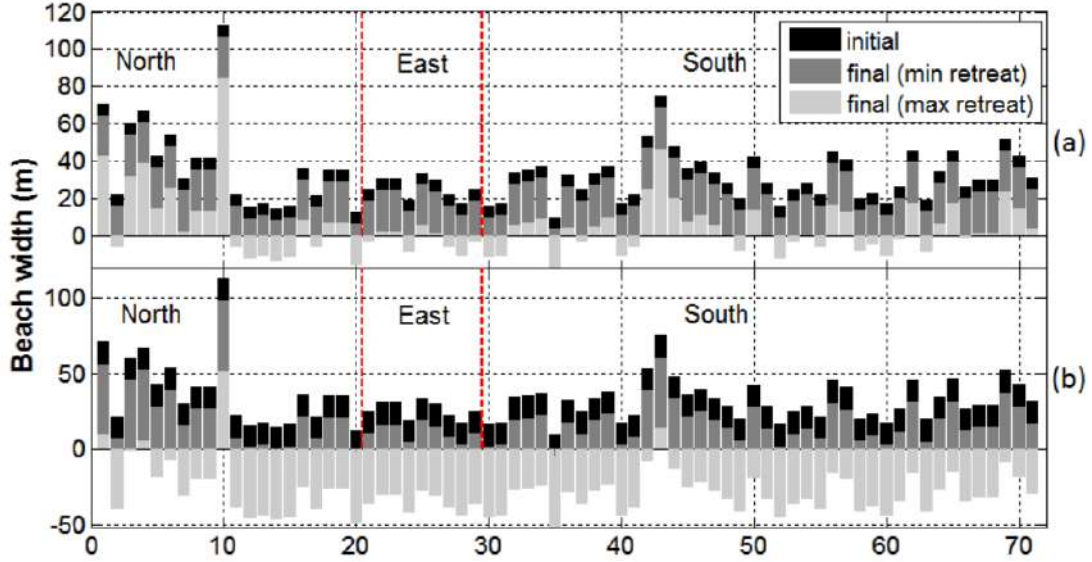


Figure 47: Minimum and maximum retreats of Cretan beaches for sea level rises of (a) 0.82 m and (b) 1.86m estimated on the basis of the low and high mean of the model ensemble projections. Final widths values less than zero show beaches that will be entirely lost. 71 Beaches Showcase. [13]

Finally, the main objective of the present contribution is to assess the vulnerability of the highly touristic Georgioupoli's beach in relation to the anticipated SLR, using the technology of Mobile AR, to bring a new way of geographic data visualization and showing the potential that can have on the future of Data Visualization in general.

### 3.3 Data Showcase

In this section all the geographical data that have been used in this thesis, are presented. The data that have been used, are not based on personal work and research. All the sources, from where the data have been extracted, are included in the reference section and they are linked in the text.

#### 3.3.1 Coastal Retreat Extraction

The model that extracts more accurately the tendency of Georgioupoli's Beach, with characteristics of low slope and sediment of sand is 1. The equation was taken

by "Assessment of vulnerability of the eastern Cretan beaches (Greece) to sea level rise" [13] and represents the mean of the tendency.

$$S = 0.05\alpha^2 + 8.12\alpha - 0.46 \quad (1)$$

$\alpha$ : Sea Level Rise (SLR) in meters

As it's stated: the models were applied using linear profiles with slopes of 1/10, 1/15, 1/20, 1/25 and 1/30. Experiments were carried out using varying wave conditions, i.e. wave heights (H) of 0.5, 1, 1.5 m, wave periods (T) 4-5 sec, and 10 different identified sediment grain sizes ( $d_{50}$  of 0.2, 0.33, 0.50, 0.80, 1, 2, 5, 10, 20 and 30 mm). For all cases, 12 sea level rise scenarios (0.10, 0.15, 0.22, 0.30, 0.40, 0.50, 0.75, 1, 1.25, 1.50, 2 and 3 m) were tested. Totally 5500 experiments were carried out in Crete [13].

Using equation (1), the mean values of coastal retreat for Sea Level Rise (SLR) 0.5 meter and 1 meter is 3.6 meters and 7.7 meters, respectively. Given this, 110 key points were extracted. The points are geo-locations(x,y) that are geo-referenced to the Hellenic Geodetic Reference System 1987 (HGRS87) (see subsection 3.3.3 for further info).

### 3.3.2 Visualization of Data

The extracted points (see Tables 1,2,3) used in a GIS Software, ArcGIS. ArcGIS is a geographic information system (GIS) for working with maps and geographic information maintained by Esri. It is used for creating and using maps, compiling geographic data, analyzing mapped information, sharing and discovering geographic information, using maps and geographic information in a range of applications, and managing geographic information in a database [47], [48].

The aerial image below (fig. 48) is a real sub-scale map exported with ArcGIS showing the tendency of the beach as mentioned. The image is also geo-referenced to the Hellenic Geodetic Reference System 1987 (HGRS87).

In the image is presented mostly the built-area of the beach (Lemonia Ragia and Pavlos Krassaki et al. [12]). The built-area is about 500 meters, and below is presented the state of the beach with 3.6m and 7.7m retreat respectively. In the first scenario most of the beach at this part has been lost, leaving no room for further exploitation for tourism by the local companies and hotels. In the second scenario, not only the most of the beach is wiped out but there will be catastrophic consequences for the local companies. There will be flooding and destruction on the roads and local infrastructure.

The built-area will lose around 1800 square meters of soil in the first scenario, and 3850 square meters in the second. Overall, the beach will lose 18000 square meters in the first scenario and 38.500 square meters in the second, considering that

the beach is around 5km (as mention in "Monitoring the changes of the Coastal Areas using Remote Sensing Data and Geographic Information Systems" [12]).

The following image was used in designing and 3D construction of the virtual content that will be super imposed the real world, in order to achieve the visualization of the future state of the beach. More on the procedure will be explained on the Implementation (section 6).

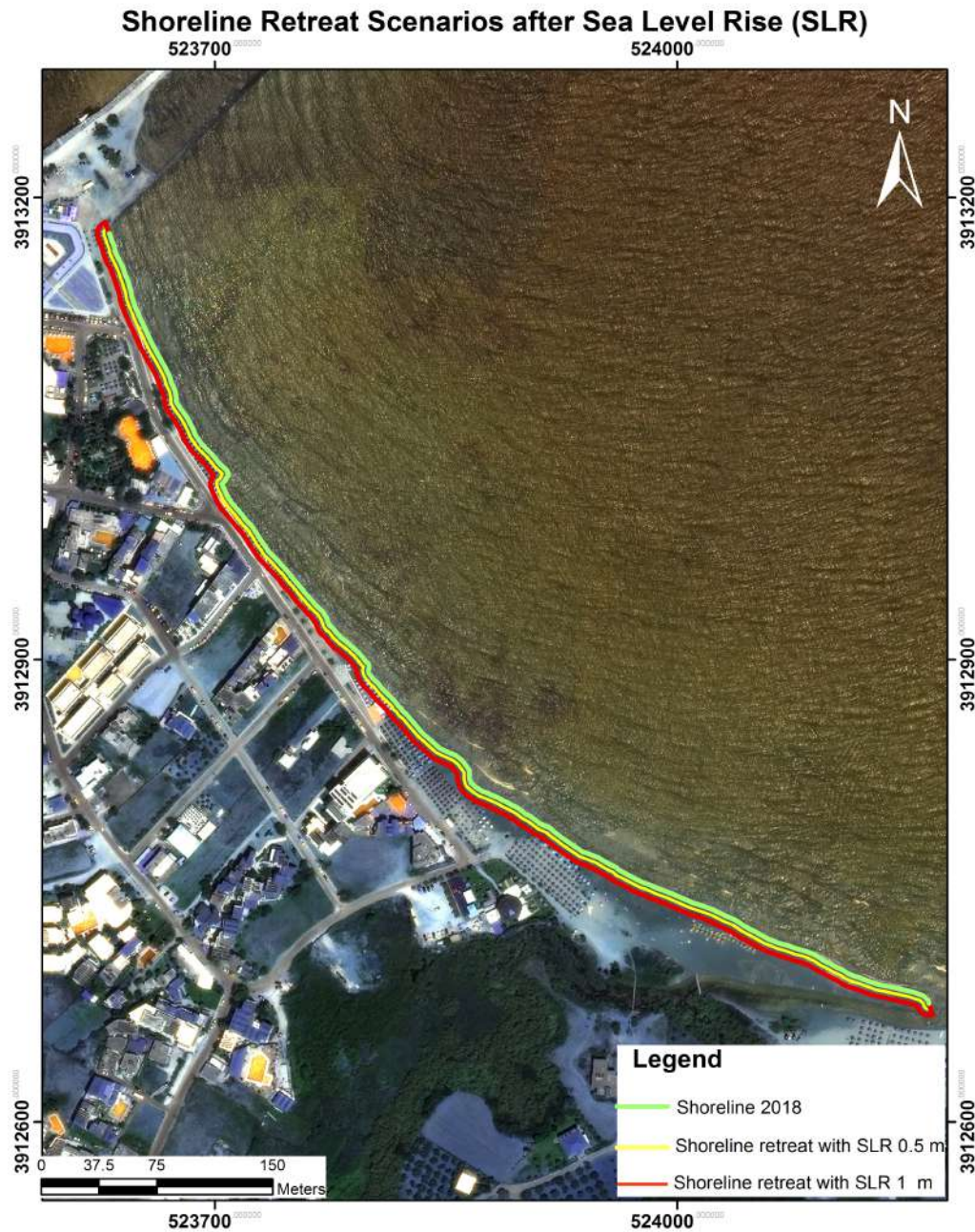


Figure 48: Shoreline Retreat Scenarios for SLR 0.5m/1m [12]



### 3.3.3 Tables of Data

Below the Tables of the points that have been used in ArcGIS for the extraction of the aerial image, see in figure 48. The Table 1 are key points of the current state of the shoreline, as taken in 2018 (green in the Image). In the Table 2 are the same points with shoreline retreat of 3.6 meters (yellow in the Image). And finally the Table 3 are the points with shoreline retreat of 7.7 meters (red in the Image). All the points are geo-referenced to the Hellenic Geodetic Reference System 1987 (HGRS87).

Table 1: Points of Shoreline - Current State [12]

No.	x	y	No.	x	y	No.	x	y
0	523631,7	3913177	37	523707,9	3913023	74	523782,4	3912914
1	523633,3	3913172	38	523709	3913020	75	523786,4	3912911
2	523634,9	3913168	39	523706,5	3913016	76	523789,3	3912908
3	523635,5	3913165	40	523704,9	3913014	77	523793,1	3912905
4	523637	3913161	41	523705,6	3913012	78	523796,2	3912901
5	523639,1	3913158	42	523707,2	3913009	79	523798	3912899
6	523640,7	3913154	43	523708,8	3913005	80	523800,1	3912897
7	523642,1	3913150	44	523710,9	3913002	81	523800,3	3912895
8	523643,3	3913147	45	523713,3	3912999	82	523799,6	3912893
9	523644,6	3913143	46	523715,6	3912997	83	523800,1	3912891
10	523645,8	3913139	47	523717,7	3912994	84	523802,1	3912888
11	523646,2	3913135	48	523719,4	3912992	85	523803,7	3912887
12	523648,5	3913129	49	523721	3912990	86	523806	3912885
13	523651	3913123	50	523723,8	3912987	87	523808,7	3912881
14	523653,9	3913117	51	523727,5	3912982	88	523813,2	3912877
15	523656,5	3913112	52	523730,1	3912977	89	523815,2	3912875
16	523659	3913106	53	523732,6	3912974	90	523816,3	3912873
17	523662,2	3913100	54	523735,2	3912970	91	523817,8	3912872
18	523665,2	3913095	55	523738	3912967	92	523819	3912870
19	523668,5	3913089	56	523741,4	3912964	93	523820,8	3912868
20	523670,8	3913084	57	523743,8	3912961	94	523821,9	3912866
21	523671,8	3913080	58	523747,4	3912957	95	523824	3912865
22	523673,9	3913076	59	523749,7	3912954	96	523825,5	3912863
23	523674,9	3913071	60	523752,2	3912951	97	523829,4	3912859
24	523675,2	3913068	61	523755,2	3912948	98	523834,7	3912853
25	523676,4	3913065	62	523757,4	3912945	99	523839,3	3912849
26	523678,8	3913062	63	523760,2	3912942	100	523845,6	3912843
27	523681,5	3913058	64	523762,6	3912939	101	523850	3912841
28	523683,7	3913054	65	523765,2	3912936	102	523856,6	3912838
29	523685,7	3913050	66	523767,9	3912933	103	523860,5	3912835
30	523687,8	3913045	67	523770,1	3912931	104	523863,2	3912831
31	523691,6	3913040	68	523772,5	3912927	105	523865,3	3912828
32	523695,2	3913036	69	523773,9	3912924	106	523865,3	3912823
33	523698,2	3913033	70	523774,6	3912922	107	523866,4	3912820
34	523701,2	3913030	71	523776,6	3912921	108	523869,9	3912817
35	523702,9	3913027	72	523778	3912919	109	523873,4	3912815
36	523705,2	3913025	73	523779,8	3912916			

Table 2: Points of Shoreline - Retreat 3.6 meters inland [12]

No.	x	y	No.	x	y	No.	x	y
0	523872,2	3912811	37	523774	3912918	74	523695,5	3913031
1	523867,6	3912814	38	523771,5	3912920	75	523692,7	3913034
2	523863,6	3912818	39	523770,5	3912922	76	523688,8	3913038
3	523861,7	3912822	40	523769,3	3912926	77	523684,8	3913043
4	523861,7	3912827	41	523767,3	3912929	78	523682,4	3913049
5	523860,3	3912829	42	523765,3	3912931	79	523680,5	3913052
6	523857,9	3912832	43	523762,5	3912934	80	523678,5	3913056
7	523854,8	3912835	44	523759,9	3912936	81	523675,8	3913060
8	523848,3	3912837	45	523757,5	3912939	82	523673,3	3913063
9	523843,6	3912840	46	523754,5	3912943	83	523671,7	3913067
10	523836,9	3912846	47	523752,4	3912945	84	523671,4	3913071
11	523832	3912850	48	523749,5	3912949	85	523670,5	3913074
12	523826,7	3912856	49	523746,8	3912952	86	523668,4	3913079
13	523822,9	3912861	50	523744,6	3912955	87	523667,4	3913083
14	523821,5	3912862	51	523741,2	3912958	88	523665,3	3913087
15	523819,1	3912864	52	523738,7	3912961	89	523662,1	3913093
16	523818	3912866	53	523735,5	3912965	90	523659	3913098
17	523816,2	3912868	54	523732,8	3912967	91	523655,7	3913104
18	523814,9	3912870	55	523729,6	3912972	92	523653,2	3913110
19	523813,6	3912871	56	523727,2	3912975	93	523650,7	3913115
20	523812,6	3912872	57	523724,4	3912980	94	523647,7	3913122
21	523810,6	3912874	58	523721	3912985	95	523645,1	3913128
22	523805,9	3912879	59	523718,1	3912988	96	523642,6	3913135
23	523803,3	3912882	60	523716,6	3912990	97	523642,3	3913138
24	523801,1	3912884	61	523714,8	3912992	98	523641,2	3913142
25	523799,2	3912886	62	523712,9	3912994	99	523639,9	3913146
26	523796,6	3912890	63	523710,6	3912997	100	523638,7	3913149
27	523796	3912893	64	523707,8	3913000	101	523637,3	3913153
28	523796,6	3912895	65	523705,5	3913004	102	523635,8	3913156
29	523795,1	3912897	66	523704	3913007	103	523633,8	3913160
30	523793,4	3912899	67	523702,4	3913011	104	523632,1	3913164
31	523790,6	3912902	68	523701,4	3913015	105	523631,4	3913167
32	523787	3912905	69	523703,6	3913018	106	523629,9	3913171
33	523784,1	3912908	70	523704,9	3913020	107	523628,2	3913176
34	523779,8	3912911	71	523702,8	3913023	108	523704,6	3913021
35	523777	3912914	72	523700,5	3913025	109	523796,4	3912894
36	523775,1	3912917	73	523698,4	3913028			

Table 3: Points of Shoreline - Retreat 7.7 meters inland [12]

No.	x	y	No.	x	y	No.	x	y
0	523869,3	3912808	37	523772,1	3912914	74	523695,2	3913025
1	523864,9	3912811	38	523770,4	3912916	75	523692,5	3913028
2	523859,7	3912816	39	523768,1	3912918	76	523689,8	3913031
3	523857,7	3912822	40	523766,8	3912921	77	523685,3	3913035
4	523857,6	3912826	41	523765,6	3912924	78	523681,1	3913041
5	523857	3912826	42	523764,1	3912926	79	523678,7	3913047
6	523855	3912829	43	523762,4	3912928	80	523676,9	3913050
7	523852,6	3912831	44	523759,5	3912931	81	523675	3913054
8	523846,5	3912834	45	523756,9	3912934	82	523672,5	3913057
9	523841,2	3912836	46	523754,4	3912937	83	523669,6	3913062
10	523834,2	3912843	47	523751,3	3912940	84	523667,9	3913065
11	523828,8	3912848	48	523749,2	3912943	85	523667,3	3913070
12	523823,5	3912854	49	523746,4	3912946	86	523666,6	3913073
13	523819,8	3912858	50	523743,5	3912949	87	523664,4	3913078
14	523818,8	3912859	51	523741,5	3912952	88	523663,5	3913081
15	523815,8	3912862	52	523738,2	3912956	89	523661,7	3913085
16	523814,9	3912863	53	523735,7	3912958	90	523658,5	3913091
17	523813,1	3912865	54	523732,5	3912962	91	523655,4	3913096
18	523811,6	3912867	55	523729,3	3912965	92	523652	3913103
19	523810,4	3912868	56	523726,3	3912969	93	523649,5	3913108
20	523809,7	3912869	57	523723,8	3912973	94	523647	3913114
21	523807,6	3912871	58	523720,9	3912978	95	523643,9	3913120
22	523802,7	3912876	59	523717,9	3912982	96	523641,2	3913127
23	523800,3	3912879	60	523714,9	3912985	97	523638,5	3913135
24	523798,1	3912881	61	523713,3	3912987	98	523638,2	3913137
25	523795,8	3912884	62	523711,6	3912989	99	523637,3	3913140
26	523792,9	3912888	63	523709,7	3912992	100	523636	3913144
27	523792,2	3912891	64	523707,6	3912994	101	523634,8	3913147
28	523792	3912892	65	523704,3	3912998	102	523633,5	3913151
29	523791,9	3912893	66	523701,8	3913002	103	523632,2	3913154
30	523791,8	3912894	67	523700,2	3913006	104	523629,9	3913158
31	523790,2	3912896	68	523698,7	3913009	105	523628,2	3913162
32	523787,6	3912899	69	523697,4	3913016	106	523627,4	3913166
33	523784,3	3912902	70	523698,3	3913018	107	523626,1	3913169
34	523781,5	3912905	71	523699	3913019	108	523624,4	3913174
35	523776,9	3912908	72	523699,7	3913020	109	523697	3913012
36	523773,9	3912912	73	523697	3913023			

## 4 Requirements Analysis

### 4.1 Introduction

Before we begin to think what technologies we might need to build our app, we need to have an idea of what we wish to build, what our application is going to be about, what current needs it fulfills and what new abilities it might give to the users. After having the initial, general idea for our AR application and what we aim to do by creating it within the context of this thesis, it's time to begin mapping it out. In doing so, it will help eliminate any problems and ensure that functionality that we chose to be in the application does not get missed.

In the next sections we provide the requirements gathering process and the characteristics our application should have. Because of the nature of this application, the requirements gathering process was our own and did not include stakeholders and potential users of the application.

### 4.2 Pre-Requirements

Having seen the possibilities of AR and the current state of the technology, and having in mind the case scenarios that were examined in sub-section 3.3 it was quite clear to us that, for both scenarios, that the visualization will take place using Augmented Reality on an outdoors, on-site, mobile augmented reality application. The user will be able to visit the area of interest and experience in real time the visualization with his/her smartphone device using our Mobile AR application. The experience should give him an understanding and clear view of the future changes that coastline will be undergone. Due to inability of visualizing the whole beach at once, three locations were selected along the beach, where user can go and enable the AR experience.

#### 4.2.1 General Requirements

The application needs to be fully functional and provide a complete experience to the end-user. As a location based experience, the physical presence in the Georgioupoli's Beach is required. The application will be available to everyone with a smartphone that meets the requirements of the AR Software (*see 4.4 for Hardware Requirements*). The device needs to be equipped with specific sensors to support the 3D visualization in the real world and the availability of these sensors needs to be checked in order to ensure the optimal flow of the experience. The application needs to provide a means of navigation throughout the spatially distributed content. Since all the information is delivered on a location basis, an interface that helps the user locate the available information relative to his location needs to be included. The application will not be functional outside the area of Georgioupoli, instead a

message will pop-up to the user.

The application is targeting any visitor or inhabitant of the village of Georgioupoli, with no age or educational requirements. The goal is to inform the user about the dangers of coastal erosion, and the future state of the beach. Below we outline additional functional requirements about the overall system:

- The application should be simple-to-use and be minimal.
- The system must provide the 3D visualization to users in Georgioupoli.
- The graphics of AR should be realistic.
- The application should guide the user on how to use the app.
- The application should navigate the user to key locations.
- The application should not provide redundant information.
- The system needs access to the GPS sensor.
- The application needs to have access to the phone's camera.
- The application needs internet access.
- The application should have room for upgrade and further improvement.

#### **4.2.2 Augmented Reality Requirements**

In our application, it is clear that we cannot make use of Fiducial Marker Based Tracking (see 2.5) due to inability of placing markers on coastal environment. Modeled Based and Natural Feature Tracking seem promising, but they are in early stages. Moreover, the environment which we are called to work in, makes the use of such tracking methods more challenging because of the lighting, the constant changes in the scenery and the scale of it. Hybrid Based Tracking may seem the most suitable in our case, but the testing results have shown that GPS can't be trusted as a method of positioning virtual objects accurately in the real world. Thus, a more sophisticated method should be used, combining more than one tracking methods. Simultaneous Localization and Mapping, also known as SLAM, is a combination of Hybrid Based and Natural Feature tracking methods. SLAM makes use of the device's sensors (gyroscope, accelerometer) and feature points of the world (detected by camera feed), in order to create a map of the world and find the position of the device relative to it. By making use of SLAM in combination with some the other tracking methods (GPS, compass etc.), it is believed that we will provide the desired results.



The user will get to the desired location using his/her device's GPS, and when he gets there the AR mode will be enabled. The user should scan the area around and point at the coastline (using some helpful indicator). In addition to this, more requirements the app should have, are:

- The system must run on all devices that meet the requirements of AR Software that we used (*see 4.4 for Hardware Requirements*).
- The system needs to provide 3D Visualization of coastal zone changes.
- The users must be able to enter the AR experience after reaching the location of interest.
- The user needs to be able to re-calibrate the AR content and place it based on his/her visual position.
- The User Interface in AR mode should be minimal and not flooding the screen, giving only the mandatory information.
- The representations should be accessible from all the available viewpoints.
- The AR experience should be as realistic as possible.

#### **4.2.3 Map and Navigation Requirements**

Overall the Map/Navigator requirements:

- The system needs to include a map to assist in navigation.
- The systems should calculate the user's distance from the point of destination.
- The map needs to include point mark over the areas of interest.
- The map needs to annotate the user's location.
- The user must be able to enable/disable the information shown on the map and have zoom in/out capabilities.
- The minimum area to interact with a location in AR mode will be 10 meters radius.

All the listed pre-requirements have been taken into consideration, before proceeding into the development phase of the application.

## 4.3 Use Case Scenarios

In this section we provide the use case diagrams depicting a user's interaction with the application and the relationships between the user and the different use cases in which he is involved.

