UNIVERSITY OF THESSALY

SCHOOL OF ENGINEERING

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

ENHANCING ACCESS TO CULTURAL HERITAGE WITH AUGMENTED REALITY

Diploma Thesis

Pantelis Dimitroulis

SUPERVISORS:

Tsompanopoulou Panagiota

Vasilakopoulos Michail

Volos 2018

COPYRIGHT

ACKNOWLEDGEMENTS

First and foremost, I would like to extend my appreciation to my advisor, Tsompanopoulou Panagiota and also to Feygas Athanasios for their invaluable support and guidance, which has been instrumental in the development of this thesis. Of course I would like to thank all my friends and fellow students in Volos. I would like to thank my family, for their support and patience throughout the last 5 years.

ΤΕΧΝΙΚΕΣ ΕΠΑΥΞΗΜΕΝΗΣ ΠΡΑΓΜΑΤΙΚΟΤΗΤΑΣ ΓΙΑ ΤΗ ΒΕΛΤΙΩΣΗ ΠΡΟΣΒΑΣΗΣ ΣΕ ΧΩΡΟΥΣ ΠΟΛΙΤΙΣΤΙΚΗΣ ΚΛΗΡΟΝΟΜΙΑΣ

ΠΕΡΙΛΗΨΗ

Τα τελευταία χρόνια η χρήση της τεχνολογίας της Επαυξημένης Πραγματικότητας έχει αυξηθεί σε πολλούς τομείς, όπως ο τουρισμός, η διαφήμιση, η ιατρική κ.α. με έναν από τους πιο διαδεδομένους να είναι η πολιτιστική κληρονομιά. Υπάρχουν διάφορα πρότζεκτ που αξιοποιούν την τεχνολογία αυτή για να εμπλουτίσουν με νέες λειτουργίες τα μουσεία, βελτιώνοντας παράλληλα την εμπειρία του επισκέπτη. Με παρόμοιο στόχο, αυτή η διπλωματική εργασία παρουσιάζει μία πρότυπη εφαρμογή Εικονικής Πραγματικότητας για κινητές συσκευές. Σκοπός της εφαρμογής είναι να βελτιώσει την προσβασιμότητα σε χώρους πολιτιστικής κληρονομιάς, προσφέροντας εκπαιδευτικά πολυμέσα και διαδραστικά παιχνίδια στους επισκέπτες. Η εφαρμογή έχει αναπτυχθεί με τα εργαλεία: Vuforia Augmented Reality SDK, Unity Engine και C#, αξιοποιώντας τεχνολογίες Αναγνώρισης Εικόνας και τρισδιάστατων εικονικών μοντέλων. Η συγκεκριμένη υλοποίηση δεν απαιτεί σύνδεση δικτύου, ούτε και πρόσθετο εξοπλισμό ή ετικέτες όπως QR codes μέσα στο μουσείο. Συνεπώς, μία τέτοια εφαρμογή καθιστά την εμπειρία του χρήστη βολική, ενώ παράλληλα διευκολύνει την ενσωμάτωσή της στο μουσείο. Επιπλέον, εξετάζουμε την περίπτωση μελέτης του αρχαιολογικού μουσείου Βόλου εξηγώντας τις προκλήσεις κατά τη διάρκεια της διαδικασίας ενσωμάτωσης, όσο και τα πρακτικά πλεονεκτήματα και πιθανούς περιορισμούς της εφαρμογής.

ABSTRACT

In recent years, Augmented Reality (AR) has gained ground in many fields like medicine, tourism, surgery, advertising, healthcare etc. with one of the most widespread being cultural heritage. There are several projects that utilize AR technology for adding new functionalities to the museum, enhancing the visitor's experience. Similarly, this thesis presents an AR application for mobile devices, the purpose of which is to enhance accessibility to cultural heritage places offering educational multimedia and interactive games to visitors. The app has been developed with Vuforia Augmented Reality SDK, Unity Engine and C# utilizing Image Recognition Technology and three-dimensional models. This implementation does not require an Internet connection, extra hardware or additional tags like QR codes in the museum. Therefore, not only does it make the user experience convenient, but also it facilitates the app integration to the museum. Moreover, we examine the case-study of the Archaeological Museum of Volos explaining challenges during the integration process as also practical advantages and possible limitations of the app.

CONTENTS

LIST OF ABBREVIATIONS

2D	**2**-**D**imensional
3D	**3**-**D**imensional
AR	**A**ugmented **R**eality
GPS	**G**lobal **P**ositioning **S**ystem
IDE	**I**ntegrated **D**evelopment **E**nvironment
QR code	**Q**uick **R**esponse **code**
RFID	**R**adio **F**requency **Id**entification
SDK	**S**oftware **D**evelopment **K**it
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
UI	**U**ser **I**nterface

CHAPTER 1

INTRODUCTION

Augmented Reality (AR) has become increasingly well known in research as well as in industry, recently. Suitable software and more efficient hardware have enabled researchers to pursue AR projects at much lower cost than some years ago. For example, in the last five years, smartphones involved all the required hardware that AR needs, making the final product more accessible to people. Trying to take advantage of it, our project combines these two technologies. Our application provides an AR experience to museum visitors through their smartphone. That means guests can have easy access to exhibits-related content without any special hardware.

1.1 What is AR?

First of all, we need some clarifications about AR technology and the necessary equipment. AR is an interactive experience of the physical environment the elements of which are mixed with computer-generated 3D models, texts, images, sounds, etc. It is also referred to as Mixed Reality or Computer-mediated Reality [1]. However, in this thesis, it will be mentioned only with the "AR" term for clarity reasons. Another word that adds confusion to this technological field is VR which stands for Virtual Reality. This term is completely different and it should not be confused with AR. Virtual Reality is an interactive experience of a virtual world which is totally computer generated and it doesn't involve any optical, sound or haptic stimuli from the physical world. Now, as the AR needs input from the real-world environment, it requires specific hardware and software. The hardware typically consists of a camera, a display and several movement sensors like accelerometer, gyroscope, GPS and solid state compass. As said previously, nowadays smartphones include all these. Generally speaking, the AR experience can be combined with additional equipment like AR headset. Furthermore, instead of a smartphone, someone can use solely smart-glasses for all AR functions. The AR software often relates to GPS navigation and also Pattern Recognition which can identify 2D or 3D objects like texts, images etc. in the physical environment.

Nonetheless, AR software depends greatly on the type of AR. It can be categorized according to the technological methods used. Although several taxonomies have been proposed [2], the five general AR categories [3] are: 1) Marker-less, 2) Marker-Based, 3) Location-Based, 4) Projection-Based, 5) Superimposition-Based. Firstly, Marker-less AR does not take into account the environment. It only adds computer-generated content with camera background. The second category is very common and this is the one utilized in our project. Virtual content appears in the camera based on specific markers in the physical world like images and QR codes. The next one called Location-Based AR uses GPS technology to estimate the user location in order to display the corresponding virtual content. It requires outdoor space because of the satellite use. Projection-Based AR is a little different because instead of a display, it works with a projector. As its name shows, it projects images to the real world, for instance, a virtual keyboard on a desk. The last type of AR replaces physical objects with virtual 3D models. One good example of that is a Superimposition-Based AR app which can show if a new refrigerator fits in the kitchen by replacing virtually the old one.

1.2 Application Areas

1.2.1 Examples

AR applications extend to numerous fields. These include the following.

- Advertising and Marketing. Products like shoes, clothes, furniture can be personalized to appear according to customer's desires.
- Classroom. Augmented objects like books, boards, posters can alter the learning process in schools. Teachers can interact with students via 3D learning material for example in biology class, students can observe a 3D human skeletal system.
- Construction. Engineers can see the 3D model of a building by aiming the blueprint with a smartphone camera.
- Healthcare and Medicine. AR can help doctors greatly by providing them with essential information during the diagnosis through AR glasses.

- Museums. Visitors' experience is enhanced using AR app through smartphone or tablets. Virtual informative labels and 3D models of exhibits improve the interaction of the museum tour.
- Tourism. Tourists can receive guidance info by targeting with the mobile camera a shop, a hotel, or a plaza.

1.2.2 AR and Cultural Heritage

AR technology is primarily based on computer vision, therefore movements of the targets or external stimuli like lighting changes can deteriorate its performance. One of the most suitable application fields for the current state of AR technology is cultural heritage. Especially museums provide a desired static environment with limited interventions from external factors like the sun or the weather. Thus, there is a variety of AR applications in museums. Firstly, guests can be provided with multimedia like texts, images, and videos. Not only does it provide more information, but also it boosts the interactivity during a museum tour. In addition, 3D models of exhibits could be displayed around the room. By providing the possibility for visitors to change the angle, the size or the lighting of these virtual exhibits, they can see a wide range of viewpoints. Another possible scenario is interactive games or quizzes which encourage people to explore various exhibit rooms.