### 4.3.1 User Enters the Application

The figure bellow illustrates the interactions available after the user has launched the application, and wants to proceed to the core functionalities.

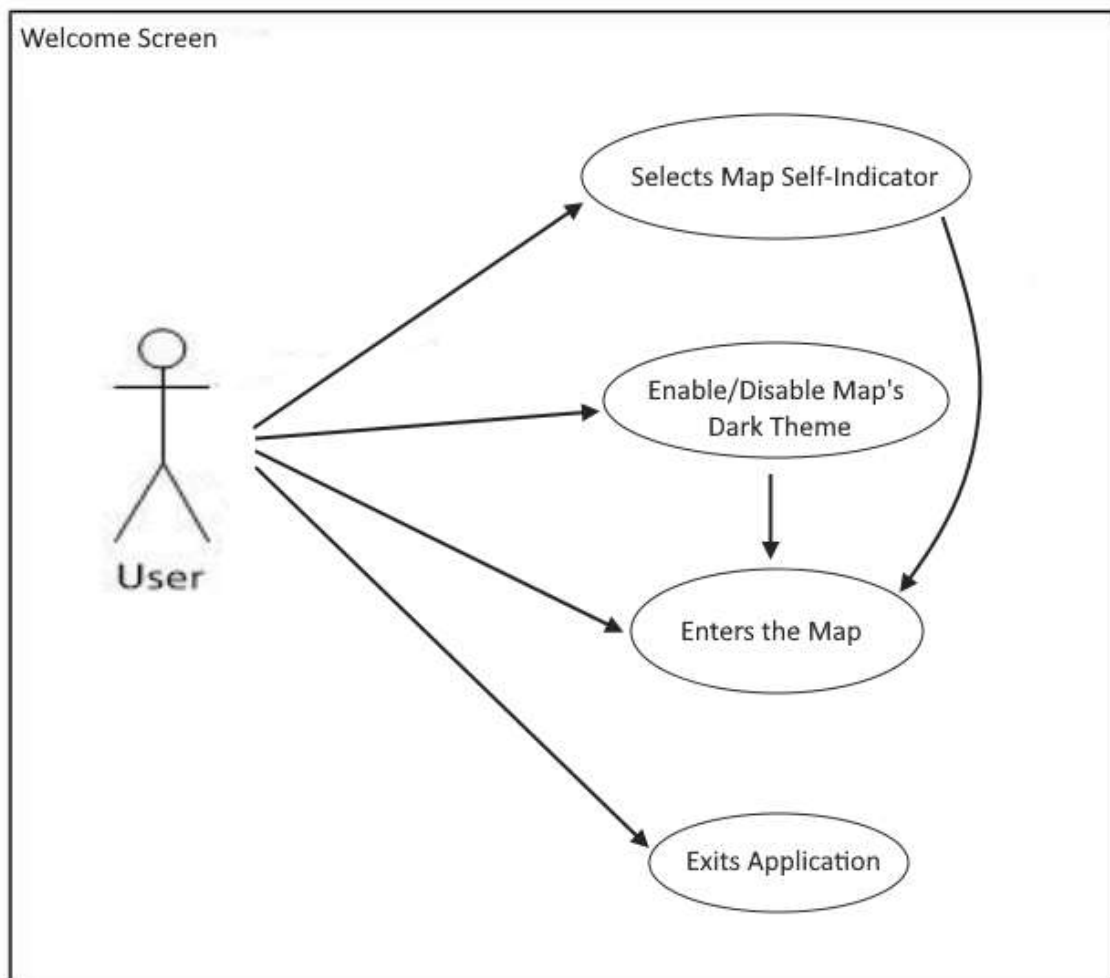


Figure 49: Initial Use Case

- **User enters the map:**

The user presses the "Start" button to enter the map and proceed to the main functionality of the application using the default map indicator and map's theme.

- **User selects map indicator:**

The user can select one of the two different self-indicators that represent the user's location on the map.

- **User selects map theme:**

The user can enable or disable map's dark theme mode. The default theme is a light-street view map. Dark mode gives the map a dark-black theme.

- **User exits application:**

The user can press device's "back" button to exit the application and terminate his experience.

#### 4.3.2 User is Located Outside of Georgioupoli

The interactions depicted below are available after the initial process of the application when user tries to enter the map while he is not located in the area of Georgioupoli.

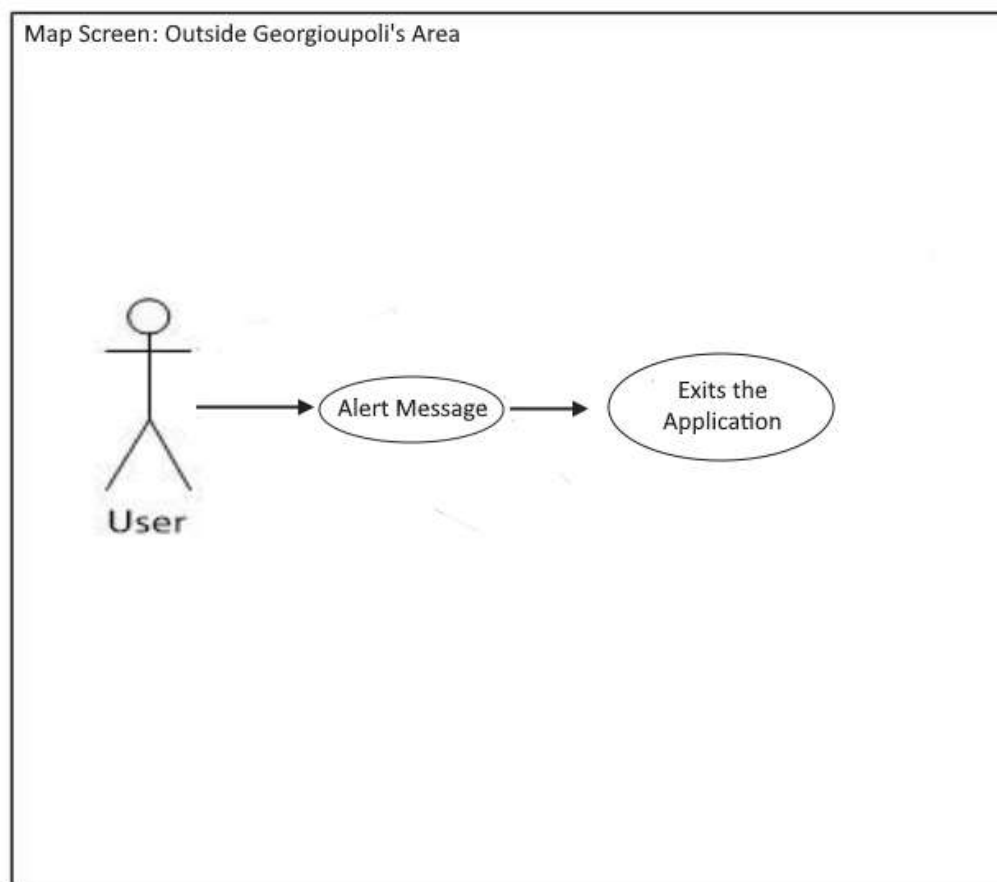


Figure 50: User isn't in Georgioupoli Use Case

- **User reads alert note and exits:**

The user tries to enter the map while he is located outside the area of Georgiopolis. An alert message pops up and an "exit" button is the only option he/she has.

### 4.3.3 User Enters the Map

The interactions depicted below are available after the initial process of the application and after user has entered the map screen. The map settings interaction cases will be described in more detail in the following use case figure (see fig. 52).

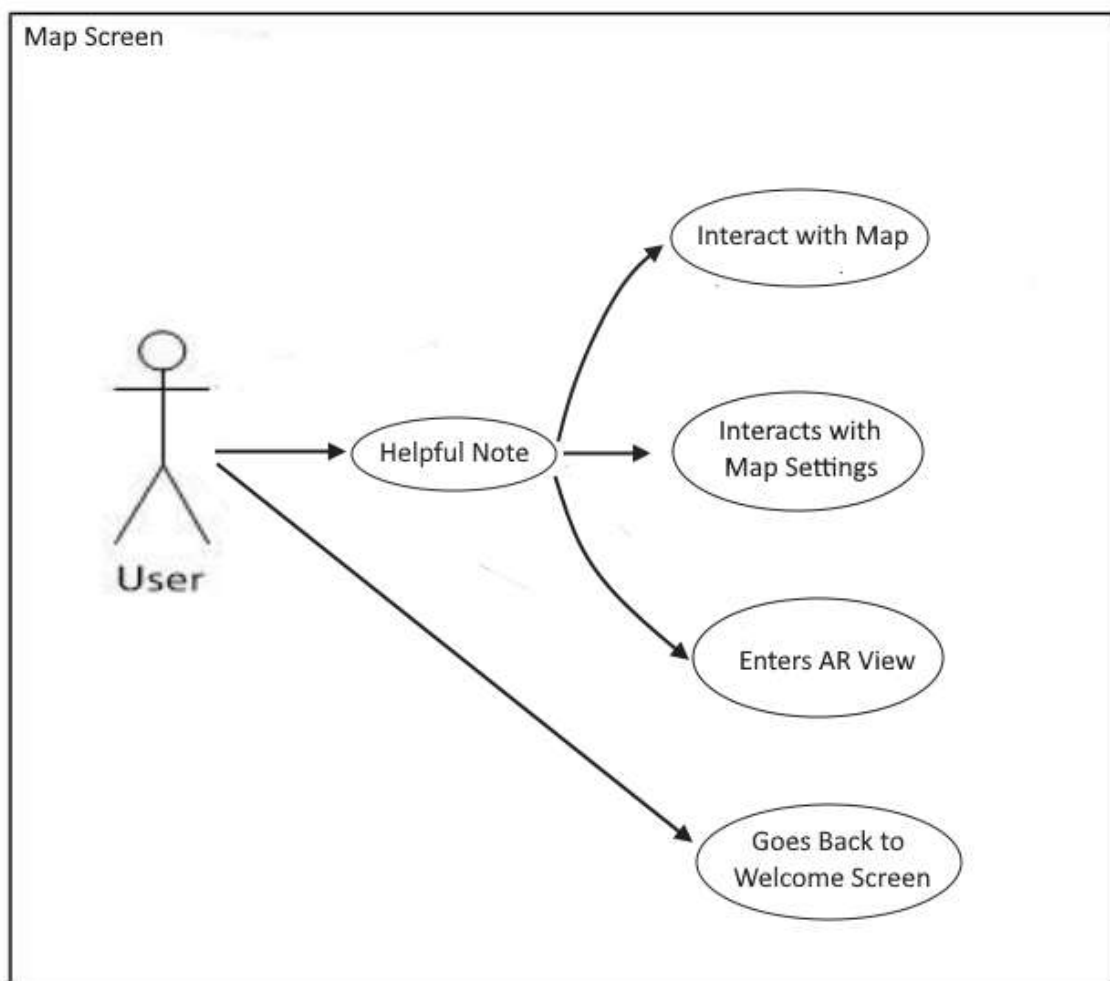


Figure 51: Map Use Case

- **User reads helpful note:**

At the beginning, a helping note with useful directions pops up. The user can read it and press the "dismiss" button to continue the experience.

- **User interacts with map:**

The user can interact with map by rotating it. User can rotate the map using one finger by touching and dragging it on the screen. With the right rotation he/she can find the location of interest's indicators and read at the top of the screen to distance from there.

- **User interacts with map settings:**

The user can press the "settings" button to interact further with the map and the locations (interaction cases will be described in more detail in the following use case).

- **User enters in AR view:**

The user can press the "AR" button, that will be enabled after he reaches to the selected location, in order to enter in the AR mode.

- **User returns to initial screen:**

The user can press device's "back" button to return to the previous screen and re-launch the map with different look or exit the app.

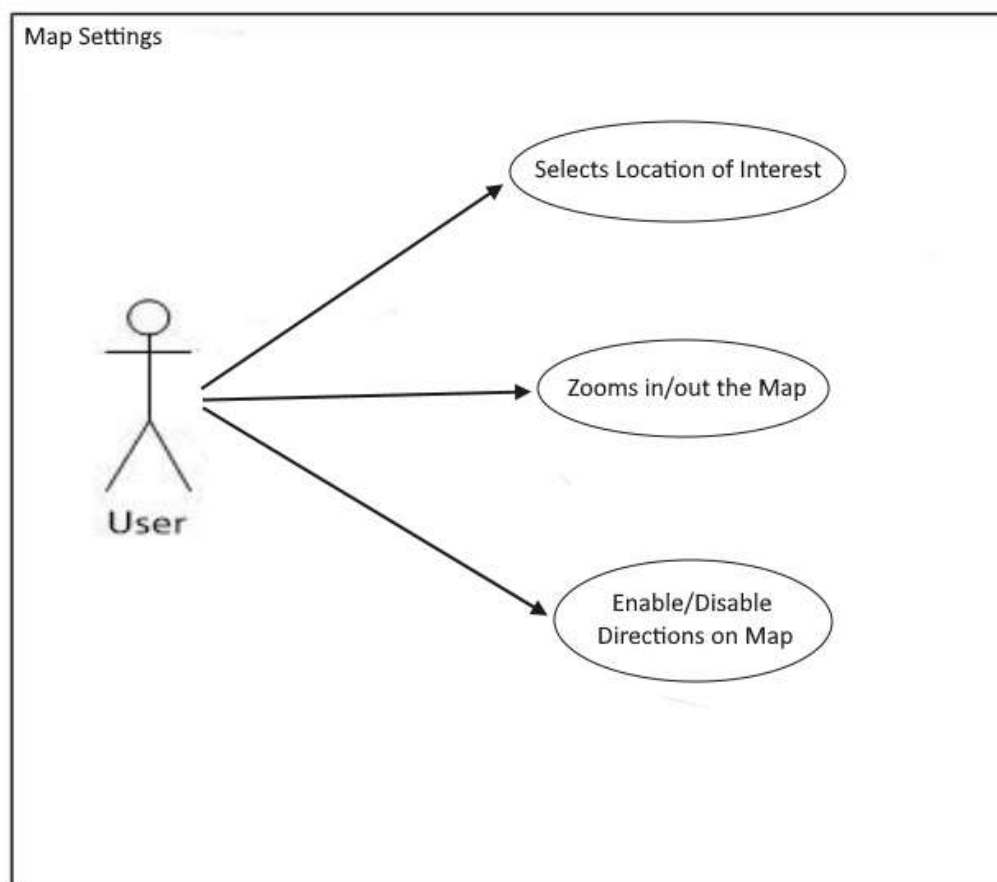


Figure 52: Map Settings Use Case

- **User selects location:**

The user can use a drop-down list to select one of the three locations of interest. The distance from user's location updates accordingly.

- **User enables/disables directions:**

The user can enable or disable the map directions to the selected location. By default it's disabled. The directions change based on the selected location.

- **User zooms in/out:**

The user can press the "plus" (+) button to zoom in the map, or use the "minus" (-) button to zoom out the map and view larger area.

#### 4.3.4 User Enters AR View

The interactions depicted below are available after the user has entered the AR view screen and has seen the procedure that has to be followed in order to experience the visualization. The calibration interaction cases will be described in more detail in the following use case figure (see fig. 54).

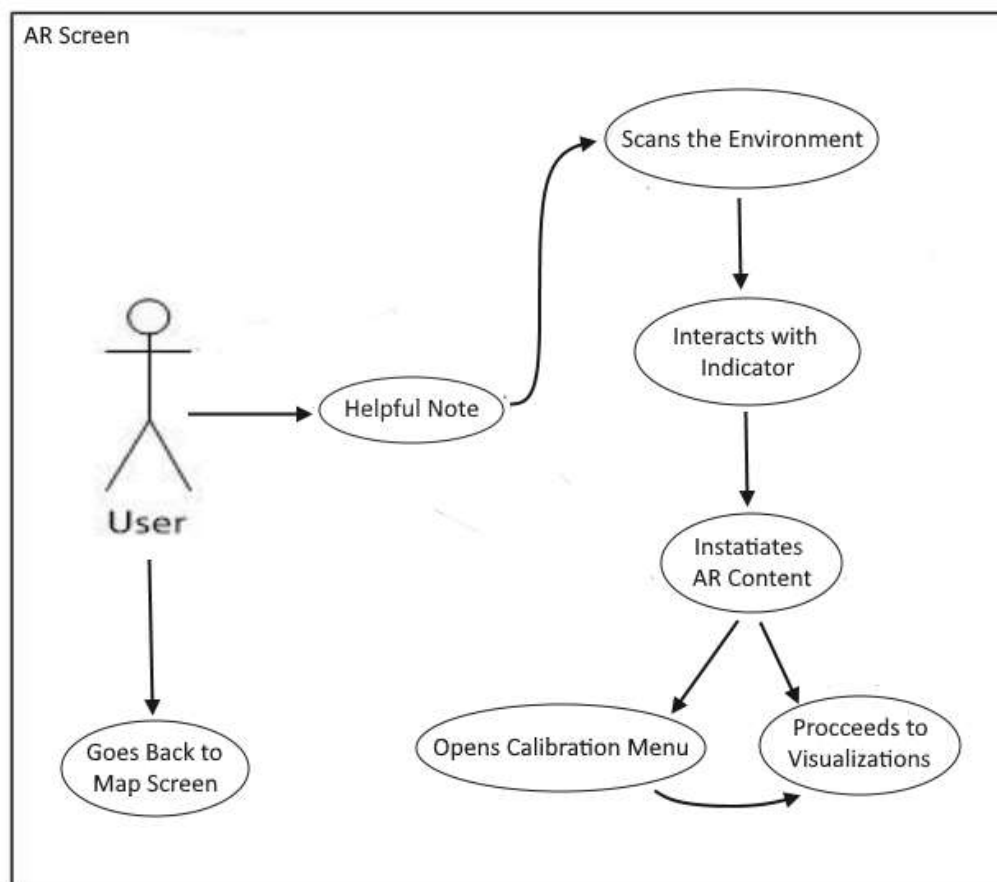


Figure 53: AR View Use Case



- **User reads helpful note:**

At the beginning, a helping note with useful directions pops up. The user can read it and press the "dismiss" button to continue the experience.

- **User scans the environment:**

The User should scan the environment to detect surfaces/planes and feature points. The more the scanning the better the results will be.

- **User interacts with the indicator:**

The user, after successfully scanning the environment, will see an indicator on the scanned surface. The indicator is practically an arrow that shows the direction the device's camera looks at. The user can move the indicator by rotating, rolling and pitching the device.

- **User instantiates AR content:**

The user move his indicator accordingly in order to align it with the shoreline he sees in the real world. The tip of the arrow is where the shoreline starts. Then he presses the "Place Content" button to instantiate the AR content and two new buttons are shown 1) "Settings" button 2) "Done" button.

- **User calibrates content:**

The User presses the "Settings" button in order to calibrate better the AR content (interaction cases will be described in more detail in the following use case).

- **User proceeds to visualization:**

The User presses the "Done" button in order to proceed to the visualization and the core AR functionality.

- **User returns to map screen:**

The user can press device's "back" button to return to the previous screen and access the map after he/she had finished the visualization on the selected location. Then the user can select a new location or terminate the app.

For better results in visualization, a little calibration by user might be needed. The initial height of the content is determined by the scanned ground, so no further calibration will take place on this axis.

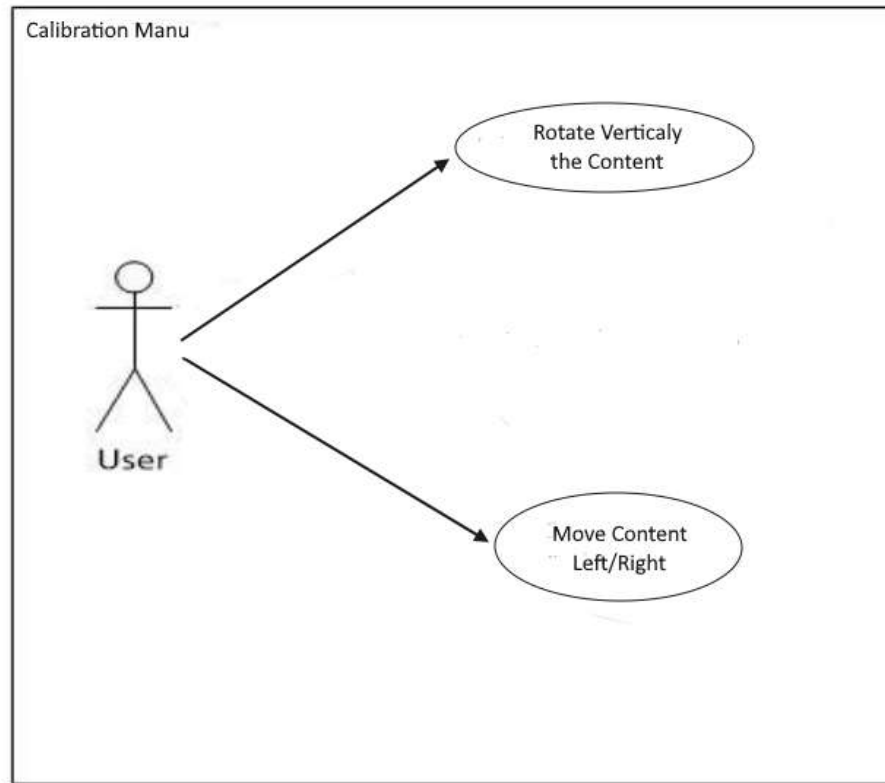


Figure 54: AR Calibration Use Case

- **User rotates vertically the content:**

The user can press the "rotation left" or "rotation right" button in order to rotate the content to match the real world.

- **User moves the content:**

The user can press the "left arrow" or "right arrow" button in order to move the content to match the real world.

#### 4.3.5 User Experiences the AR Visualization

The interactions depicted below are available after the user has proceeded to the AR visualization screen. If all the previous steps have been followed successfully, the user should be able to see how the beach will be in its future state and realize how the erosion will affect the shoreline of the location.

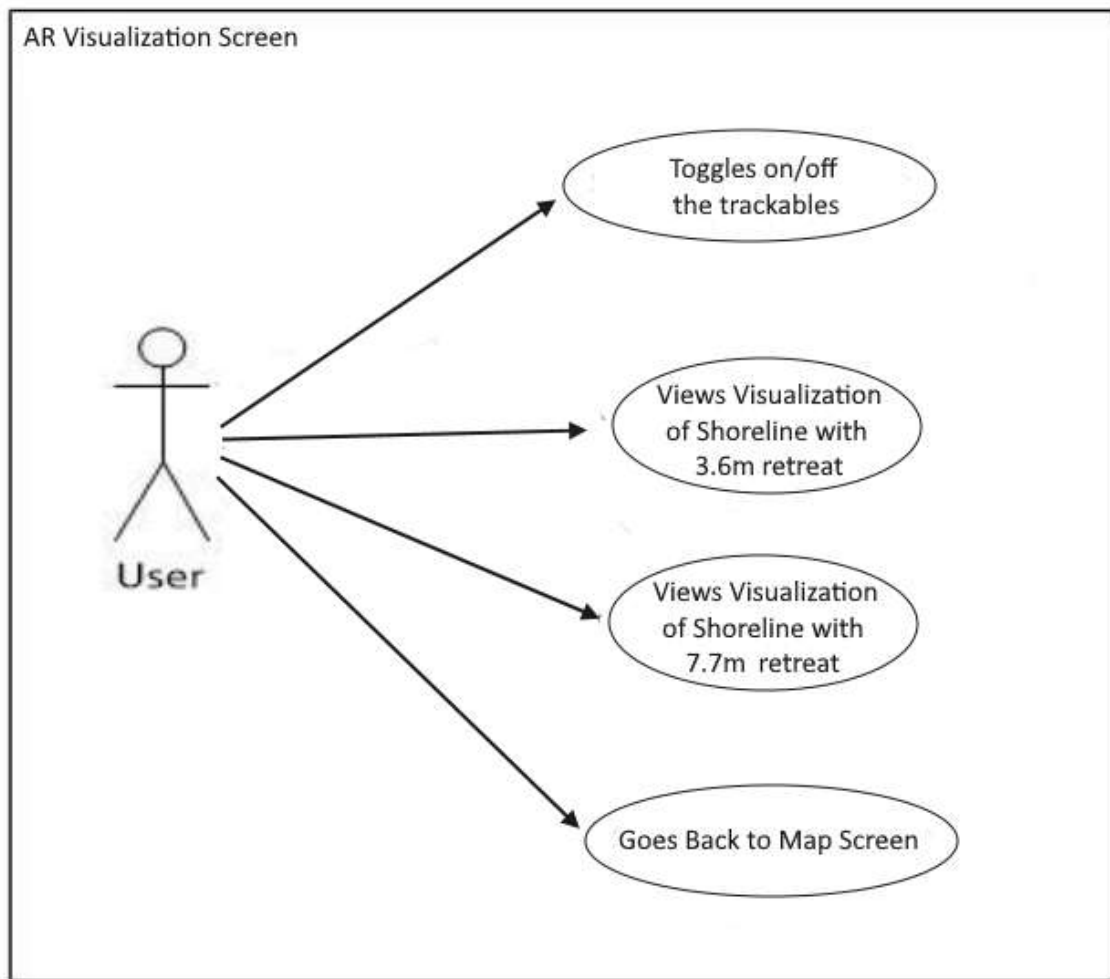


Figure 55: AR Visualization Use Case

- **User toggles off trackables:**

The user can press the "toggle trackables" (white) button in order to turn off the visualization of the trackables the device has detected. Visualizing the trackables helps the user understand what the device sees, but in this phase they are not important and distract the user.

- **User views visualization with 3.6m retreat:**

The user can press the "3.6m retreat" (yellow) button and see in AR the shoreline to change in real time. The new shoreline represents the future state of the current shoreline with 3.6m retreat.

- **User views visualization with 7.7m retreat:**

The user can press the "7.7m retreat" (red) button and see in AR the shoreline to change in real time. The new shoreline represents the future state of the current shoreline with 7.7m retreat.

- **User returns to map screen:**

The user can press device's "back" button to return to the previous screen and access the map after he/she had finished the visualization on the selected location. Then the user can select a new location or terminate the app.

## 4.4 Hardware Requirements

In order to execute successfully the application and experience all the features that has to offer, there are specific hardware requirements your device needs to meet based on the developing platforms we chose and the pre-requirements we have set on the previous subsections:

- The device should have GPS.
- The device should have compass, accelerometer and magnetometer.
- The device should have a back camera.
- The device should have gyroscope.
- The device should have connection to the internet.
- The device should have at least 1GB free RAM.
- The device should run Android 7.0+/iOS 11+.
- The device should be supported by ARCore/ARKit SDKs (see ARCore/ARKit Supported Devices [\[49\]](#) for more information).

## 5 End User Experience

In this section, the application is presented from the point of view of an end user. We make an introduction to the final app declaring its components based on what the user experiences. Furthermore, the final screens that a user sees and interacts with, are presented following by a mini-guide on how to use the application properly and what should be the expectations of a user candidate.

### 5.1 Introduction to the App

Before proceeding to the final screens and the user experience, an intro to this application is essential. First of all, the name of our application is Coastal Zone AR (CZAR for short), and it will be called by this name in this section. The main functionality and the structure of Coastal Zone AR (CZAR) is displayed below in figure 56.

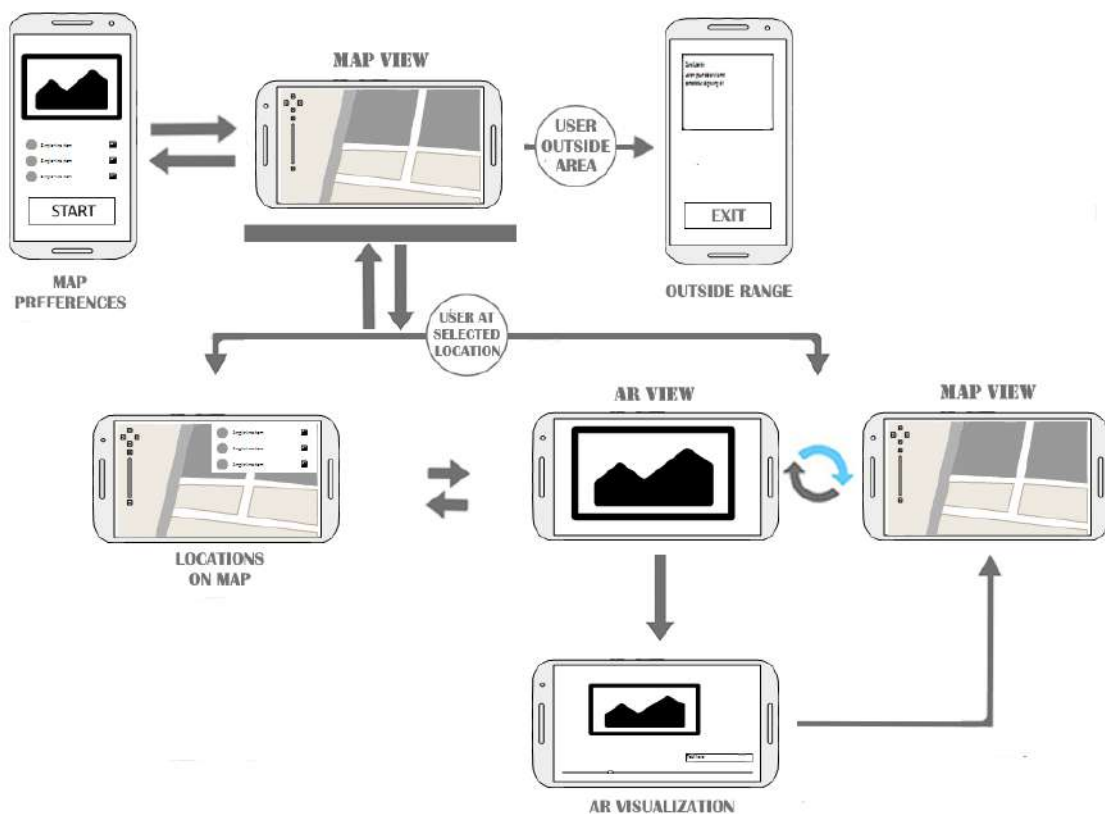


Figure 56: App Basic Structure and Navigation

In short, there is an initial screen where the user starts his/her experience and basically starts the core app. From there, based on user's location the application will provide different content and interaction. The AR visualization can take place

only on specific locations. There are no dead-ends in the app, the user can navigate from any final screen to any other final screen. In the case where the user is far from the area of Georgioupolis the only interaction that's given to the user is to exit the application. Later on this section, we will analyze each part of the diagram above (fig. 56) and we will provide the Graphic User Interface (GUI) of each part.

The application runs on all devices that meets the requirements we mentioned in the previous section (see Hardware Requirements 4.4) on both Landscape and Portrait orientation of the device, no matter the phase of the app.

## 5.2 Initial/Welcome Screen

Upon the activation of the application the user is facing an initial/welcome screen as shown in figure 57. The purpose of this screen is to welcome the user and let him dive into the app smoothly, providing some sort of freedom with some options. In this screen the user can choose whether to use the humanoid self-indicator or the pin-arrow one. The user can also select to enable the dark theme mode for the map and give it a darker look.



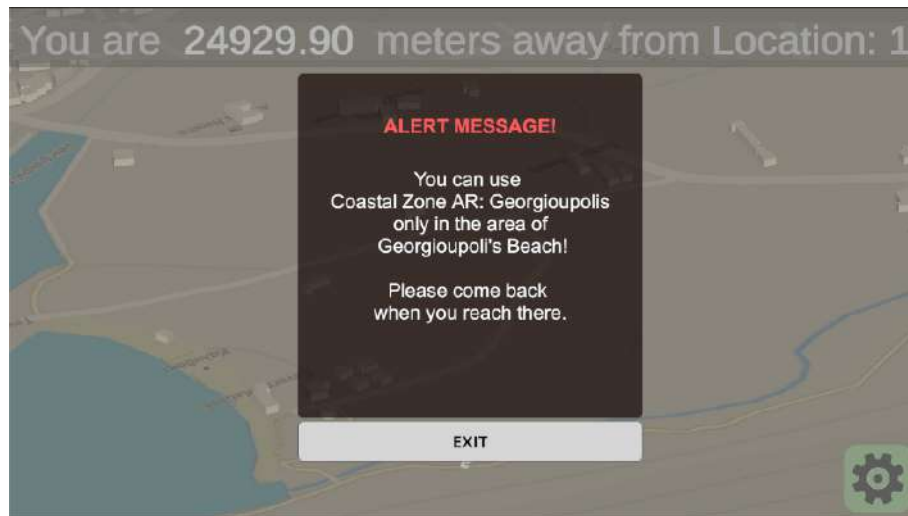
Figure 57: Welcome Screen

After the user has finalized his/her selections, he/she can proceed to the core of the CZAR: Georgioupolis by touching the start button. If the user just presses the start button without any further interaction, the default options will be selected which are: 1) Humanoid indicator and 2) Dark theme: off.



### 5.3 Not-in-area Screen

After the welcome screen, the application will try to locate user's geographical location. In case the user is located outside the area of Georgioupoli's beach, the app will pop an alert message blocking any further interactions and forcing the user to exit the application as shown in figure 58. The user should be in a range of 2000 meters (2km) around the area of Georgioupoli's beach, so he/she can execute successfully the CZAR: Georgiopolis.



(a)



(b)

Figure 58: Not-in-area Screen (a) Light Map (b) Dark Map

## 5.4 Map Screens

### 5.4.1 Entering Map View

Once the initial steps have ended, the user will proceed to the map view. A note message will pop to him/her which gives useful information and tips on how to use the application properly (fig. 59).



Figure 59: Map Screen: Note

After dismissing the note, the user sees the map of the area, a self-indicator that shows the user's location on the map and probably 3 or less blue cones that indicate the locations the user should visit in order to witness the AR visualization. On the top of the screen there are information about the location that is selected and the distance of the user from there (fig. 60, 61). User can rotate the map around his/her position by dragging his/her finger on the screen.



Figure 60: Map Screen: Humanoid/Light Map



Figure 61: Map Screen: Pin Arrow/Dark Map

#### 5.4.2 Choosing Location

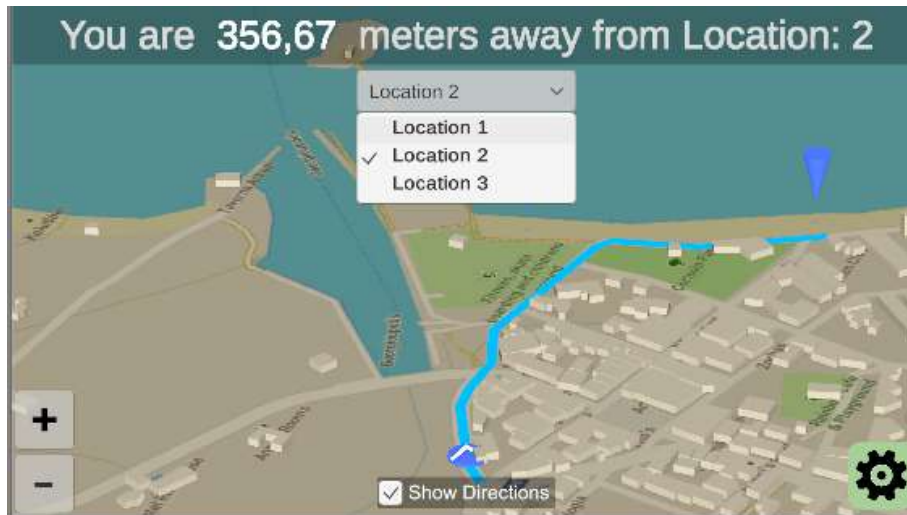
The locations are three. Location 1 indicates the first location the user finds from left to right when facing the North (the sea side on the map), Location 2 the second and Location 3 the third. In case the user cannot recognise which location is where, he/she can press the "Settings" button, which is down-right on the map, as shown in the above figures. When the user taps on the button, a drop-down list of the selections, a checkbox for directions and two buttons for zoom in/out will pop on the screen (fig. 62).



Figure 62: Map Screen: Map Settings

### 5.4.3 Navigating to Selected Location

By touching on the location window of the drop-down list, the list will expand letting the user to select a new location. The user can also enable the directions on the map by checking the box at the middle bottom of the screen (fig. 63a). All the changes and selections will persist even after the settings have been hidden by pressing again the "Settings" button (fig. 63b).



(a)



(b)

Figure 63: Map Screen: (a) Select Location & Enable Directions (b) Hide Settings

### 5.4.4 Reaching Selected Location

When the user approaches the selected location within 10 meters range, then the "AR" button will be enabled on the screen as shown highlighted in figure 64, above.

The button stays enabled for as long as the user is within 10 meters range of the location that he/she has selected before. By touching the button the AR experience begins.

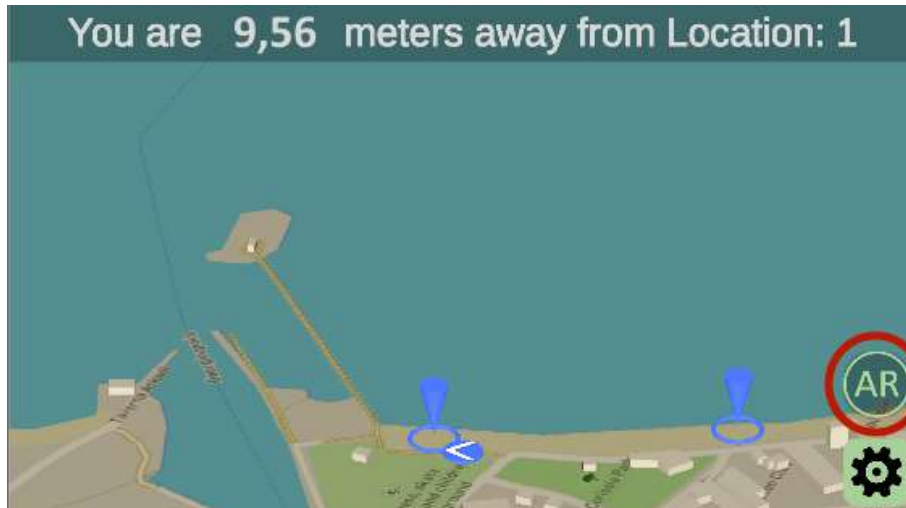


Figure 64: Map Screen: AR Button

## 5.5 AR Screens

### 5.5.1 Entering AR View

The AR experience begins with another note to the user. This note acts as a helpful tool-tip, giving advice and directions to the user in order to experience the AR visualization as good as possible. On top of the note, the selected location is stated (fig. 65). The message and the buttons are the same for all three locations.

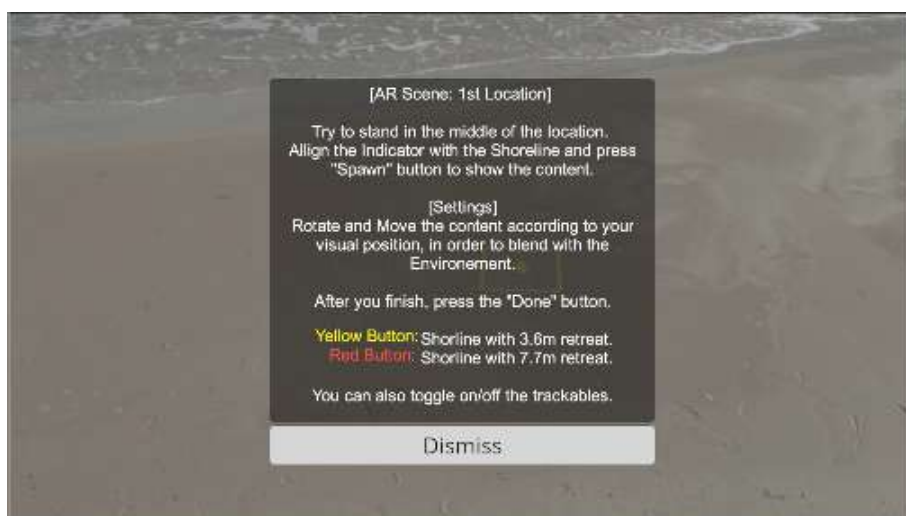


Figure 65: AR Screen: Note



### 5.5.2 Scanning the Environment

After dismissing the helpful note, the scanning process will begin. The application will ask the user to scan the ground (fig. 66). Scanning and finding a planar surface or ground is essential at this point in order to proceed further into the AR visualization.



Figure 66: AR Screen: Scanning Initiation

### 5.5.3 Detecting Planes & Shoreline

When the user has successfully detected the ground, an arrow indicator will be shown on top of that. While scanning, the detected surfaces will be highlighted with grey dots in order to help the user identifies the area he has scanned. While scanning, the "Place" button will be activated (fig. 67). Pressing this button, the AR content will be instantiated on top of the real world to the position and the rotation the arrow is indicating.

The User can align the arrow indicator with the shoreline (fig. 68) and press the "Place" button to bring the virtual sea in life. The position of the user and the indicator is crucial for successful placement.





Figure 67: AR Screen: Ground Detected



Figure 68: AR Screen: Indicating Shoreline

#### 5.5.4 Placing Virtual Content

Once the button is pressed, the virtual sea will be instantiated on top of the real one while new options/buttons will be enabled on the screen (fig. 69). The virtual content is an area of 55-60 meters length (based on the location). The virtual sea should match the real one in terms of shoreline for those meters only, the rest of the real sea will be visible as it is (will be shown later in this section).

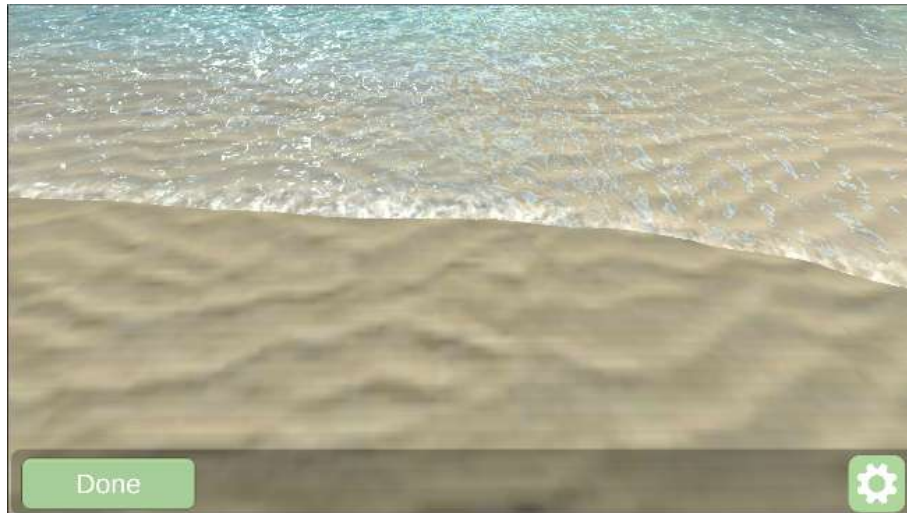


Figure 69: AR Screen: AR Content on top of Real World

When the AR Content has been instantiated, the user can use the "Settings" button to open a calibration menu. The calibration menu can be used in order to move or rotate the virtual content accordingly, in order to make it blend with the real environment as best as possible. The better the calibration the better the AR experience will be. When the user decides that the virtual content is aligned correctly, he/she can press the "Done" button to proceed to the AR visualization (fig. 70).

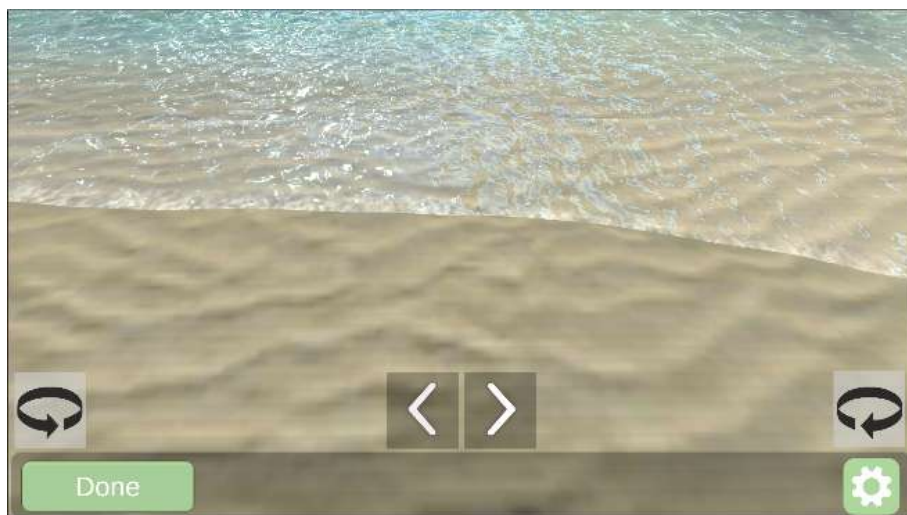


Figure 70: AR Screen: Calibration Menu

### 5.5.5 AR visualization

After the calibration has ended and the user proceeds to the AR visualization he/she will be able to 1) visualize future changes of the beach in AR at the current location and 2) receive some important information about the changes he/she is witnessing.

In the figure 71a the initial screen of the AR visualization is presented, and in the figure 71b (on the right) the final result after the user has seen both scenarios. The virtual content has been put on top of the real world and the user can clearly see the difference on the shoreline comparing those two figures. Below, the whole procedure has been separated into smaller parts, providing the final screen and the options the user has during the AR visualization.

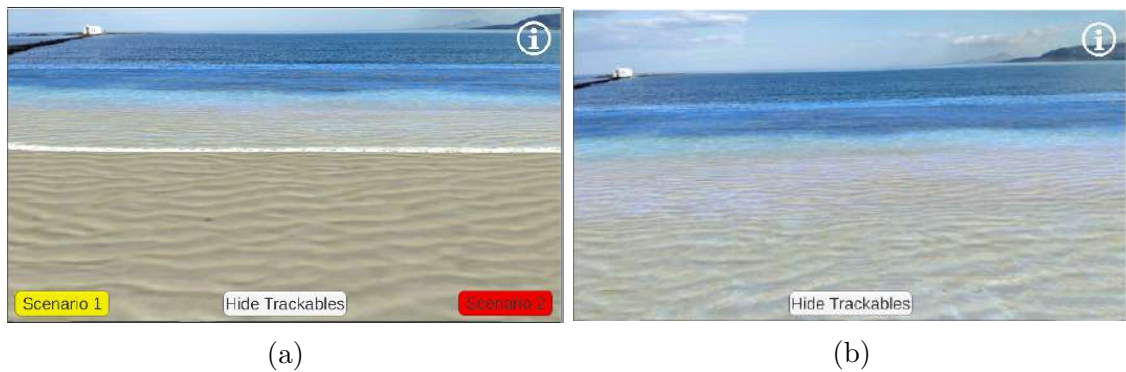


Figure 71: AR Screen: (a) Before (b) After

### 5.5.6 Beginning Visualization

Initially, the user has placed the virtual content and calibrate it accordingly. In figure 72, the virtual content has been placed and the virtual shoreline is aligned with the real shoreline of the beach in its current state. The user sees 3 buttons. A yellow button for visualization of Scenario 1 ( $SLR = 0.5m$  & mean retreat  $3.6m$ ), a red button for Scenario 2 ( $SLR = 1m$  & mean retreat  $7.7m$ ) and a white button for hiding the trackables that we mention before (scanned planes visualizer-grey dots). Finally, in the upper right corner there is an "info" button.