1.3 Our Concept

In our concept, we wanted to provide museum visitors with an informative and engaging experience employing AR technology in a user-friendly way. Considering the widespread use of smartphones today by the everyday person, we turned into Mobile App Development. The final idea is to offer easy access to educational multimedia and interactive games for the museum's guests through their smartphones. The app can be downloaded through the museum's network or a personal one although its use does not need an internet connection. There are two options for the user: visit and game mode. During the first one, the user is informed via a museum's floor plan of where the augmentable exhibits are located. When one of them is targeted with the camera, multimedia buttons appear on the screen. They include exhibit's description, images,

audio, and videos. The user is also updated via the same museum map about which of these exhibits have remained unseen. Moreover, the user can join a treasure hunt game by choosing the game mode. There are virtual "coins" (i.e. rotating cubes that hover above the exhibits) all around the museum attached to several augmentable exhibits. The user's goal is to collect all the "coins" by tapping on the touchscreen. Each museum room represents a level. The game begins with only the first level unlocked and when the user collects all the room's coins, then the next level (room) will be unlocked. Therefore, there is a museum floor plan which informs the user for his/her score in each room. This map provides also the user with descriptions of the exhibits that he/she is looking for in each room. When all the "coins" are collected a winning text is displayed congratulating the user. The app utilizes several software tools and equipment. In more details, it uses image recognition software which provides the image target files and 3D models for the virtual objects. Moreover, it requires a smartphone camera, touchscreen for input and sensors like gyroscope and accelerometer in order to navigate around the virtual content. More technical details are included in the relative chapters.

In chapter 2, similar AR projects in museums are presented about their approach and the technology which they applied. In chapter 3, we introduce several development tools for AR and make comparisons based on the desired features of our project. After that, we also present the utilized software and hardware. In chapter 4, the application of our project is explained in detail describing both the design and the development. This chapter includes also screenshots and code snippets which help to understand deeply our concept. Chapter 5 is dedicated to the case-study of the Archaeological Museum of Volos in which our app was tested. Last but not least, in chapter 6, there is a discussion about the pros and cons of this implementation, as also possible improvements with future technology.

CHAPTER 2

RELATED WORK

2.1 Other Museum Projects

AR tools can be utilized to enhance accessibility to cultural heritage places. We present related projects and studies in order to give a general idea about the possibilities which AR has to help museum visitors and the stage that is currently developed. In general, most of the museums are static environments with one-way interaction. However, AR can alter the visitor's experience rendering it more dynamic and engaging. We summarize several other projects in the same field, stating their goals and the utilized tools, as also 2 studies which examined the impact of AR systems on the user's behavior.

Firstly, "The Speaking Celt" [4] is a similar project which presented an AR app that helps smartphone users during a museum tour. Display cases were located near exhibits in order for visitors to target them with their smartphones. In this way, animated 3D figures appeared on the screen. These figures could speak to the user giving them information and guidance for each artifact. The app was developed with Wikitude's Augmented Reality SDK (Image Recognition and 2D Tracking Technology) for Android and iOS. The app code was basically written in Javascript and also several 3D development tools were used for the animated avatars.

Similarly, this implementation [5] used image targets like cards, in order to visualize 2D and 3D models of exhibits and also related metadata nearby. The 3D figures were rotating continuously, therefore the user could view them from different angles. The user could also use a special marker to remove a virtual exhibit from his/her environment. These AR functions were implemented with ARToolkit (open source AR SDK, now acquired from DAQRI), OpenVRML libraries while the images and videos were captured with a webcam.

From this paper [6] published in 2008 we can see the difference in the equipment that was required several years ago compared with the current one which can be a single smartphone device. In detail, an AR prototype was presented with 2D tracking for museum paintings offering a multimedia guide. Its AR system was developed with the MAGIC Engine software framework, built above of OpenCV for video capturing, ARToolkitPlus for 2D tracking and OGRE3D for the insertion of virtual figures. The

hardware components consisted of an Ultra Mobile PC and an ordinary webcam making it a little bit inconvenient for users to carry the system during their visit.