Figure 72: AR Screen: Current State (Close Distance)

#### 5.5.7 First AR Scenario

Pressing the yellow button, the user witnesses the first scenario which is the future state of the beach with shoreline retreat of 3.6 meters inland. This is the erosion the beach is expected to develop with sea level rise (SLR) of 0.5 meter. After this, the button will be disabled (fig. 73).



Figure 73: AR Screen: 3.6m Retreat (Medium Distance)



### 5.5.8 Second AR Scenario

Pressing the red button, the user witnesses the second scenario which is the future state of the beach with shoreline retreat of 7.7 meters inland. This is the erosion the beach is expected to develop with sea level rise (SLR) of 1 meter. After this, the button will be disabled (fig. 74).



Figure 74: AR Screen: 7.7m Retreat (Far Distance)

### 5.5.9 Info Signs Enabled

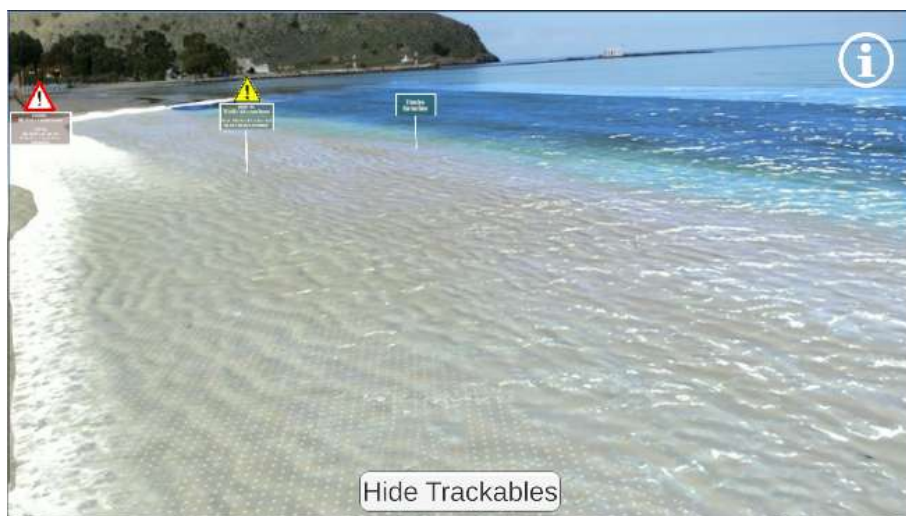


Figure 75: AR Screen: Info Signs

At any of the states of the visualization, the user can press the "info" button on the top right. Pressing this button, 3 signs will pop on top of the virtual sea that will indicate the distance the shoreline has been moved inland. Green sign: current state, yellow sign: shoreline with 3.6m retreat and red sign: shoreline with 7.7m retreat. The signs will pop in the middle of the virtual sea (fig. 75). User can disable the signs by pressing again the "info" button.

The user can move close to the signs as they are stable in the scene. At first sight, the signs give some information to the user about the shoreline state and the potential threats that each scenario holds (fig. 76).

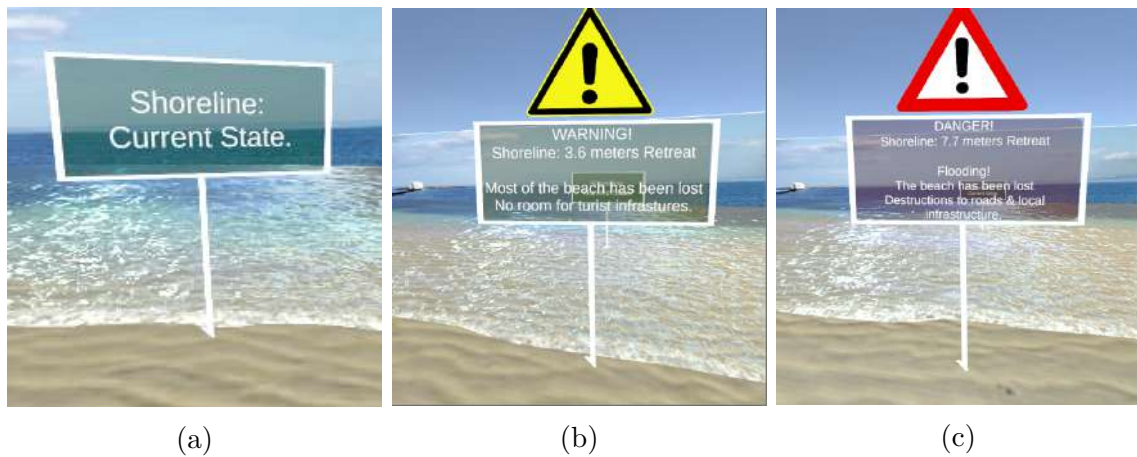


Figure 76: AR Screen: Signs, (a) Current Shoreline, (b) Retreat 3.6m, (c) Retreat 7.7m

### 5.5.10 Information Panels

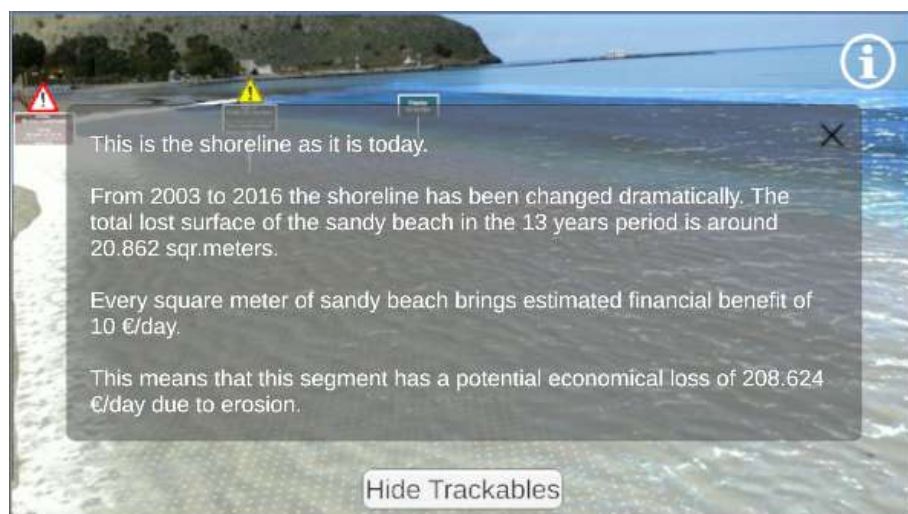


Figure 77: AR Screen: Current Shoreline, Info Panel



By tapping on one of the three signs, an info panel pops up on the screen giving useful information about the beach and the scenarios. By tapping the first sign 124a, the following panel will pop (fig. 77) giving info about the past of the beach.

Tapping on the other two signs, the user gets information about the scenarios that he has witnessed. All the info panels can be closed by pressing the "X" button on the top right of the panel and they can be reopened by tapping on the signs again. The information that user sees for both scenarios, are shown in the figure 78 below.



(a)



(b)

Figure 78: AR Screen: Info Panels, (a) Retreat 3.6m, (b) Retreat 7.7m

That's the whole experience from the user's point of view. Users can relive all the experiences for all the three locations on the map. The information on the panels is about the whole beach of Georgioupoli (built-part 0.5km, whole beach 5km) and

not on the specific part the user has lived the visualization.

## **5.6 Usage of the App**

In this section, there are some additional useful information about the application CZAR: Georgioupolis. How to use it effectively, what the expectations of a candidate user should be or what consequences will have the non recommended usage of the app.

### **5.6.1 Giving Access to Hardware**

To begin with, the first time the application will be launched, the user will be asked to give access to the application on the GPS of the device and the Camera. The GPS is used during the map view of the app and it is needed in order to find the user's location and to calculate accurately his/her distance from the selected location that has been selected. The camera is used only on the AR scene of the app and it is needed in order to experience the AR visualization. The camera gives us the feed of the real world and allows the device to scan the real world in order to detect plane surfaces and feature points in order to create a map of the world that it is facing. Not giving access to the GPS will result in inaccurate calculations of the distance and the user's location, and not giving access to the camera practically it kills the app and it will not work properly and the way it was intended.

Another important aspect is that the device should have internet connection. Connecting the device to the internet allows the device to be located even more accurately in combination with the GPS but more importantly, the device can download the maps for the map view of the application on demand. Not having connection to the internet will not prevent you from using the application but the functionality will not meet the requirements and it will not be pleasant. The map will not be downloaded and practically it will be impossible for the user to navigate around the area.

After this initial process, the application will be ready to be launched. It might not ask you again for permission after the first time. This depends on the personal permission settings of the user's device.

### **5.6.2 Helpful Tips**

In proceeding, the user can use the application in both portrait and landscape mode. In the portrait mode the UI on-screen buttons will be a little smaller since the resolution of the phone changes and becomes smaller on the top and bottom of the device. Another reason for smaller buttons is that the field of view in portrait mode is limited both in map and AR view. It is recommended to use the application in

landscape mode for better visual experience and easier usage in general (especially on smartphone devices). In the figure 79 below you can see the difference as it's clear that the field of view is wider in landscape mode in AR. This specific figure is on a device with 1920 x 1080 resolution (16:9), this might differ in devices with different aspect ratio and resolution.

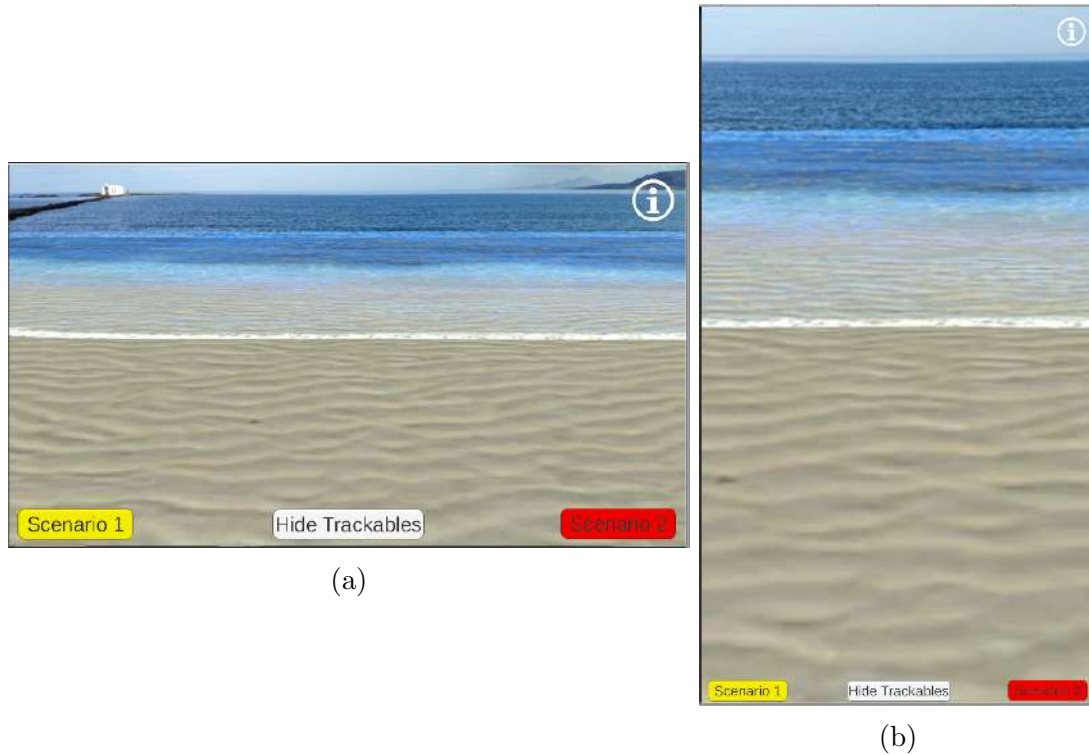


Figure 79: Orientation: (a) Landscape (b) Portrait

Calibration and proper instantiating of the virtual content is key for a nice experience. On every location there are key objects that will help align the virtual world with the real one. Try to use those as a guide for your calibration phase. Since there is no option on moving forward or backwards the virtual content, make sure you have indicated the shoreline as close as possible. If you fail achieving that, just press the back button to re-launch the AR experience from the map view. Improper alignment will result in poor experience and the illusion of blending real and virtual world will be lost.

Closing, we have to mention that the weather phenomena and the sea condition will effect your experience. It is recommended to try using the app on sunny days with low to non wavy sea as the good lighting is key for mapping the real world and calm waters help on better scanning of the beach finding more feature points. All the visualization you are about to witness is based on the shoreline that the beach tends to have and not on extreme conditions and phenomena.

## 6 Implementation

In this section, we analyse our application from the point of view of a programmer/developer. We mention all the software we used in the developing phase and how we integrate them in our app. We breakdown the development process and we describe the back-end development of the app in details.

### 6.1 Developing Platform

#### 6.1.1 Unity3D

CZAR: Georgioupolis was fully developed in Unity3D Game Engine. The version we used was Unity 2019.2 (fig. 80). This version was selected as all the software and packages we wanted to use had as requirement Unity 2019.1 versions and later. 2019.2 was the latest version we could use after the developing process had began. Further upgrade wasn't necessary since there wasn't any improvement on the tools we already used. The basics of Unity Editor and UI elements will not be explained in this thesis, and the reader should have an overall knowledge of the Unity's UI and some programmatic terms.

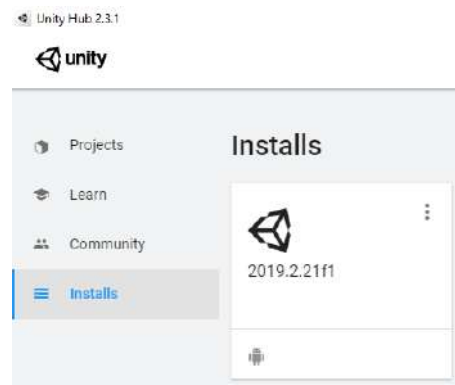


Figure 80: Unity Version

#### 6.1.2 Android Studio

Since we need to compile our application for Android smartphones we also used Android Studio 3.2. It was used by Unity until the end of the development process without any problem, importing the essential packages in order to build in devices with android 7.0+ which is the minimum requirements for the ARCore SDK. Other than this, there was no problem on building even on Android 10 which was realised

on our smartphone in early 2020.

### 6.1.3 Hardware

Initially the application was developed on a laptop with CPU: Intel Core i7 3632QM @ 2.20GHz, GPU: AMD Radeon HD 7600M Series, RAM: 8GB DDR3 and SSD: Samsung 840 EVO. This hardware was enough for the requirements of the project and the tools we used but the rendering and building process has taken enough time.

Later, the development was migrated to a better PC with: Intel Core i7 8700k @ 3.70GHz, GPU: nVidia GTX 1050ti, RAM: 16GB DDR4 and SSD: Samsung 860 EVO. This hardware was far more superior than the previous one, accelerating the development process by far. The rendering was much faster because of the better CPU and GPU.

## 6.2 Software & Packages

Other software and Unity packages we used and they were essential on the development of the application are listed below. Also, we give a small description on how we implemented them into our Unity project.

### 6.2.1 ARCore

In order the application to run on a smartphone device, ARCore should be installed on the targeted device. On our testing device we had the latest version ARCore 1.16 which was released on March 10, 2020. ARCore can be installed very simply by Google Play Store. Installing Google ARCore on a device will allow it to run all the application that have been developed on top of the ARCore's SDK functionality. The ARCore library is available on Google Play Store only on the supported devices (fig. [81](#)).

Anyone can build an application on top of Google's ARCore SDK on the supported platforms. At Google's ARCore documentation website [\[28\]](#) can someone download the SDK and find useful guides. In our thesis though, we didn't build on top of ARCore directly but we used Unity's Package AR Foundation as we mention in section [2.6.3](#). AR Foundation will be explained later in this section.

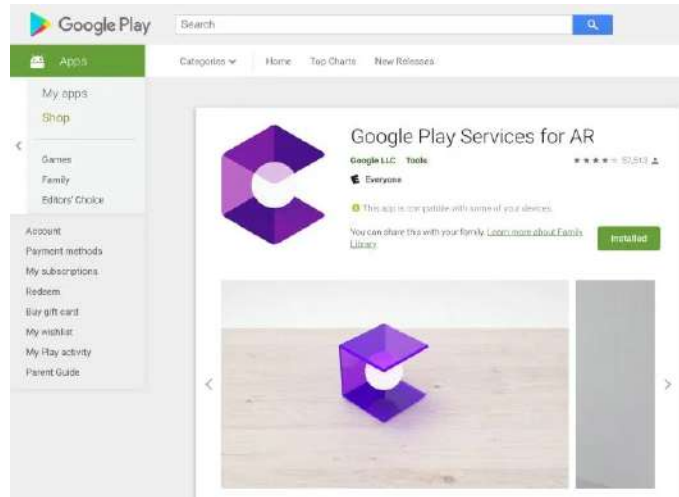


Figure 81: Google Play Store: ARCore

## 6.2.2 Mapbox

The Mapbox Maps SDK for Unity is a collection of tools for building Unity applications from real map data. It enables Unity developers to interact with Mapbox web services APIs (including the Maps, Geocoding and Directions APIs) and create game objects via a C#-based API and graphical user interface. In our thesis we used Mapbox SDK for the creation of map in map view of the application. Mapbox is also tested for using in an AR Navigation but unfortunately the tests showed that GPS inaccuracy and the failure of correct alignment was disturbing and it cannot be used for such purpose.

In our project we used Mapbox SDK 2.0 for Unity. The latest version so far is Mapbox 2.1.1 with minor changes from the version we used, so we didn't proceed to update to prevent errors since we were totally satisfied with the functionality so far.

Mapbox SDK can be downloaded in Mapbox's site for Unity [52]. In order to use the SDK, we need an account to get an API key. After getting the API key, the downloaded package should be imported into Unity (fig. 82).



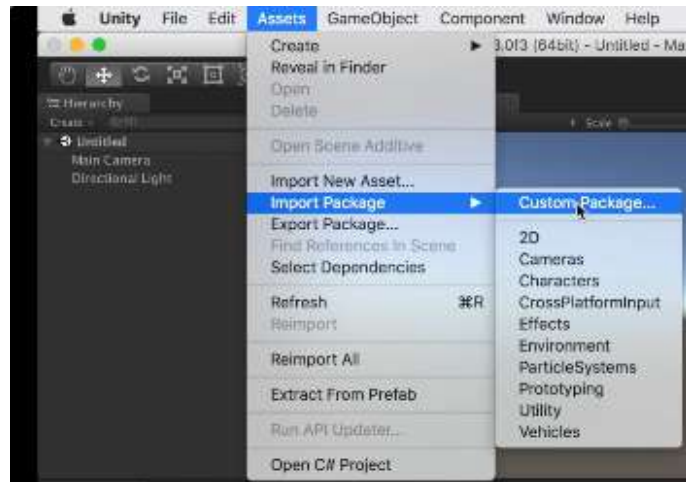


Figure 82: Importing Packages in Unity [52]

After importing the package into Unity, we will be asked to insert our personal API key in order to activate Mapbox functionality (fig. 83). When the API key is inserted, the SDK will be activated and all the given examples and tools will be available to the developer to use them in his/her project.

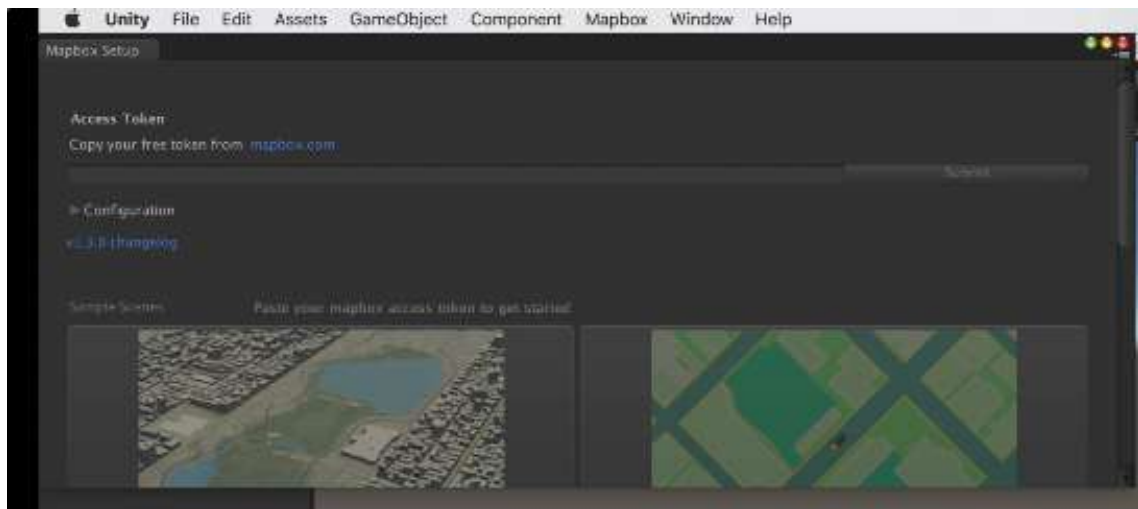


Figure 83: Inserting Mapbox API key in Unity [52]

Mapbox is free for the first 50.000 users which is more than enough for the use we were about to do with it.

### 6.2.3 AR Foundation

AR Foundation allows us to work with augmented reality platforms in a multi-platform way within Unity. This package presents an interface for Unity developers

to use, but doesn't implement any AR features itself. To use AR Foundation on a target device, you also need a separate package for that platform (for example, ARKit XR Plugin on iOS or ARCore XR Plugin on Android) [34].

AR Foundation is a set of 'MonoBehaviours' (Unity scripts) and APIs for dealing with devices that support the following concepts:

- World tracking: track the device's position and orientation in physical space.
- Plane detection: detect horizontal and vertical surfaces.
- Point clouds, also known as feature points.
- Anchor: an arbitrary position and orientation that the device tracks.
- Light estimation: estimates for average color temperature and brightness in physical space.
- Environment probe: a means for generating a cube map to represent a particular area of the physical environment.
- Face tracking: detect and track human faces.
- Image tracking: detect and track 2D images.

AR Foundation package and all the essential packages alongside with it, can be imported within Unity's package manager (fig. 84).

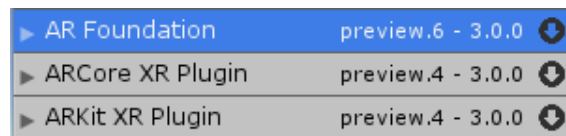


Figure 84: Essential Packages in Unity's Package Manager

Importing these packages allows us to build applications on top of ARCore and ARKit SDKs, without focusing on a specific platform. We imported all the core functionalities of ARCore and ARKit SDKs by importing ARCore/ARKit XR Plugins and that's all we need to import in order to implement ARCore/ARKit in our project. The project has been built using AR Foundation 3.0.0. This version allowed us to use all the core functionalities of ARCore/ARKit we need and it's compatible with the new Lightweight Render Pipeline (LWRP) for Unity. We will discuss more on LWRP later in this section. Further update on AR Foundation caused errors in our projects due to some core changes, providing no further capabilities in our project. More guides and examples can someone find at AR Foundation's documentation page [34], which helped us a lot during the development phase.

#### 6.2.4 Lightweight Render Pipeline

The Lightweight Render Pipeline (LWRP) is a prebuilt Scriptable Render Pipeline, made by Unity. The technology offers graphics that are scalable to mobile platforms, and can also be used for higher-end consoles and PCs. With LWRP we are able to

achieve quick rendering at a high quality without needing compute shader technology. LWRP uses simplified, physically based Lighting and Materials.

The LWRP uses single-pass forward rendering. By using this pipeline we get optimized real-time performance on several platforms. The LWRP is supported on the following platforms:

- Windows and UWP
- Mac and iOS
- Android
- Xbox One
- PlayStation 4
- Nintendo Switch
- All current VR platforms

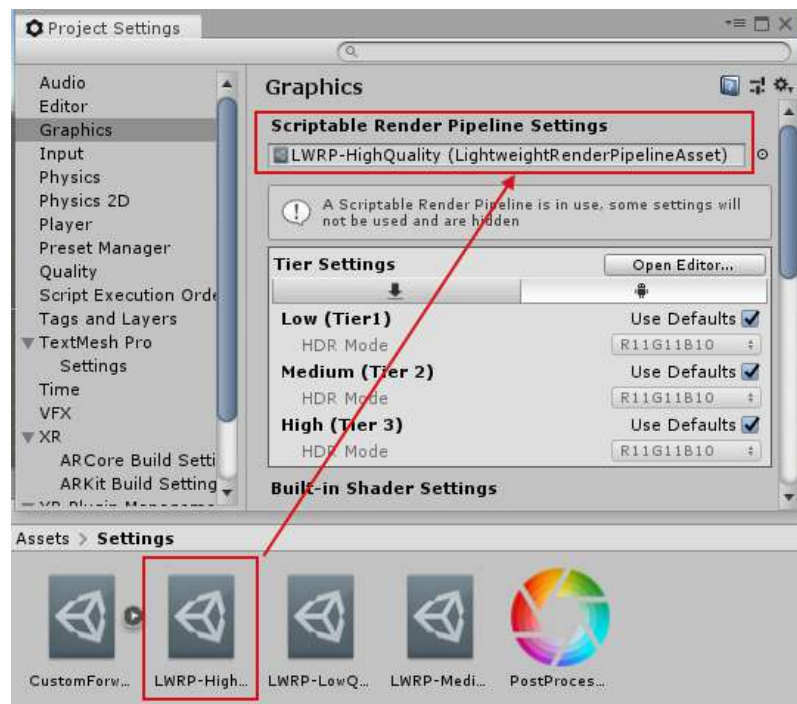


Figure 85: LWRP Asset Assignment

To use the Lightweight Render Pipeline (LWRP), we have to import it via Unity's package manager, then we have to create a LWRP Asset and assign the asset in the Graphics settings. We can have multiple LWRP assets and switch between them (fig. 85).

The LWRP Asset controls several graphical features and quality settings for the Lightweight Render Pipeline. It is a scriptable object that inherits from 'RenderPipelineAsset'. When we assign the asset in the Graphics settings, Unity switches from the built-in render pipeline to the LWRP. We can then adjust the corresponding settings directly in the LWRP, instead of looking for them elsewhere (fig. 86).

LWRP became compatible on September 2019 using Unity 2019.2 with AR Foundation version 3.0. This was a huge leap in the development of our app. Better graphics were achieved using less resources and finally we had access to Shader Graph (see next section). Documentation site: [\[53\]](#)

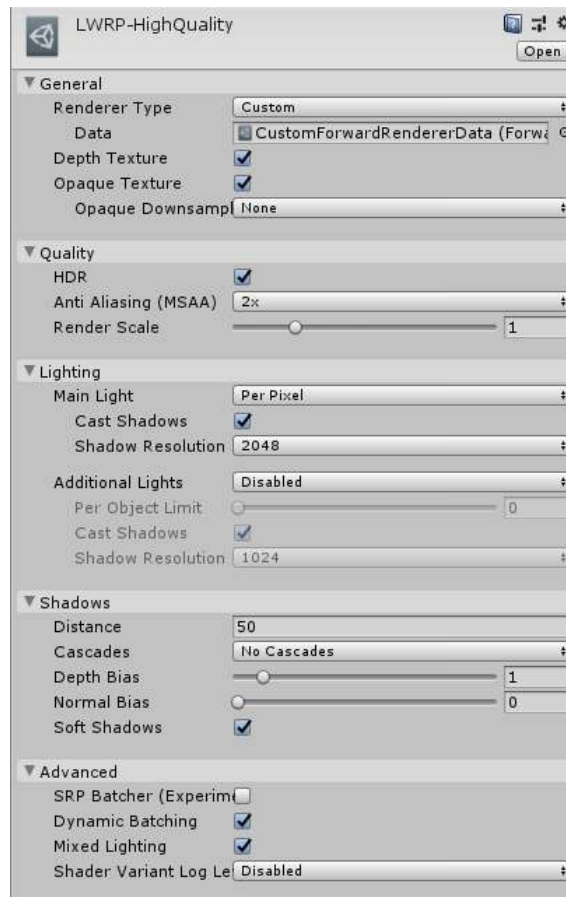


Figure 86: LWRP Settings

### 6.2.5 Shader Graph

Shader Graph enables the building of shaders visually (fig. 87). Instead of writing code, we create and connect nodes in a graph framework. Shader Graph gives instant feedback that reflects the developer's changes, and it's simple enough for users who are new to shader creation. In our project, it helped a lot on the creation of a water shader that was rendered successfully on a mobile device (we will analyze later the creation of our water shader).

Shader Graph is available through the Package Manger window in Unity versions 2018.1 and higher. Shader Graph comes included with the HDRP and LWRP packages. When you add either SRP (HDRP or LWRP) to your project, Unity

automatically loads Shader Graph and installs it in the project. It is recommended to avoid installing or updating Shader Graph independently of the prebuilt SRP packages. Shader Graph builds shaders that are compatible with the SRP, but they are not compatible with the built-in renderer.

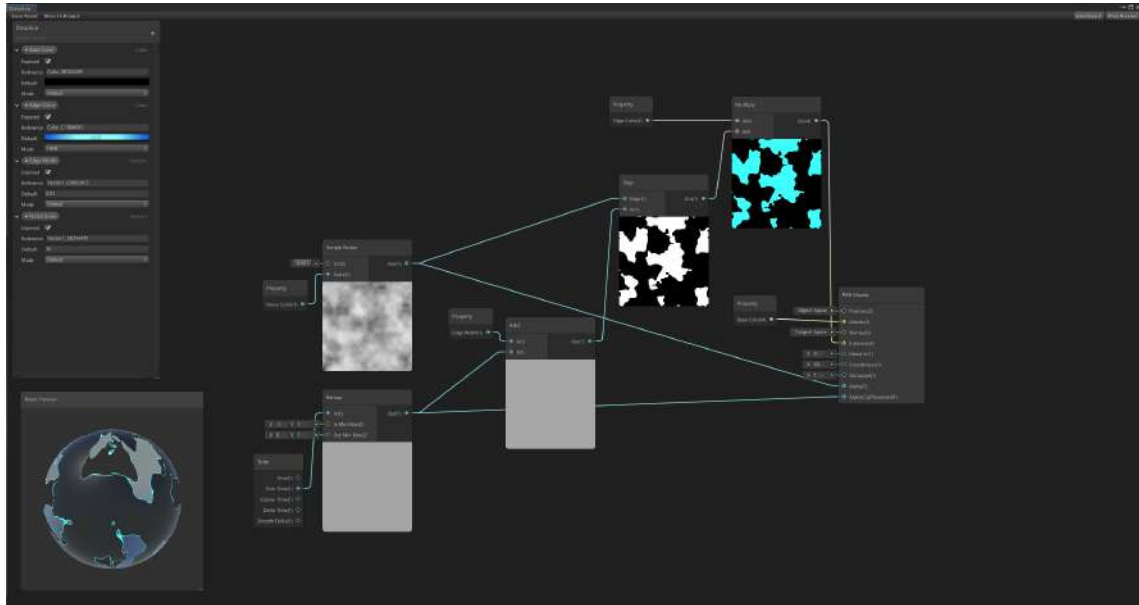


Figure 87: Shader Graph

Shader Graph package versions on Unity Engine 2018.x are Preview versions, which do not receive bug fixes and feature maintenance. To work with an actively supported version of Shader Graph, use Unity Engine 2019.1 or higher. More info on Shader Gragh documentation site [\[54\]](#).

## 6.2.6 Rest of Packages

After the analysis of the most important software and packages in our project, here we have listed the rest of the packages that were imported in our project. Most of the following packages were not essential for the development of our application and some of them were imported automatically by Unity during the creation of the project or because it was essential for the functionality of the previously mentioned packages (e.g. Core RP Library). Below, all the imported packages from Unity's package manager and the versions we used (fig. [88](#)).

Some useful packages are:

- Unity UI [\[57\]](#) is a UI toolkit for developing user interfaces for games and applications. It is a GameObject-based UI system that uses Components and the Game View to arrange, position, and style user interfaces. You cannot use Unity UI to create or change user interfaces in the Unity Editor.

- The Terrain Tools [55] package helps improve the workflow for creating Terrain in Unity. The Unity 2019.1 package contains brand new sculpting Brushes, and a collection of utilities and tools to help automate tedious tasks. In the Unity 2019.2 package, new Brush Mask Filters were added to enhance sculpting, and Material painting tools to help us achieve beautiful results on Terrain (used in the terrain creation for our AR scenes).
- ProBuilder [56] helps to build, edit, and texture custom geometry in Unity with the tools available in the ProBuilder package. You can also use ProBuilder to help with in-scene level design, prototyping, collision Meshes, and play-testing. ProBuilder also comes with a Scripting API, so that you can write C# scripts to make your own tools and customization (great package for modeling basic geometry without the usage of a 3rd party software).

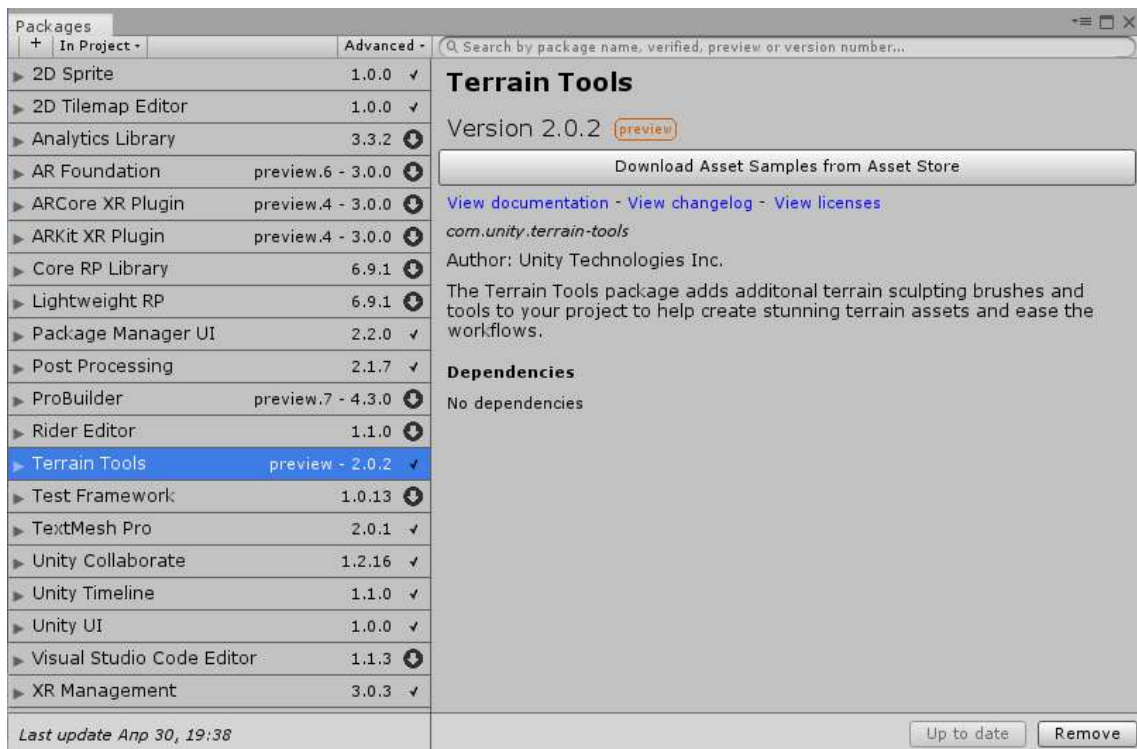


Figure 88: Imported Packages

## 6.3 Structure of the App

In this section, we show the basic structure of the application in Unity game engine and the parts each scene contains and how they are connected together.

In Figure 89, there are the scenes that have been build in Unity, and as you can see we have turned our platform to Android. These six are the scenes of the



application but the Map Scenes are basically the same, with the only difference of the map color and the three AR scenes differ only on the virtual content while all AR scenes share the exact same functionality. The order of the scenes in the build settings are specific for the correct loading of each one. For example First AR scene is the number 1 in the build setting as it is the first selection on the dropdown list in Choosing Location [5.4.2].

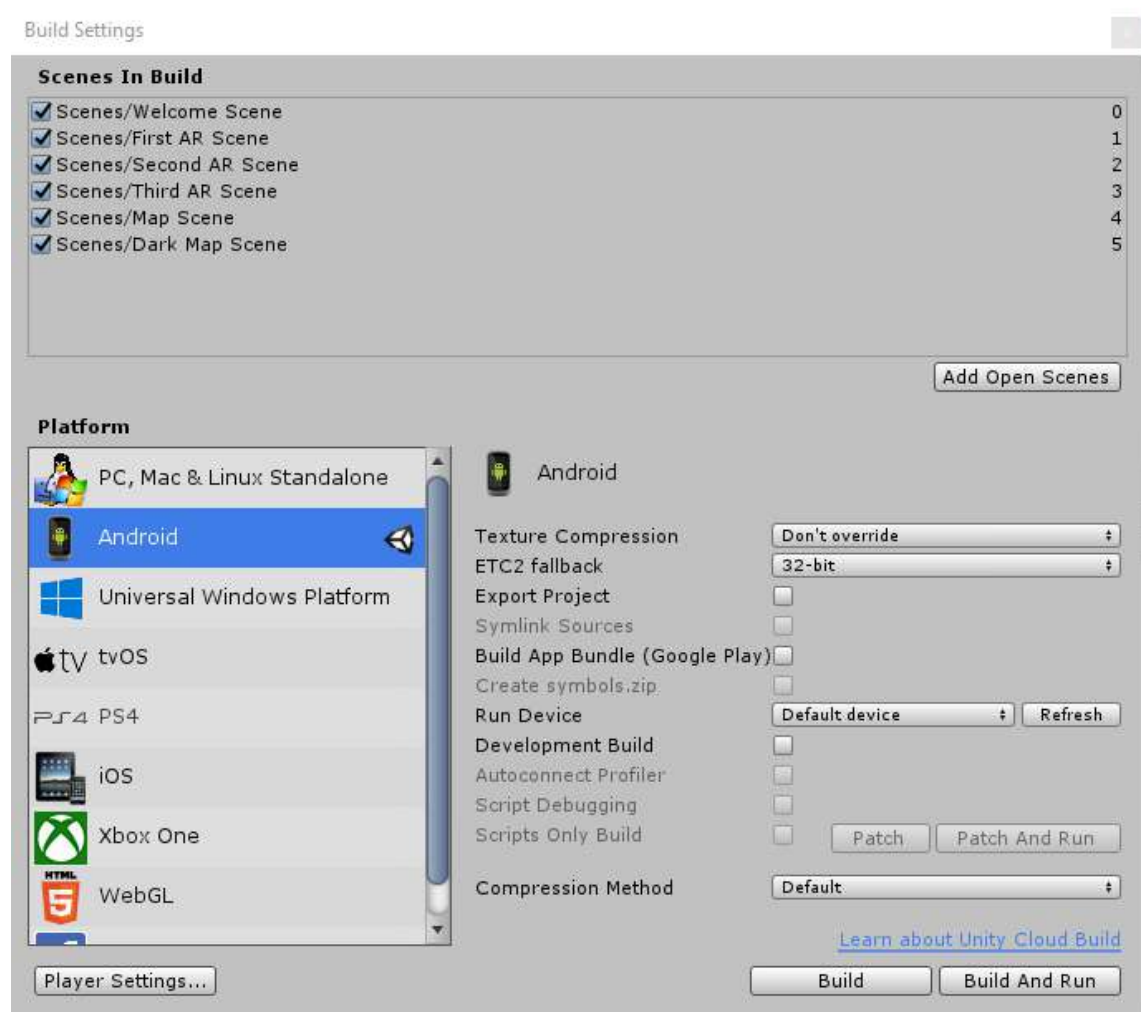


Figure 89: Scenes of the Project

Below in figure 90, the structure of the project is displayed and how those six scenes are connected with each other. The application launches on the welcome screen, then based on the user's preferences, one of the the map scenes is loaded. This will be the map scene for the rest of the session. The AR Scenes are loaded only when the AR button (that mentioned in User Experience) press. Which AR scene will be loaded depends on user location. First AR Scene is for the first location, second AR scene for the second location and third AR Scene for the third location as they are described previously on this thesis.

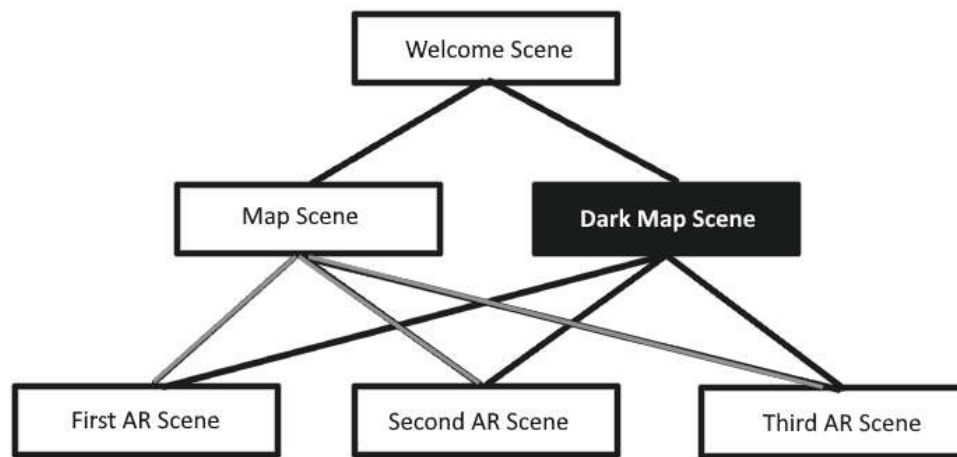


Figure 90: Structure of the Project

In the rest of this section, we make an in-depth description of the development of each scene and how each package was integrated into the scenes.

## 6.4 Welcome Scene

The initial scene the user sees. It's a simple scene which was designed in order to introduce smoothly the user into the application and saves the user's preferences for the rest of the session.

### 6.4.1 Welcome Scene Components

In figure 91 are all the components the scene has in Unity editor.

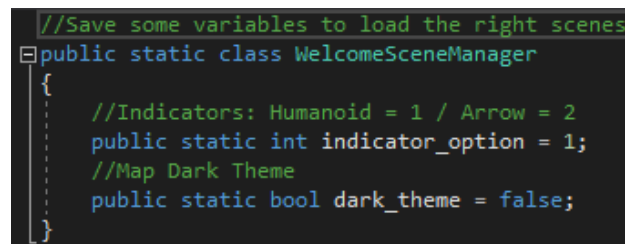


Figure 91: Welcome Scene Components in Unity Editor

The scene includes only UI elements, mainly canvases, texts, images and buttons. There are also: 1) Camera object for the rendering of the scene. Every scene requires at least one Main Camera in order to be rendered. 2) The 'EventSystem' is responsible for processing and handling events in a Unity scene. A scene should only contain one 'EventSystem'. The 'EventSystem' works in conjunction with a number of modules and mostly just holds state and delegates functionality to specific, over-rideable components. This object is generated automatically. 3) An 'Application Quit' empty game object which contains only one script for the termination of the app when the 'back' button on the device has been pressed.

### 6.4.2 Interactions

The user can interact only with the toggle boxes (on/off) and the 'Start' button. The preferences of the user should be saved during the whole session of the application, so we had to save them for further usage on the next scenes. To do that, the 'WelcomeSceneManager' created, a public static script. This kind of scripts are accessible from all the scenes in our project, compared with 'MonoBehaviour' scripts that should be attached to game objects and work only on the objects' scene locally. In figure 92 the script and the two preferences of the user we save: 1) Indicator and 2) Map Theme.

A screenshot of a code editor showing the 'WelcomeSceneManager' script. The code is written in C# and includes comments in green. The script is a public static class with a single method block containing two static variables: 'indicator\_option' and 'dark\_theme'.

```
//Save some variables to load the right scenes
public static class WelcomeSceneManager
{
    //Indicators: Humanoid = 1 / Arrow = 2
    public static int indicator_option = 1;
    //Map Dark Theme
    public static bool dark_theme = false;
}
```

Figure 92: Welcome Scene Manager Script

The user can only choose one of the two indicators and at any given time only one toggle box can be enabled. With any action there is a reaction, enabling/disabling the boxes accordingly and saving the option in 'WelcomeSceneManager'. By default, 'indicator\_option = 1' (humanoid) and 'dark\_theme = false'. The script that handles user's actions shown in figure 93.

Each function is called on value change of each toggle box, respectively. Update is called once per frame to handle the case in which user has not selected either of the indicators.

```

void Update()
{
    if (!humanoid_checked && !arrow_checked)
    {
        humanoid_toggle.GetComponent<Toggle>().isOn = true;
        arrow_toggle.GetComponent<Toggle>().isOn = false;
        humanoid_checked = true;
        arrow_checked = false;
        WelcomeSceneManager.indicator_option = 1;
    }
}

public void HumanoidToggleChange()
{
    humanoid_checked = !humanoid_checked;
    if (humanoid_checked)
    {
        humanoid_toggle.GetComponent<Toggle>().isOn = true;
        arrow_toggle.GetComponent<Toggle>().isOn = false;
        WelcomeSceneManager.indicator_option = 1;
    }
}

public void ArrowToggleChange()
{
    arrow_checked = !arrow_checked;
    if (arrow_checked)
    {
        humanoid_toggle.GetComponent<Toggle>().isOn = false;
        arrow_toggle.GetComponent<Toggle>().isOn = true;
        WelcomeSceneManager.indicator_option = 2;
    }
}

public void DarkThemeToggle()
{
    dark_theme_checked = !dark_theme_checked;
    WelcomeSceneManager.dark_theme = dark_theme_checked;
}

```

Figure 93: Toggles Handler Script

Finally, the 'Start' button enables a script for loading the right scene based on the user's selections (fig. 94).

```

public class StartButton : MonoBehaviour
{
    public void LoadMap()
    {
        if(WelcomeSceneManager.dark_theme)
            SceneManager.LoadScene("Dark Map Scene");
        else
            SceneManager.LoadScene("Map Scene");
    }
}

```

Figure 94: Start Button Script

## 6.5 Map Scene

The map scene acts as a guide to the user. It helps on the navigation and acts as the main scene that connects the three AR scenes together. There are two Map scenes in the project (as shown before), but only one is loaded each time during one session. The main and only difference of the two scenes, is the theme which is loaded on the map (fig. 95). For this reason, we will analyze only one of the two in this sections. All the elements and everything you are about to read is representative for both scenes.

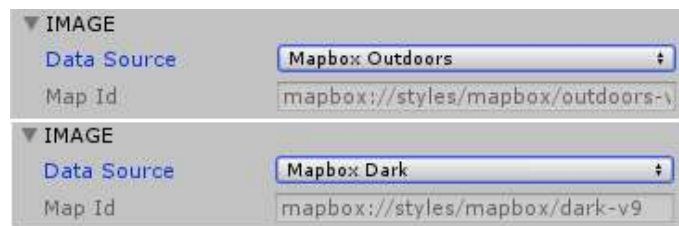


Figure 95: Map Theme: Light (Up), Dark (Down)

### 6.5.1 Map Scene Components

In figure 96 are all the components the scene has in Unity editor.

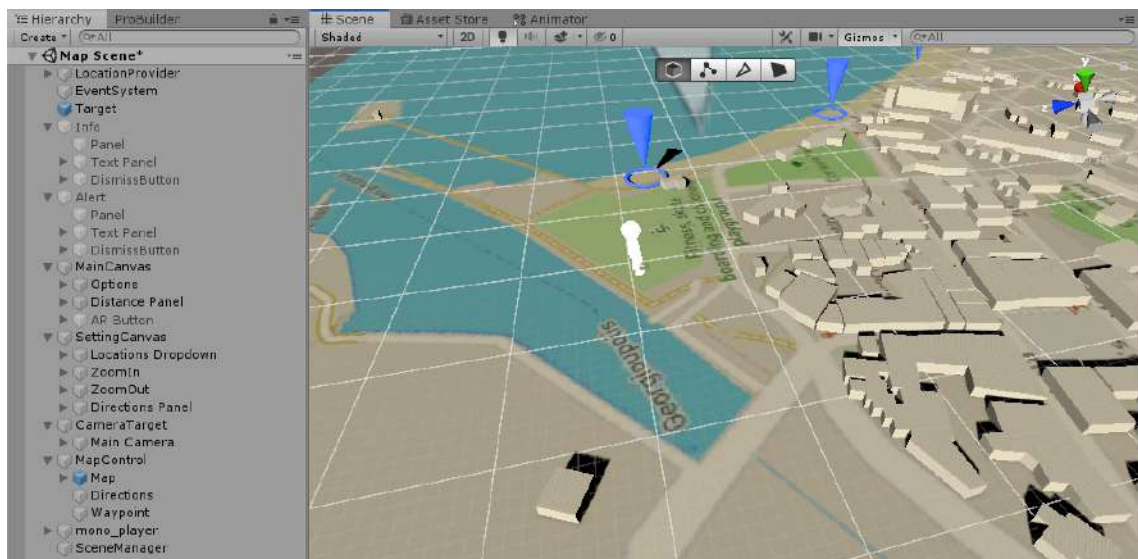


Figure 96: Map Scene Components in Unity Editor

The main components are:

- 4 UI Canvases: 1) 'Info', 2) 'Alert', 3) 'MainCanvas' and 4) 'SettingsCanvas'.
- One 'LocationProvider' object which give access to location providers of the device the app is running on.
- The 'EventSystem' which we mentioned earlier.
- One 'Target' object which synergizes with 'LocationProvider' to apply orientation and translation of movement in the app.
- The 'MapControl' object which includes the Map, the Directions and Waypoints (those two are used for providing directions to the user).
- The 'mono\_player' object, which is basically the player that represents the user on the map.
- The 'SceneManager' that controls mainly the functionality about the loading of AR Scenes.

### 6.5.2 Integrating Mapbox

Mapbox SDK is used on this scene. Here we analyse the integration of the mapbox's parts in the current scene enabling its functionality. All the following steps, are after you have imported the Mapbox package into Unity. All the resources that have been used were included in mapbox's package.

First, we have to import the 'LocationProvider' to enable access to location sensors of the device. 'LocationProvider' should include "LocationProvider'Factory' script in order to work as intended (fig. 97).

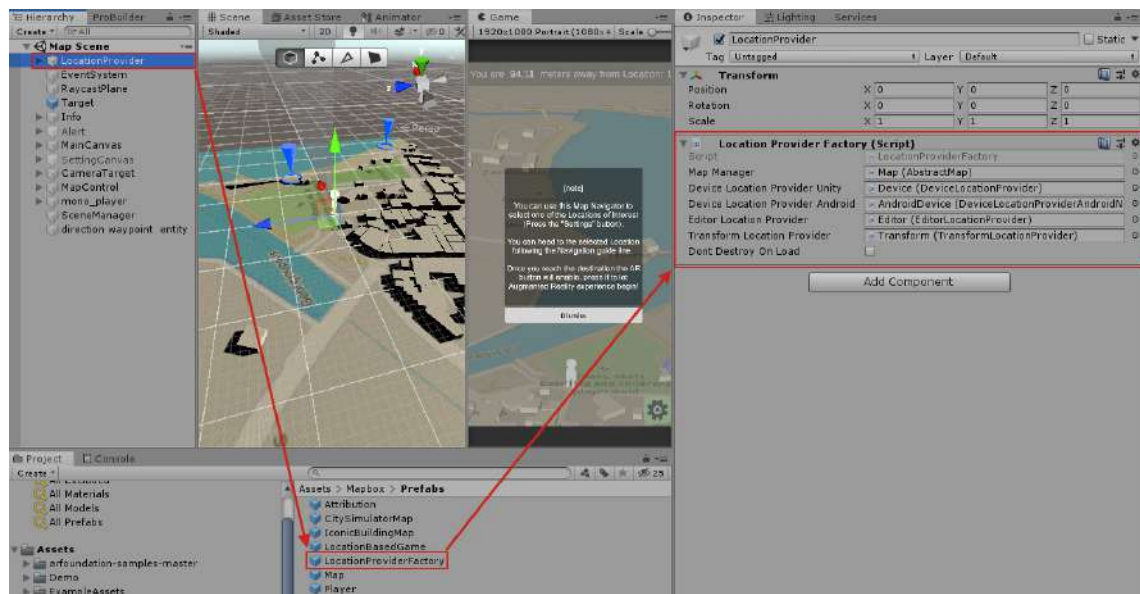


Figure 97: Location Provider

Since we have access to the location providers of the device, we need to make use of them. The 'Target' object has this role in our scene, which includes scripts given by Mapbox that help it synergize with 'LocationProvider'. The scripts attached on it are shown in figure 98. 'Target' object is instantiated accordingly based on the



'LocationProvider"'s data.



Figure 98: Target's Components

The final step is to include one 'Map' prefab in the scene. This creates the map that is shown in the scene. Map prefab includes two scripts: 1) 'Abstract Map' which is for the customization of the map and 2) 'Initialize Map with Location Provider' which is for the initialization of the map based on the Location of the user when using a device outside Unity's Editor (fig. 99).

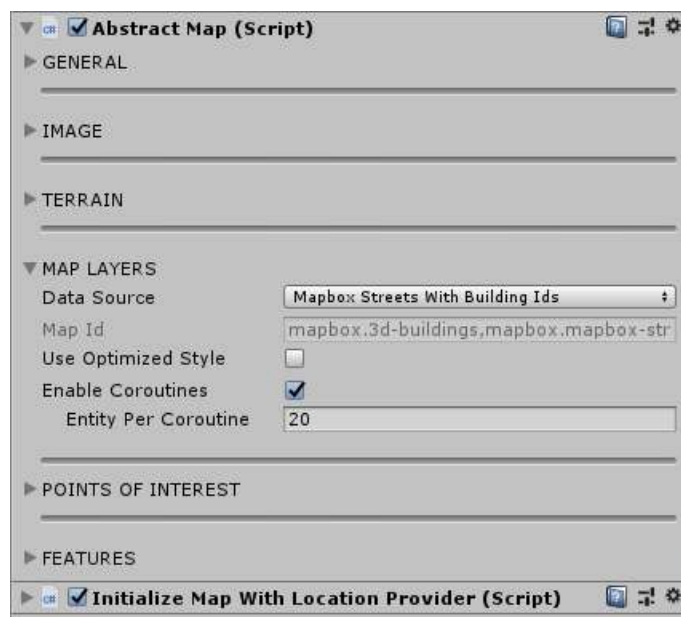


Figure 99: Map Components

After these steps, we have a map in our scene that is loaded based on the current location we are at. The next step was to indicate the locations we wanted on this map. The blue cone indicator on the map was placed using the 'AbstractMap' script which gives the option to place points of interest on specific geo-locations using any prefab we want. So, we had to put three points on the map, using the blue cone as prefab (fig. 100)

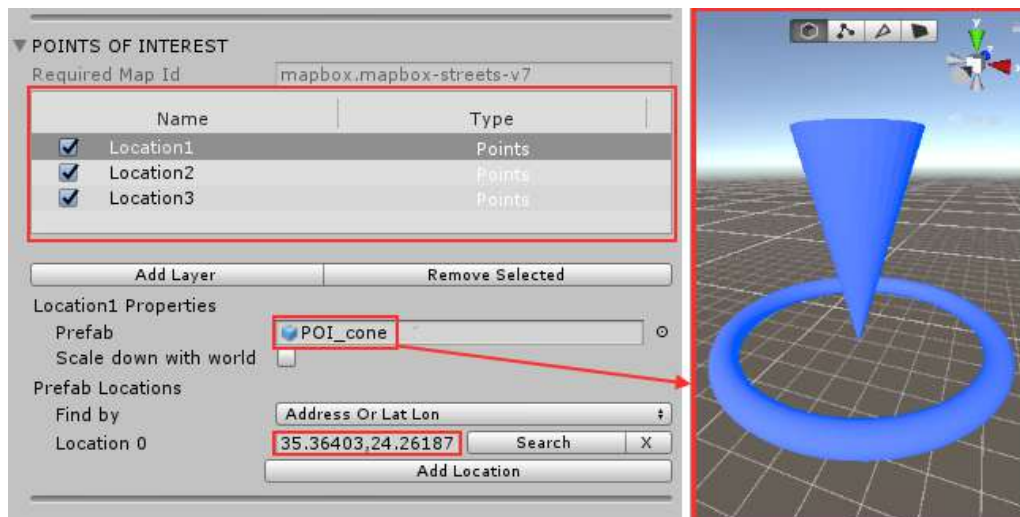


Figure 100: Inserting POIs on Map (Left), Cone Prefab (Right)

### 6.5.3 Self-Indicator Functionality

In the scene is loaded a self-indicator, which represents the user's position on the map. The object that controls the self-indicator is 'mono\_player'. The 'mono\_player' object is an empty object that includes a 'Character Movement' script about the instantiation's position of the player and his movement (fig. 102, 101). The script included in Mapbox, in the 'Astronaut Game' example scene. As children it contains two objects the 'Humanoid' and the 'Arrow' (fig. 103).

The player's location initially is the same as the 'Target' object we mentioned earlier, and it keeps following the 'Target' position which corresponds to the GPS changes that received from "LocationProvider". The player always looks at the 'Target' and with this the player is facing the direction the user moves. We also have set the movement speed of the player by testing different values and choosing the one that looks visually better. The animator in the script is responsible for the walking animation of the humanoid.



Figure 101: Character Movement Component

```

void Update()
{
    foreach (var item in Materials)
    {
        item.SetVector("_CharacterPosition", transform.position);
    }

    var distance = Vector3.Distance(transform.position, Target.position);
    if (distance > 0.1f)
    {
        transform.LookAt(Target.position);
        transform.Translate(Vector3.forward * Speed);
        CharacterAnimator.SetBool("IsWalking", true);
    }
    else
    {
        CharacterAnimator.SetBool("IsWalking", false);
    }
}

```

Figure 102: Character Movement Script

The Humanoid 3D model was downloaded for free by turbosquid.com [58] including the animation, and the arrow model created using Unity's basic 3D model shapes. During the loading phase of the Map Scene, and based on the option that has been saved in 'WelcomeSceneManager', one of the two children-objects is disabled.

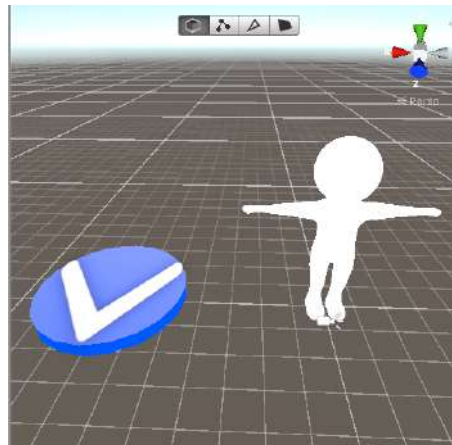


Figure 103: Indicator Objects, Arrow (Left), Humanoid (Right)

#### 6.5.4 Directions Implementation

In order to provide directions to the user, we made use of Mapbox's Directions API by using the 'Directions Factory' component script. This allows our application to request directions from Mapbox.

```

public void Start()
{
    _cachedWaypoints = new List<Vector3>(_waypoints.Length);
    foreach (var item in _waypoints)
    {
        _cachedWaypoints.Add(item.position);
    }
    _recalculateNext = false;

    foreach (var modifier in MeshModifiers)
    {
        modifier.Initialize();
    }

    StartCoroutine(QueryTimer());
}

void Query()
{
    var count = _waypoints.Length;
    var wp = new Vector2d[count];
    for (int i = 0; i < count; i++)
    {
        wp[i] = _waypoints[i].GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale);
    }
    var _directionResource = new DirectionResource(wp, RoutingProfile.Walking);
    _directionResource.Steps = true;
    _directions.Query(_directionResource, HandleDirectionsResponse);
}

public IEnumerator QueryTimer()
{
    while (true)
    {
        yield return new WaitForSeconds(UpdateFrequency);
        for (int i = 0; i < _waypoints.Length; i++)
        {
            if (_waypoints[i].position != _cachedWaypoints[i])
            {
                _recalculateNext = true;
                _cachedWaypoints[i] = _waypoints[i].position;
            }
        }

        if (_recalculateNext)
        {
            Query();
            _recalculateNext = false;
        }
    }
}

```

Figure 104: Requesting Directions Script

```

void HandleDirectionsResponse(DirectionsResponse response)
{
    if (response == null || null == response.Routes || response.Routes.Count < 1)
    {
        return;
    }

    var meshData = new MeshData();
    var dat = new List<Vector3>();
    foreach (var point in response.Routes[0].Geometry)
    {
        dat.Add(Conversions.GeoToWorldPosition(point.x, point.y, _map.CenterMercator, _map.WorldRelativeScale).ToVector3d());
    }

    var feat = new VectorFeatureUnity();
    feat.Points.Add(dat);

    foreach (MeshModifier mod in MeshModifiers.Where(x => x.Active))
    {
        mod.Run(feat, meshData, _map.WorldRelativeScale);
    }

    CreateGameObject(meshData);
}

```

Figure 105: Handling Directions Script

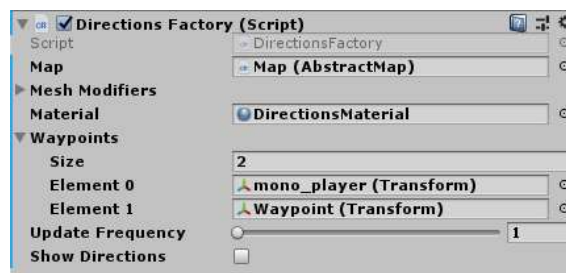


Figure 106: Direction Factory Component

The 'Directions Factory' component, takes the map in our scene and two way-

points (in our case the player and one waypoint empty object we created) and returns a route between the two of them on the map using any material the developer chooses (DirectionsMaterial in our case). The update frequency has been set to 1 second. We have inserted the option to enable or disable directions by enabling or disabling the 'Mesh Rederer' of the object, which means that when the mesh rederer is disabled the object is not rendered at all (fig. 106). This component has been assigned on the 'Directions' object of the Map Control on our scene.

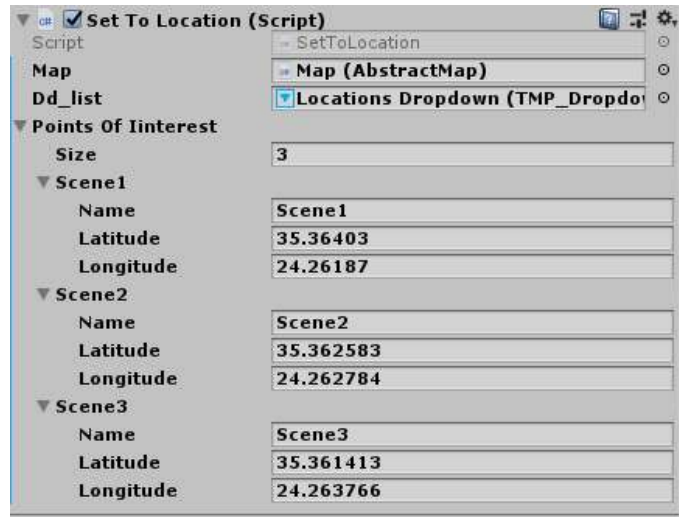


Figure 107: Set to Location Component

Each time the user selects a different location, the 'Waypoint' should change location in order to provide directions for the new selected location. To achieve that we created the 'Set To Location' component script (fig. 108). Basically, we relocate the waypoint to a new geo-location on the map based on the selection the user has selected on the dropdown list of the locations. The coordinates for the locations have been taken by Google Maps. This component is assigned to the 'Waypoint' object of the 'Map Control' on our scene.

```

using Mapbox.Unity.Utilities;
using Mapbox.Unity.Map;
using Mapbox.Utils;
using UnityEngine;

///Set a waypoint to a geo location
public class SetToLocation : MonoBehaviour
{
    [SerializeField]
    AbstractMap _map;

    [SerializeField]
    private TMPro.TMP_Dropdown dd_list;

    public GeoLocationData[] PointsOfInterest;

    //If new POI set change the direction's waypoint
    void Update()
    {
        if (GetPOIGeolocation(dd_list.value) != this.transform.GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale))
        {
            MoveWaypointToGeoLocation(this.transform);
        }
    }

    //Changing the dropdown option set new Position/Direction
    public void OptionLocation()
    {
        MoveWaypointToGeoLocation(this.transform);
    }

    //The method that sets the waypoint of the geo location
    private void MoveWaypointToGeoLocation(Transform waypoint)
    {
        waypoint.MoveToGeocoordinate(PointsOfInterest[dd_list.value].Latitude, PointsOfInterest[dd_list.value].Longitude, _map.CenterMercator, _map.WorldRelativeScale);
    }

    private Vector2d GetPOIGeolocation(int option)
    {
        return PointsOfInterest[option].GeoLocation();
    }
}

```

Figure 108: Set to Location Script

### 6.5.5 Checking Location & Distance

In order to connect all the components together and finalize the map scenes, we need to check the location of the user and calculate constantly his distance from the location he has selected. By getting this information we can provide to the user all the necessary information he/she needs for his guidance and we can enable/disable components of the scene based on user's actions.

To calculate the distance between two geo-points we use the 'GeoDistance' script (fig. 109). In this script, we calculate the distance of two points in straight line based on their coordinates. The formula we used is well-known and can be found easily on the internet. The script is a public static script and does not imported on any object of the scene.



```

public static class GeoDistance
{
    /// <summary>
    /// This class is used in order to calculate the distance between two geo locations
    /// </summary>

    public static float PoiLat;
    public static float PoiLon;

    private static float DegToRad(float deg)
    {
        return (deg * Mathf.PI) / 180.0f;
    }

    //This is the function to call to calculate the distance between two points

    public static float Calculate_Distance(float lat1, float lon1, float lat2, float lon2)
    {
        var R = 6371; // Radius of the earth in km
        var dLat = DegToRad(lat2 - lat1); // deg2rad below
        var dLon = DegToRad(lon2 - lon1);
        var a =
            Mathf.Sin(dLat / 2) * Mathf.Sin(dLat / 2) +
            Mathf.Cos(DegToRad(lat1)) * Mathf.Cos(DegToRad(lat2)) * Mathf.Sin(dLon / 2) * Mathf.Sin(dLon / 2);
        var c = 2 * Mathf.Atan2(Mathf.Sqrt(a), Mathf.Sqrt(1 - a));
        var d = R * c; // Distance in km

        //returns in meters
        return d*1000;
    }
}

```

Figure 109: Calculate Geo Distance Script

Since we have a formula to calculate distance between two points, we can calculate the distance between the user and any given point. We get the User's and point's geo-position using Mapbox's 'GetGeoPosition' function that returns the position of an object in Latitude and Longitude. After getting the coordinates it's easy to insert them in our formula to calculate their distance (fig. 110).

```

//Get Player's Geo Position
private Vector2d GetPlayerGeoPosition()
{
    return player.transform.GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale);
}

//Get POI's Geo Position
private Vector2d GetPointGeoPosition()
{
    return point.transform.GetGeoPosition(_map.CenterMercator, _map.WorldRelativeScale);
}

//Calculate distance between Player and POI position
private float GetDistance()
{
    //position[0] => Latitude
    //position[1] => Longitude
    float distance = GeoDistance.Calculate_Distance((float)playerPosition[0], (float)playerPosition[1], (float)pointPosition[0], (float)pointPosition[1]);
    return distance;
}

```

Figure 110: Get Distance Script

In the initial state of the scene, we check the user's location from the first of the three locations of interest. If the distance is greater than 2 kilometers, then the 'Alert' canvas is enabled, forcing the user to exit the application. If user's distance is less than 2 kilometers then the 'Info' canvas is enabled and the session is continued

normally (fig. 111).

```
// Start is called before the first frame update
void Start()
{
    playerPosition = GetPlayerGeoPosition();
    pointPosition = GetPointGeoPosition();

    //Away from location, pop alert else info
    if (GetDistance() > 2000)
    {
        alert_canvas.SetActive(true);
    }
    else
    {
        info_canvas.SetActive(true);
    }
}
```

Figure 111: Enable Info/Alert Script

On every frame, we get user's and point's locations. The distance is displayed always on top of screen for user's information. If the user reaches within 10 meters radius of the point the AR button is enabled on his screen. We also check if the distance is zero, because initially the 'GetDistance' function returns zero or null values until the providers get the right values (fig. 112).

```
// Update is called once per frame
void LateUpdate()
{
    //Get player's and point's position every frame
    playerPosition = GetPlayerGeoPosition();
    pointPosition = GetPointGeoPosition();

    //Show the player's distance from POI location
    UpdateDistanceText();

    //Enable AR button when player reaches 10 meters
    //close to POI location based on the selection of user
    if (GetDistance() <= 10 && GetDistance() != 0)
    {
        AR_Button.gameObject.SetActive(true);
    }
    else
    {
        AR_Button.gameObject.SetActive(false);
    }
}
```

Figure 112: Enabling AR button Script

When user presses the AR button, AR Scenes are loaded based on the selection of the dropdown list the user has selected (fig. 113).

```
//Load the Scene the User has selected
public void LoadARScene()
{
    SceneManager.LoadScene(dd_list.value + 1);
}
```

Figure 113: Load AR Scene Script

All the described functionality is part of the 'Check Location' script which is one of the components of 'SceneManager' object in our scene. Also, 'SceneManager' contains the component for enabling the right indicator and going back to 'Welcome Scene' (fig. 114).

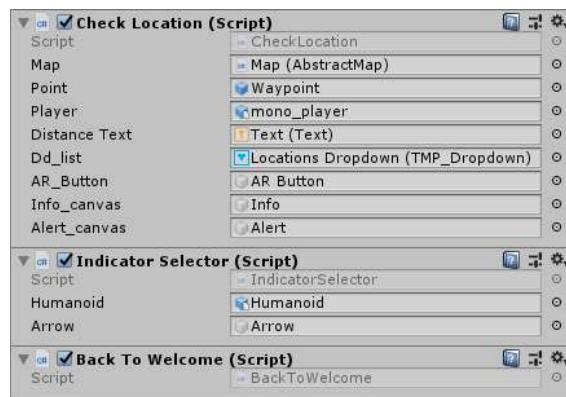


Figure 114: Scene Manager Components

## 6.6 AR Scene

The AR Scenes are where the user experiences the Augmented Reality feature. As we've already mentioned, there are three AR Scenes that represent the three locations of interest the user can visit for the AR visualization. In this section we will analyze the functionality of the AR scenes. All three AR scenes are the same with the only difference of the terrain modeling. All the functionality and components we analyze here apply on all AR scenes.

### 6.6.1 AR Scene Components

In figure 115 are all the components the scene has in Unity editor.

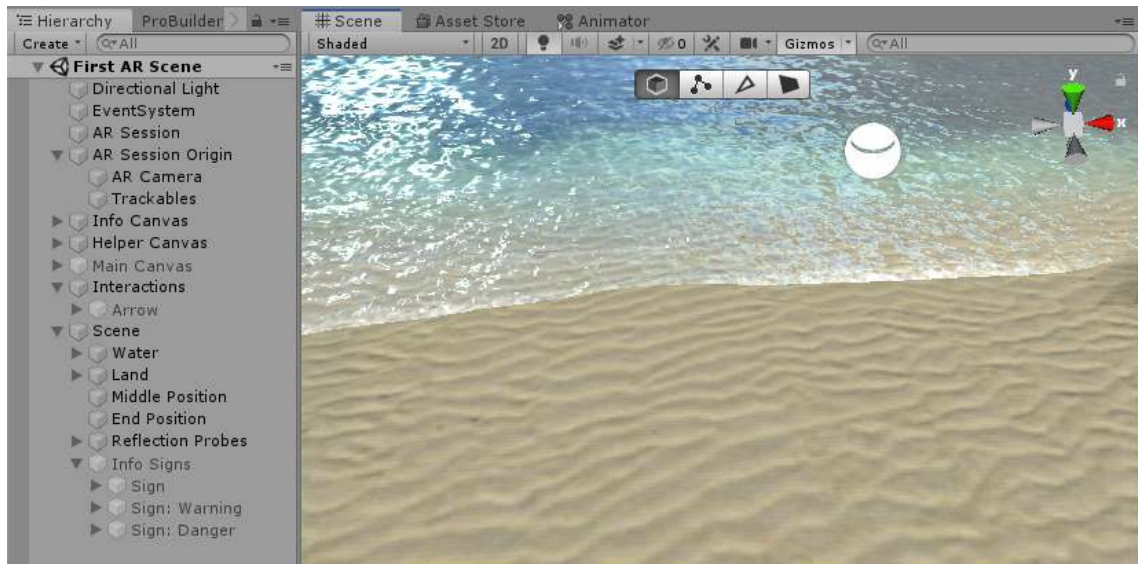


Figure 115: AR Scene Components

The main components are:

- 'Directional Light' act as the Sun in our scene.
- 'EventSystem'.
- 'AR Session' important for AR Foundation.
- 'AR Session Origin' important for AR Foundation.
- 3 UI Canvases 'Info Canvas' for the info message, 'Helper Canvas' for instantiation and calibration of the content and 'Main Canvas' which provides the main interactions of the user with the content.
- 'Interactions' for interacting with indicator mainly and functionality for calibrating the content.
- 'Scene' which contains the virtual content (terrain, water, signs) and the 'Reflection Probe' for reflection purposes on the scene and especially on water.

## 6.6.2 Setting Up AR Foundation

To activate the functionality of AR Foundation in our scene we have to import and set up two main components: 1) 'AR Session' and 2) 'AR Session Origin'. The 'AR Camera' should be our main camera in the scene and it should be child of the 'AR Session Origin'. After this the following components should be added to each object as shown in the figures below.

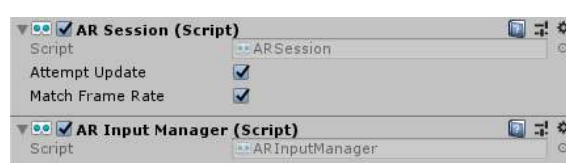


Figure 116: AR Session Components

'AR Session' controls the lifecycle and configuration options for an AR session.

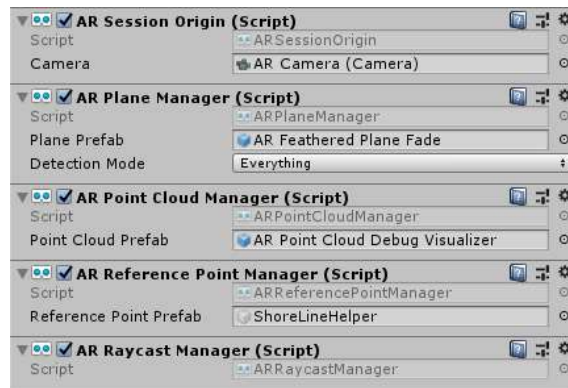


Figure 117: AR Session Origin Components

'AR Session Origin' is the object that represents the user. User sees and interacts with the virtual environment using this object. Rotation or movements of this object represents rotation and movement of the user in the scene. Any further functionality on user's side using AR Foundation's components should be made through this object.

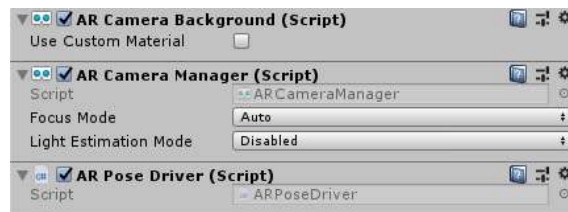


Figure 118: AR Camera Components

'AR Camera' is the "eyes" of the user, user sees anything that renders this camera.

We have enabled AR Foundation for our project, and we are able to detect plane surfaces, raycast on them and detect feature points. There is another step we need to take since we don't use the default renderer of Unity but the LWRP. First we need to create a custom forward renderer (Create - Rendering - Lightweight Render Pipeline - Forward Renderer) then click on it and add feature - ARBackgroundRendererFeature. We assign the new renderer to the LWRP settings asset. Enable 'Depth' and 'Opaque' texture (fig. 119).

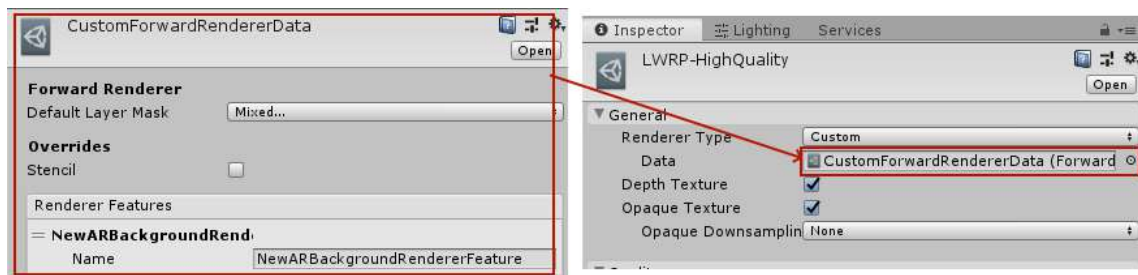


Figure 119: Custom Forward Renderer for AR

Finally, we have enabled completely AR Foundation to run on all the supported devices using LWRP.

### 6.6.3 Creating Water Shader

Before, AR Foundation announces support of LWRP, the water creation was a big issue. There was no knowledge on our part on shader programming and it was something completely different than scripting programming in Unity. Thanks to big Unity's community there is a big variety of water shaders on Unity's Store. Some of them were really good and looked pretty realistic and were for mobile devices too. Unfortunately, none of them were tested in an AR environment. Our requirements for the graphics were to be as realistic as possible. A lot of testing made with shaders on Unity's default pipeline. All the tested shaders were problematic on the AR environment and they needed a lot of resources on the hardware side.

When the LWRP finally supported by AR Foundation this problem solved. We could make our own water shader in 'Shader Graph' in schematic way. Lesser knowledge required to create a shader compared to the default way. But most important, we could have more realistic graphics with less resources, optimized to run on mobile devices. Below, is the graph on 'Shader Graph' for the water shader we created.



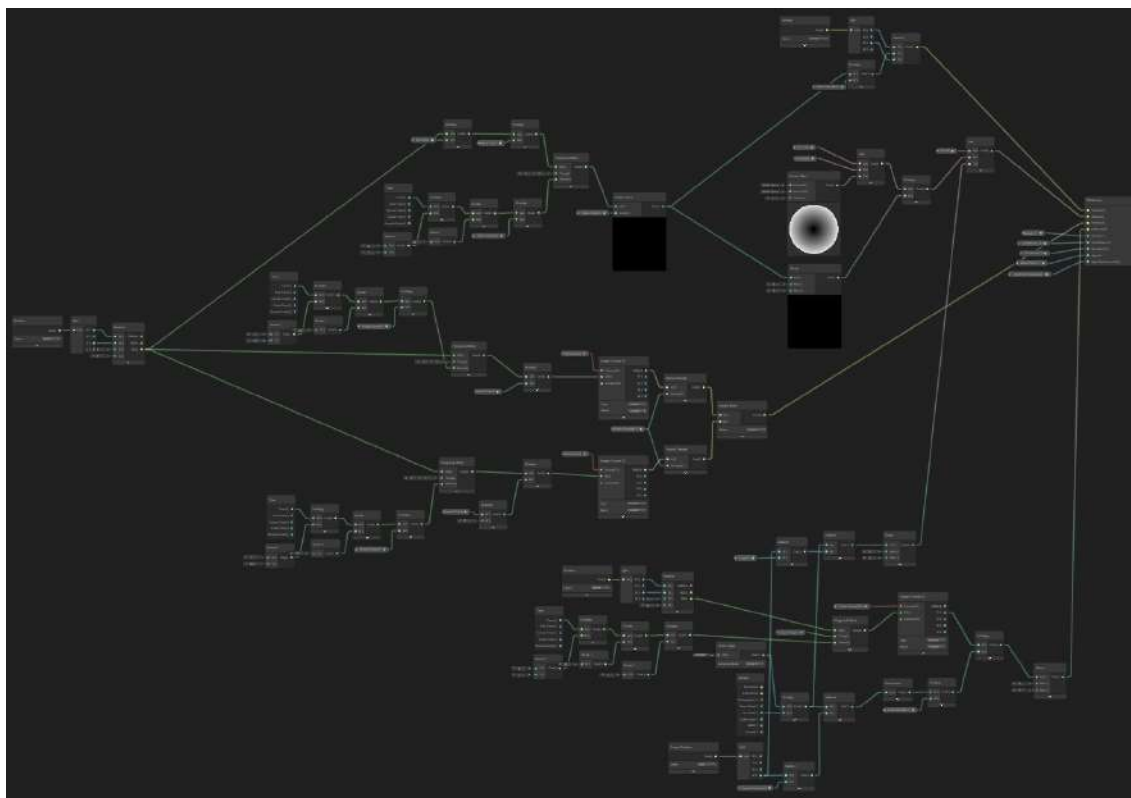


Figure 120: Our Water Shader on Shader Graph

We can change the variables of this shader by making them public. The shader should be attached on a material and this material to be assigned at any object. Then it's easy to change and test the shader using Unity's Editor.

The basic idea is:

Two colors blend together to create the color of realistic-looking water. Two normal maps blend to create the wavy surface of a natural sea. Normal maps are 2D textures that give their geometry on the applied surface. After this, we have waves that we can adjust in our needs. We can increase or decrease their length, strength, scale, direction and frequency. Finally, the water emits a foam-like texture when collides with other objects and changes the color of the depth a bit. Making this, we achieve a shore foam effect like the real water when hits the land. We adjust the metallic and smoothness based on the feeling we want our water to give and how reflective we want it to be (fig. 121).

It's really important to give to the shader the properties that shown on figure 122. Metallic, by testing, is the type of material that represents better our purpose and makes a material more reflective (water is reflective). The surface should be transparent like the real water is and finally and most important the blend should be additive. That makes the material to adjust its color based on the surface it is on, blending with the environment. That was the only way we achieved to make our water more realistic since the water is a colorless material that takes the color of the environment.

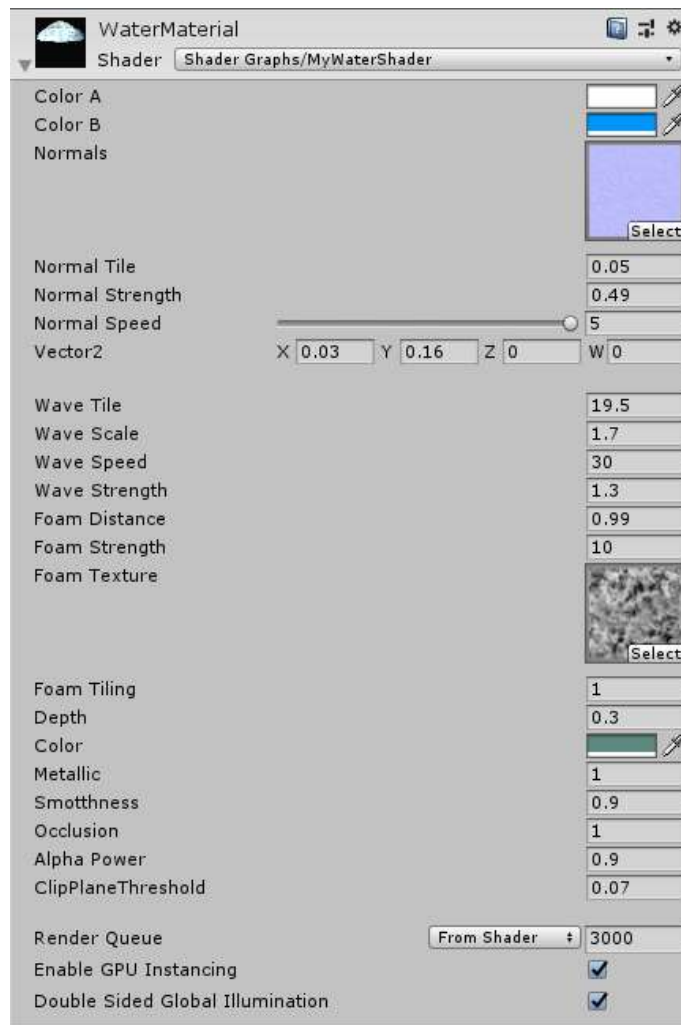


Figure 121: Our Water Shader Public Variables

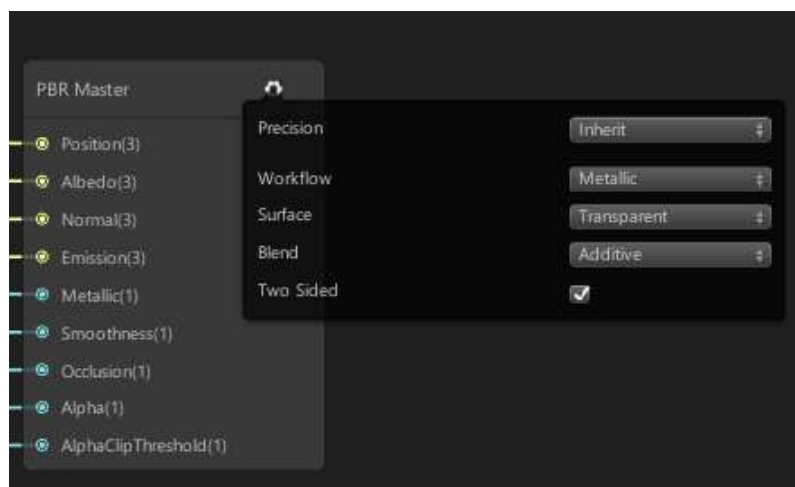


Figure 122: Essential Water Shader Properties on Shader Graph

Below an example of the water shader with alpha and additive blend. Standing in the shoreline someone is able to see the sand through the water. The water gets darker only on high depths which is not our case.

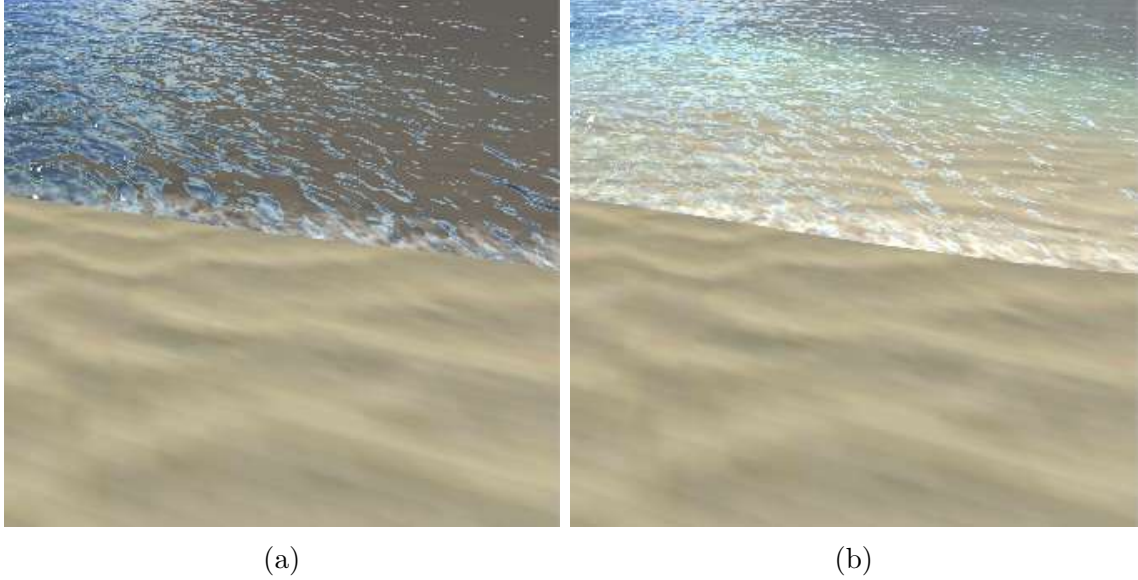


Figure 123: Water Shader: (a) Alpha Blend, (b) Additive Blend

And finally, a showcase of the water tested with the default Unity's renderer and our shader in LWRP.

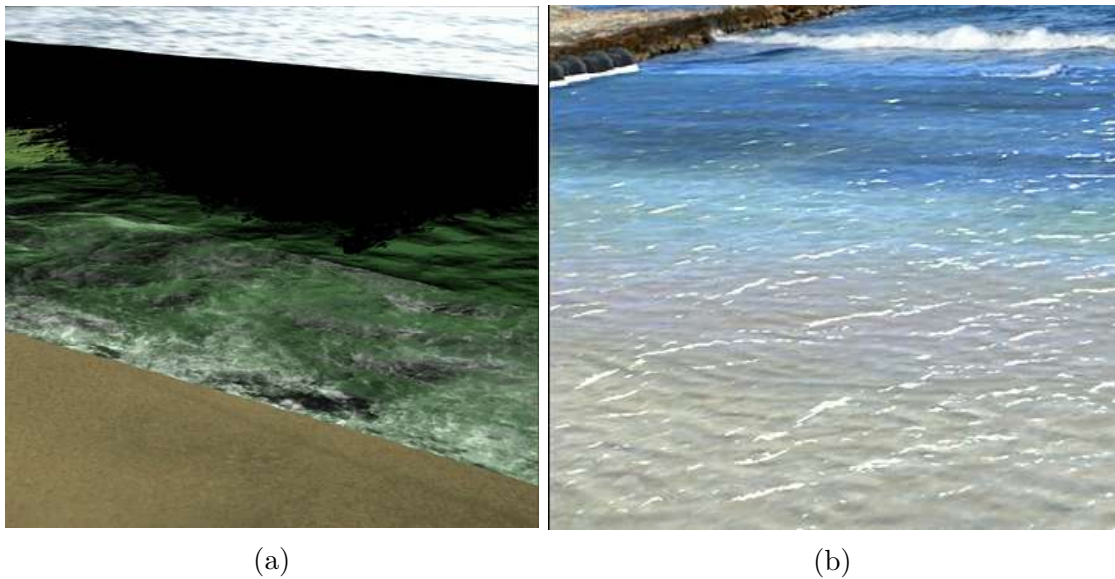


Figure 124: Water: (a) Default Unity Render, (b) Lightweight Render Pipeline

#### 6.6.4 Designing Virtual Content

After we have implemented all the tools we needed, it was time to create the virtual content that would be put on top of the real world. The initial idea was to use only the water object. Soon, it was understandable that there was no way to make the water interact on a physical way with the real world. The AR content is being put on top of the real world and there was no way the two worlds to interact with each other. So, we decided to create a terrain based on the shape of the beach at the specific location. Using the terrain, we could achieve interaction with the water. We could give to the water whatever shape we wanted to, based on the terrain we designed.

To design the terrain based on the real world and give it a shape to correspond to the shoreline as it is, we used the figure 48. This was the guide to our terrain modeling during the whole process (fig. 125). The scale should be on point, so we use real scale inside Unity, this means that 1 unit in Unity corresponds in 1 meter in the real world. The specific part in figure 125 has 27 meters length and 58 meters width. Similar dimensions have been used to the other two segments.

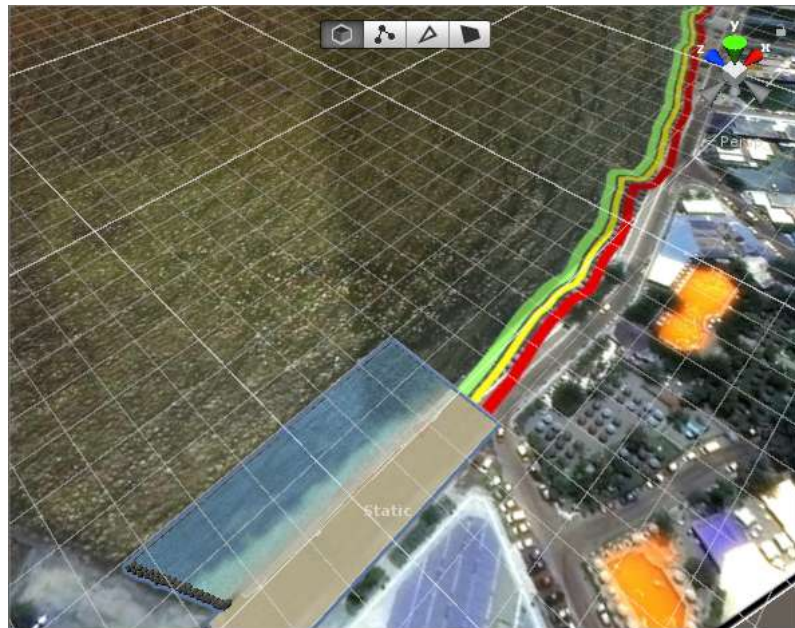


Figure 125: Designing 3D Content based on the Data

To create the three differences in the shoreline (current-green, scenario 1-yellow and scenario 2-red), we elevate our terrain to 1 meter height. Then we created three layers, one with 0.33m height to represent the current shoreline, one with 0.66m to represent the first scenario and one with 0.99m to represent the second scenario. Making this we could elevate the water accordingly to visualize the scenario we wanted (fig. 126).





Figure 126: Layer Designing on 3D Content

To create a mini animation while the water was rising, we used the lerp method. Lerp makes the object to move from one position to another gradually during a time window at a given speed. They have been set two positions that the water moves towards them when the user presses the button for one of the scenarios. The script is shown in figure 127.

```
//Parameters for Lerp
private float startTime, midDistance, totalDistance;
private Vector3 startPos;
private bool firstElevation = false;
private bool secondElevation = false;

// Update is called once per frame
void Update()
{
    //The first Lerp
    if (firstElevation == true)
    {
        float currentDuration = (Time.time - startTime) * speed;
        float journeyFraction = currentDuration / midDistance;
        this.transform.position = Vector3.Lerp(startPos, midPos.position, journeyFraction);
    }
    //The second Lerp
    else if (secondElevation == true)
    {
        float currentDuration = (Time.time - startTime) * speed;
        float journeyFraction = currentDuration / totalDistance;
        this.transform.position = Vector3.Lerp(midPos.position, endPos.position, journeyFraction);
    }
}

//Get the position, time and distance for the first Water Lerp
public void FirstElevation()
{
    startPos = this.transform.position;
    startTime = Time.time;
    midDistance = Vector3.Distance(startPos, midPos.position);
    firstElevation = true;
    sign_warning.SetActive(true);
}

//Get the position, time and distance for the second Water Lerp
public void SecondElevation()
{
    startPos = this.transform.position;
    startTime = Time.time;
    totalDistance = Vector3.Distance(midPos.position, endPos.position);
    firstElevation = false;
    secondElevation = true;
    sign_danger.SetActive(true);
}
```

Figure 127: Water Lerp Script

On the edge of each layer, we added a UI component, the signs. Each sign, is a UI element that always looks at the user. They are enabled after the user has selected to view each scenario. We added box colliders on all the three signs making possible the raycasting on them for enabling the info panels.

Finally, a 'Reflection Probe' object added to the scene. A 'Reflection Probe' is rather like a camera that captures a spherical view of its surroundings in all directions. The captured image is then stored as a 'Cubemap' that can be used by objects with reflective materials like our water. The result is that the reflections on the water can change convincingly according to its environment.

### **6.6.5 Interactivity in AR**

There are basic interactions for the user in the AR mode. From placing the virtual object to interact with the signs. Below we will demonstrate the core functionality of user's interactions in the AR Scenes.

The most important interaction of the user, is to scan the environment around him, detect the ground and place the virtual content on top of the real world. Once the user enters the AR scene, the phone starts to scan the environment to detect plane surfaces and feature points. This was enabled when we attached on 'AR Session Origin' object the 'AR Plane Manager' component and the 'AR Point Cloud Manager' component. The trackables that have been detected are saved locally by AR Foundation within the 'AR Session Origin'. To check if we have detect a plane, we send a raycast ('AR Raycast Manager' component used for this). If the raycast hits a trackable plane then we show the arrow indicator to the user, otherwise we show a message to the user to scan more. This functionality is shown below in [figure 128](#).



```

Pose hitPose;
void Update()
{
    if (placedPrefab == null)
        return;

    if (m_RaycastManager.Raycast(new Vector2(Screen.width / 2, Screen.height / 2), s_Hits, TrackableType.Planes))
    {
        // Raycast hits are sorted by distance, so the first one
        // will be the closest hit.
        hitPose = s_Hits[0].pose;

        if (placementIndicator != null)
        {
            placementIndicator.SetActive(true);
            placementIndicator.transform.position = hitPose.position;
            placementIndicator.transform.rotation = hitPose.rotation;
            spawn_button.SetActive(true);
        }

        if (scan_txt != null)
            scan_txt.SetActive(false);
    }
    else
    {
        if (placementIndicator != null)
        {
            placementIndicator.SetActive(false);
            spawn_button.SetActive(false);
        }

        if (scan_txt != null)
            scan_txt.SetActive(true);
    }
}

```

Figure 128: Check for Tackable Panes Script

When a plane has detected, the user is able to instantiate the AR content in the real world. When user presses the 'Place Here' button is called a method named 'SpawnScene'. Its functionality is shown below in figure 129.

These two parts of code, is part of the 'PlaceOnPlane' script which is attached as component to 'AR Session Origin' object and makes use of all the AR components we added earlier.

```

//This is called by a button
public void SpawnScene()
{
    //Add a reference point where user hits spawn
    //position and rotation according to pose of the phone
    referencePoint = m_ARReferencePointManager.AddReferencePoint(hitPose);
    s_ReferencePoints.Add(referencePoint);

    //Destroy indicator, scan text, button
    Destroy(placementIndicator);
    Destroy(scan_txt);
    Destroy(spawn_button);

    //Instantiate object based on the position and rotation of the point we created before
    m_ARSessionOrigin.MakeContentAppearAt(placedPrefab.transform, referencePoint.transform.position, referencePoint.transform.rotation);

    //Activate the buttons
    done_button.SetActive(true);
    settings_button.SetActive(true);
}

```

Figure 129: Spawn Scene Script

Other interactions the user has in AR are: the calibration of the content (script fig. 130), 'touch' the signs of the scene to read the info (script fig. 131) and disable the rendering of the trackables' visualizer (script fig. 132), which disables the grey dots on the detected planes.

```

public void RotateLeft()
{
    rotation = Quaternion.AngleAxis(rotation.eulerAngles.y + 5.0f, Vector3.up);
    m_ARSessionOrigin.MakeContentAppearAt(content.transform, content.transform.position, rotation);
}

public void RotateRight()
{
    rotation = Quaternion.AngleAxis(rotation.eulerAngles.y - 5.0f, Vector3.up);
    m_ARSessionOrigin.MakeContentAppearAt(content.transform, content.transform.position, rotation);
}

public void MoveLeft()
{
    content.transform.position = new Vector3(content.transform.position.x - 0.3f, content.transform.position.y, content.transform.position.z);
}

public void MoveRight()
{
    content.transform.position = new Vector3(content.transform.position.x + 0.3f, content.transform.position.y, content.transform.position.z);
}

```

Figure 130: Calibration Content Script

```

void Update()
{
    for (var i = 0; i < Input.touchCount; ++i)
    {
        if (Input.GetTouch(i).phase == TouchPhase.Began)
        {
            // Construct a ray from the current touch coordinates
            Ray ray = Camera.main.ScreenPointToRay(Input.GetTouch(i).position);
            RaycastHit hit;
            // Create a particle if hit
            if (Physics.Raycast(ray, out hit, Mathf.Infinity) && hit.transform.name == "Sign0")
            {
                info0.SetActive(true);
            }
            else if (Physics.Raycast(ray, out hit, Mathf.Infinity) && hit.transform.name == "Sign1")
            {
                info1.SetActive(true);
            }
            else if (Physics.Raycast(ray, out hit, Mathf.Infinity) && hit.transform.name == "Sign2")
            {
                info2.SetActive(true);
            }
        }
    }
}

```

Figure 131: Raycast Signs Script

```

private bool show;
// Start is called before the first frame update
void Awake()
{
    m_ARPlaneManager = GetComponent<ARPlaneManager>();
    m_ARPointCloudManager = GetComponent<ARPointCloudManager>();
    show = true;
}

public void HideTrackables()
{
    show = !show;

    foreach (ARPlane plane in m_ARPlaneManager.trackables)
    {
        plane.gameObject.SetActive(show);
    }

    foreach (ARPointCloud pointCloud in m_ARPointCloudManager.trackables)
    {
        pointCloud.gameObject.SetActive(show);
    }
}

```

Figure 132: Toggle Trackables Script

## 7 Conclusion

### 7.1 Summary

In this thesis we presented the design of a mobile Augmented Reality application aimed for consumer-grade mobile phones with the ultimate goal of increasing the environmental awareness of the public audience. In addition, to make aware the coastal erosion problem on the beach of Georgioupoli and general on Crete's beaches we offered an approach for visualizing coastal changes on-site using Augmented Reality. By employing 3D representation through AR we aimed to enhance user-experience on coastal erosion and bridge the gap between digital content and the real environment. Our design was focused on providing an expandable application and an idea, that can easily envelop more beaches and locations requiring some preparation and will enable future experts to display their digitized collections using different forms of data presentation. Data visualization and Augmented Reality seems promising and we wanted to present an idea combining those two, showing their capabilities and their future potential.

During this process one of the most valuable lessons we benefit from is the unpredictable problems and obstacles someone can face and how he/she can manage to overcome them. Augmented Reality is a field that has just reached the wider public. Mobile AR especially, where the processing power and sensor availability is limited, makes the originally envisioned result even harder to reach. The experiences someone wants to create are tightly correlated to the available technologies. Each use case varies greatly, leading the developer to review and reestablish the initial requirements. Nonetheless, there is much research being done and Augmented Reality has evolved greatly with the addition of more sophisticated algorithms and specialized hardware. It's a rapidly evolving field. We were witnessing constant changes during the process and they continued even after the end of it.

Although outdoors Mobile Augmented Reality presents several challenges on a technological aspect (concerning localization and registration), it already seems more than capable of providing novel experiences to a wide audience. The availability and technological advances of modern smartphones allows for an ideal integration of the technology that can enhance the understanding of geo-spatial datasets and the generation of more meaningful experiences.

### 7.2 Evaluation

#### 7.2.1 Technical Characteristics

To evaluate the resulting application we conducted field tests with two different devices. The devices were: a high-end Xiaomi Mi 8 (Snapdragon 845 CPU @ 2,8GHz, Adreno 630 GPU, 6GB RAM, 4G LTE, Dual frequency-GPS, Geo-magnetic

sensor, accelerometer, gyroscope) and a mid-range Xiaomi Mi 8 SE (Snapdragon 710 CPU @ 2,2 GHz, Adreno 616 GPU, 4GB RAM, GPS, Geo-magnetic sensor, accelerometer, gyroscope). While one of the devices (MI 8) has better hardware, the performance is not noticeably better. Both devices run the application smoothly and all the functionalities work fine. The better GPS accuracy of the Mi 8 does not present a better experience in the map view. In AR mode the visualization was working as intended on both devices but the higher-end Mi 8 seemed to drain the battery a little bit faster. This might have nothing to do with the application and it may be a result of poorer battery health of the device or extensive battery usage of the more power consuming hardware of the given device (CPU, GPU, GPS). Unfortunately, we hadn't an iOS device available to conduct tests on both operating systems.

After a full test experience, visiting two of the three locations and running the application for 1.5 hour, the application had consumed approximately 15% of the phone's battery. The data that were received and sent were around 5 MB and mostly for mapping purposes. The average RAM consumption was 400-500 MB. All the measurements depend entirely on the user's usage and time of execution of the application. The application requires 85.24 MB of the phone's storage in order to be installed.

## **7.2.2 Goal and Functionality**

After testing and showing our application to a variety of users, we received feedback about the functionality of the application and most importantly about its usefulness.

Most of the users, find it challenging to place the virtual content right. This varied from location to location as it was easier to align the virtual content in some locations than others. More specifically location 3 was the most challenging because of the permanent existence of beach umbrellas. After a small period of training though, the users were able to put the content on the right place and experience the visualization as it was meant to be. Other than this, the users find it easy to use the app and navigate around.

We received positive feedback about the graphics of the application, and especially on the AR component as they looked really realistic. Graphics really help in AR and give a clear and understanding way to look into the future changes. Of course, there is plenty of room for improvement, especially in the lighting of AR Scenes in order to present better the feeling of depth.

The signs and the UI in general were simple and intuitive, and they added a nice touch to the overall experience and to the knowledge we would like to transfer to the user about coastal erosion. Some users preferred more clearly visible colors on the UI.

The application, was hard to use on bad weather conditions (clouds, wind etc.). Especially when there was wavy sea, it was almost impossible to anchor right the virtual content and there was drifting of the graphics in the scene making the AR experience really poor.

Overall, there were users non-relevant with AR technology and users with deep knowledge in AR. The non-relevant users seemed very enthusiastic about the whole AR experience and made comments about the optical results and the graphics while the relevant users focus more on the functionality and the possible improvements. Some mentioned that they would prefer a cardboard experience while others had no problem with the use of smartphone as it is easier to use. Most of them agreed that it was an original and unique application and an innovative use of Augmented Reality.

Augmented Reality technology has a lot of problems to solve in order to be fully functional and available to everyone. We believe that our goal, given the available tools, was achieved and we showcased an interesting way for landscape visualization. Showcasing the capabilities of AR, we provide useful knowledge and environmental awareness using a way more interactive and fun than the usual ones.

## **7.3 Future Work**

The field of Augmented Reality is a quickly evolving one where new technologies are rising with high frequency so our application should follow this progress. The techniques used in this application should not be taken for granted as tracking and registration in AR are far from solved. Below, we will mention some changes that can be made on the future that haven't been done so far due to lack of software/hardware equipment and time. We worked with everything we had available and with the software that has been released and tested by then.

### **7.3.1 Increase Environmental Understanding**

In our application, the environmental understanding of the application is based only on the GPS sensor on the map view and the SLAM (Simultaneous Localization and Mapping) capabilities of ARCore/ARKit in AR view.

We wanted to implement an AR navigation for the user but the GPS inaccuracies make it extremely hard to work as intended. A new approach named VPS (Visual Positioning System) allows the device to track its position on the world based on visual feedback it gets through the camera. This allows the augmentation of the real world with a more immersive way, anchoring objects in centimeters accuracy on the real world. Such systems are still in progress, Google announced its VPS on Google Maps but it's still on testing and not widely available. Another interesting VPS that is available for AR use is Sturfee VPS. Unfortunately, it supports only 15 major cities around the world. A VPS would largely enhance the capabilities of the

application and the overall experience.

A major upgrade, would be the newly announced Depth API by Google. Since we already use ARCore as the AR functionality, it should be easy to implement Depth API giving depth capabilities to our application and better understanding of the world and the mesh of it. Implementing some kind of depth understanding the capabilities are endless. We will be able, probably, to interact virtual objects with the real ones. In our case, our virtual water might be able to collide with the real ground and there will be no need of a virtual terrain. Less graphics means better performance and experience so the remaining resources will be available for further content and visualization.

### **7.3.2 Improve Shoreline Detection Method**

In order to understand the application where the shoreline is, it depends on user's actions and perspective. We would like to relief the user from such a task, and let the application to determine where the shoreline is and proceeds on its own from there. There are probably many methods to achieve this, but an elegant way and a more future proof would be to implement an edge detection algorithm alongside with machine learning. There are plenty machine learning software (e.g. TensorFlow) which after a training period, can detect objects on the real world. Using this technology in the future in combination with some edge detection algorithm to detect effectively the shoreline on real time might decrease the need of the user's interaction in the detection phase. This is only an initiative idea, that may not be achievable soon enough.

### **7.3.3 Visual Upgrade & Optimization**

We might achieve a nice visual result in terms of graphics, but surely sky is the limit. LWRP was released during the late phase of our application development and thus the visual result is not the best we could achieve with the given tools. The lack of knowledge on such new way of rendering and shader creation was a key factor for this result. The community is still on the training curve of the newly added features. It is strongly believed that the visual aspect of the application can become even better in the future with the right investment of time and the optimization of the given tools.

Optimization in terms of sources consumption is a major upgrade. A mobile application should be as optimized as possible for mobile devices where the resources are limited and the energy consumption is a major factor for the best performance. Due to limited time, optimization in our app took a secondary role and it's something that would be good to be taken care of in the future.



### **7.3.4 HMD implementation**

The application would be nice to be implemented as a cardboard application too, offering a more immersive experience trying to simulate the experience of a standalone HMD. The biggest step would be to bring the application on a Head Mounted Display, like the ones we mentioned in this thesis earlier. This requires a whole new different perspective on the developing phase and the usage of different SDKs. The HMDs so far, are not the best for outdoors AR applications but they surely bring a wide variety of tools and their world understanding are far better than a mobile smartphone until now.

### **7.3.5 Adding More Scenarios**

Last but not least, more scenarios about other beaches of Crete would be nice addition. The beach of Georgioupoli is not the only beach in Crete that has been undergone extensive erosion. There are plenty of them on the north and east side of Crete that are in extreme danger. So, the addition of more beaches is really important to achieve our initial goal to increase the environmental awareness of the public about coastal erosion.

## 8 Bibliography

### References

- [1] Paul Milgram, Haruo Takemura, Akira Utsumi, and Fumio Kishino. Augmented reality: a class of displays on the reality-virtuality continuum. In Hari Das, editor, *Telemanipulator and Telepresence Technologies*, volume 2351, pages 282 – 292. International Society for Optics and Photonics, SPIE, 1995.
- [2] Ivan E. Sutherland. A head-mounted three dimensional display. In *AFIPS '68 (Fall, part I)*, 1968.
- [3] T. Caudell and D. Mizell. Augmented reality: An application of heads-up display technology to manual manufacturing processes. In *System Sciences, 1992. Proceedings of the Twenty-Fifth Hawaii International Conference on*, volume ii, page 659–669, 1992.
- [4] T. Hollerer S. Feiner, B. MacIntyre and A. Webster. A touring machine: prototyping 3d mobile augmented reality systems for exploring the urban environment. *IEEE International Symposium on Wearable Computers*, 1:208–217, 1997.
- [5] Clemens Arth, Raphael Grasset, Lukas Gruber, Tobias Langlotz, Alessandro Mulloni, and Daniel A. Wagner. The history of mobile augmented reality developments in mobile ar over the last almost 50 years. 2015.
- [6] B. Thomas, B. Close, J. Donoghue, J. Squires, P. De Bondi, M. Morris, and W. Piekarski. Arquake: an outdoor/indoor augmented reality first person application. In *Digest of Papers. Fourth International Symposium on Wearable Computers*, pages 139–146, Oct 2000.
- [7] Daniel Wagner, Thomas Pintaric, Florian Ledermann, and Dieter Schmalstieg. Towards massively multi-user augmented reality on handheld devices. In Hans W. Gellersen, Roy Want, and Albrecht Schmidt, editors, *Pervasive Computing*, pages 208–219, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.
- [8] A. Henrysson, M. Billinghurst, and M. Ollila. Face to face collaborative ar on mobile phones. In *Fourth IEEE and ACM International Symposium on Mixed and Augmented Reality (ISMAR'05)*, pages 80–89, Oct 2005.
- [9] Paraskevi Theodorou, P Kydonakis, Maria Botzori, and Constantina Skanavis. Augmented reality proves to be a breakthrough in environmental education, 07 2018.
- [10] Tien-Chi Huang, Chia-Chen Chen, and Yu-Wen Chou. Animating eco-education: To see, feel, and discover in an augmented reality-based experiential learning environment. *Computers Education*, 96, 02 2016.
- [11] Paul Haynes, Sigrid Hehl-Lange, and Eckart Lange. Mobile augmented reality for flood visualisation. *Environmental Modelling Software*, 109:380 – 389, 2018.

- [12] Lemonia Ragia and Pavlos Krassakis. Monitoring the changes of the coastal areas using remote sensing data and geographic information systems. In *Seventh International Conference on Remote Sensing and Geoinformation of the Environment (RSCy2019)*, volume 11174, pages 289 – 297. International Society for Optics and Photonics, SPIE, 2019.
- [13] I.N. Monioudi, A. Karditsa, A. Chatzipavlis, G. Alexandrakis, O.P. Andreadis, A.F. Velegrakis, S.E. Poulos, G. Ghionis, S. Petrakis, D. Sifnioti, T. Hasiotis, M. Lipakis, N. Kampanis, T. Karambas, and E. Marinos. Assessment of vulnerability of the eastern cretan beaches (greece) to sea level rise. *Regional Environmental Change*, 2014. cited By 0; Article in Press.
- [14] Anne-Cecilie Haugstvedt and John Krogstie. Mobile augmented reality for cultural heritage: A technology acceptance study. pages 247–255, 11 2012.
- [15] Enabling smart retail settings via mobile augmented reality shopping apps. *Technological Forecasting and Social Change*.
- [16] Majed Abdullah Alrowaily and Manolya Kavakli. Mobile augmented reality for environmental awareness: A technology acceptance study. In *Proceedings of the 2018 10th International Conference on Computer and Automation Engineering*, ICCAE 2018, page 36–43, New York, NY, USA, 2018. Association for Computing Machinery.
- [17] P. Milgram and F. Kishino. Taxonomy of mixed reality visual displays. In *IEICE Transactions on Information and Systems*, volume E77-D, page 1321 – 1329, 1994.
- [18] Ronald T. Azuma. A survey of augmented reality. *Presence: Teleoperators and Virtual Environments*, 6(4):355–385, 1997.
- [19] H. Kato and M. Billinghurst. Marker tracking and hmd calibration for a video-based augmented reality conferencing system. In *Proceedings 2nd IEEE and ACM International Workshop on Augmented Reality (IWAR’99)*, pages 85–94, Oct 1999.
- [20] Silvia de los Ríos, María Fernanda Cabrera-Umpiérrez, María Teresa Arredondo, Miguel Páramo, Bastian Baranski, Jochen Meis, Michael Gerhard, Belén Prados, Lucía Pérez, and María del Mar Villafranca. Using augmented reality and social media in mobile applications to engage people on cultural sites. In Constantine Stephanidis and Margherita Antona, editors, *Universal Access in Human-Computer Interaction. Universal Access to Information and Knowledge*, pages 662–672, Cham, 2014. Springer International Publishing.
- [21] Te-Lien Chou and Lih-Juan Chanlin. Augmented reality smartphone environment orientation application: A case study of the fu-jen university mobile campus touring system. *Procedia - Social and Behavioral Sciences*, 46:410–416, 12 2012.
- [22] Dhiraj Amin and Sharvari Govilkar. Comparative study of augmented reality sdk’s. *International Journal on Computational Science Applications*, 5:11–26, 02 2015.

- [23] Dhiraj Amin and Sharvari Govilkar. Comparative study of augmented reality sdk's. *International Journal on Computational Science Applications*, 5:11–26, 02 2015.
- [24] R. T. Azuma. The most important challenge facing augmented reality. *Presence*, 25(3):234–238, Dec 2016.
- [25] Areti Kotsoni, Despina Dimelli, and Lemonia Ragia. Land use planning for sustainable development of coastal regions. In *Proceedings of the 3rd International Conference on Geographical Information Systems Theory, Applications and Management - Volume 1: GISTAM*, pages 290–294. INSTICC, SciTePress, 2017.

### Useful Links:

- [26] Google ARCore Depth API:  
<https://developers.googleblog.com/2019/12/blending-realities-with-arcore-depth-api.html>
- [27] Google ARCore Atom Visualizer:  
<https://play.google.com/store/apps/details?id=com.signalgarden.atomvisualizer&hl>
- [28] Google ARCore:  
<https://developers.google.com/ar>
- [29] Apple ARKit:  
<https://developer.apple.com/arkit>
- [30] Wikitude:  
<https://www.wikitude.com>
- [31] Vuforia:  
<https://www.ptc.com/en/products/augmented-reality>
- [32] Unity3D:  
<https://unity.com>
- [33] Unity AR Foundation:  
<https://unity.com/unity/features/arfoundation>
- [34] AR Foundation Documentation:  
<https://docs.unity3d.com/Packages/com.unity.xr.arfoundation@3.1/>
- [35] VR Vision Website:  
<https://vrvisiongroup.com/the-future-of-job-training-augmented-reality>
- [36] BlippAR AR City:  
<https://www.blippar.com/blog/2017/11/06/welcome-ar-city-future-maps-and-navigation>

- [37] Microsoft Hololens:  
<https://www.microsoft.com/en-us/hololens>
- [38] Time Website/Hololens:  
<https://time.com/4190843/microsoft-hololens-demo-2016>
- [39] Magic Leap One:  
<https://www.magicleap.com/magic-leap-1>
- [40] Magic Leap AR Interface:  
<https://gagadget.com/en/36808-leap-motion-showed-ar-interface-in-the-style-of-iron-man>
- [41] Nreal Light:  
<https://www.nreal.ai>
- [42] AR on Focus Magazine:  
<https://www.linkedin.com/pulse/italian-publisher-mondadori-embraces-inglobes-reality-mir>
- [43] Tracking of industrial objects by using CAD models:  
<https://www.jvrb.org/past-issues/4.2007/1159>
- [44] Plane Detection using Wikitude SDK:  
<https://next.reality.news/news/mobile-ar-apps-can-now-track-any-surface-using-plane-detect>
- [45] Tim Bjarin in his 2017 Time article:  
<https://time.com/4761298/augmented-reality/>
- [46] The Multifaceted Future Of Augmented Reality:  
<https://online.maryville.edu/blog/the-multifaceted-future-of-augmented-reality/>
- [47] ArcGIS website:  
<https://www.esri.com/en-us/arcgis/about-arcgis/overview>
- [48] ArcGIS Wiki:  
<https://en.wikipedia.org/wiki/ArcGIS>
- [49] ARCore/ARKit Supported Devices:
  - 1) <https://developers.google.com/ar/discover/supported-devices>
  - 2) [https://github.com/rolandsmeenk/ARCore-devices/blob/master/arcore\\_devicelist.csv](https://github.com/rolandsmeenk/ARCore-devices/blob/master/arcore_devicelist.csv)
- [50] Game Engines Wiki:  
<https://en.wikipedia.org/wiki/Gameengine>
- [51] Unreal Engine 4:  
<https://www.unrealengine.com/unreal-engine-4>
- [52] Mapbox SDK:  
<https://docs.mapbox.com/unity/maps/>
- [53] Lightweight Render Pipeline:  
<https://docs.unity3d.com/Packages/com.unity.render-pipelines.lightweight@6.9/>

- [54] Shader Graph:  
<https://docs.unity3d.com/Packages/com.unity.shadergraph@6.9/>
- [55] Terrain Tools:  
<https://docs.unity3d.com/Packages/com.unity.terrain-tools@2.0/>
- [56] ProBuilder:  
<https://docs.unity3d.com/Packages/com.unity.probuilder@4.3/>
- [57] Unity UI:  
<https://docs.unity3d.com/Packages/com.unity.ugui@1.0/>
- [58] Turbosquid:  
<https://www.turbosquid.com/>