Other projects like this one [7] have tried to go beyond the marker-based display of 3D content adding more interactive features. The user could actually resize, rotate and reset the scale and position of the virtual figure. In more detail, special markers on a piece of paper were used as virtual buttons enabling the above functions, when the user puts his/her finger upon them. In essence, it required the user to hide a major part of the marker. This system needed a camera, speakers and a handheld laser scanner for the reconstruction of 3D models. The software was developed with ARToolKit which was preferred because of its high-level functions.

An interesting way to enhance AR experience is by personalization. This paper [8] explained how AR can be used as a means of interpretation, unique for each visitor. Not only was AR proposed to provide augmentations visually and acoustically for each exhibit, but also for tracking its impact on the user in order to adapt itself. There were three types of sensors: visual, acoustic and biosensors. The user's attention point, environmental noises and the psychological state of the user were analyzed continuously. Thus, the output of the AR device could be reformed based on the user's interests. Finally, this concept included see-through AR glasses with a speaker for 3D sound.

Furthermore, there are approaches like this one [9] in Acropolis museum in which researchers designed an AR system utilizing computer vision for markerless 3D. This method creates a 3D map of the surrounding area from captured videos making the experience more engaging. The augmentations in this project can reconstruct the physical view with a virtual one, recreate mythological appearances and highlight interesting details and annotations. This system was implemented with a server-client architecture using wireless communication. A mobile AR framework for iOS and HTML5, CSS, JavaScript for the user interface were also employed.

A similar approach to the previous one is this mobile AR architecture [10] which utilized 3D object tracking. The user could target with his/her smartphone the physical exhibit and receive related content e.g. texts and images. This idea was implemented as a mobile application for iOs with native development platforms.

Moreover, these papers [11] [12] examined how AR tools can enhance the user's experience of physical exhibits. In the first one, researchers conducted experiments on how two types of different activities based on AR guide can change the visitor's experience. They found three categories of interaction between the visitor and the device. The second paper investigated the way AR tools can contribute through multimedia to the learning process. The findings included that participants spent more time interacting with an exhibit when that was augmented. Studies like these can indicate new ways of designing AR systems for museum visitors in order to enhance learning and engagement with exhibits.

In conclusion, there were various approaches presented such as 2D or 3D tracking, personalization of the AR system through sensors and interaction with the virtual figures. The latter two were innovative features which could improve AR experience remarkably. On the contrary, the presented projects have some disadvantages. For example, some of the 2D tracking systems required extra tags near the exhibits and this can interfere with the aesthetics of a museum. In addition, one of the 3D tracking systems required additional equipment i.e. scanner which increases the budget. Another one used wireless communication which can be a limitation if the museum doesn't have a strong signal in all needed rooms. Considering software, iOS-related SDKs and the open-source ARToolkit were commonly used among these implementations. Finally, further research is needed to examine how new tools like AR headsets or other approaches like games can be combined with the above AR project ideas to enrich museum experience.

CHAPTER 3

DEVELOPMENT TOOLS

3.1 Comparison

One of the most common ways for AR app development is the use of one main platform like Unity Engine which helps the developer to manage 3D models, create the user interface, write scripts and also import an SDK for the AR functions. In this project, the Unity Engine was used due to our prior experience and familiarity with it. On the other hand, there are several AR SDKs for different kind of projects with each one having its assets and limitations. In order to choose the most appropriate for the museum project and Unity compatible, seven of them were examined. The information has been derived from two main web pages [13] [14] and the official site of each SDK which are provided below. In the following paragraphs, the seven SDK are presented with its main features and afterward, there is the comparison table 3.1 which is mainly based on the desired features for our museum project. It also contains a total rating for the documentation, the forum and existed tutorials of each SDK according to our personal research and experience. This rating indicates the ease of use for the SDK which also includes research for solving technical issues.

In order to present the SDKs, some clarifications are necessary to be mentioned about the AR features. The most common ways to augment reality is by image targets. By taking a picture, or by 3D scanning an object via a special app, we can create the targets (2D or 3D respectively). Some SKDs can recognize multiple image targets in case of tracking simultaneously several targets with the camera. There is also the “Extended tracking” which creates new markers from the surroundings using the initial marker as a benchmark. Thus detection and tracking can be accomplished from a further distance, and this enables the user to move freely around the marker. The recognition process is taking place often in a cloud server or locally in the device. Some SDKs with local recognition give the opportunity to the user to create in real-time image targets while he/she is running the app. Another AR feature is the understanding of the environment. The two most common ways to interact generally with the environment is the SLAM technique and the Plane detection. SLAM stands for simultaneous localization and

mapping, a process whereby a device can create a map of its surroundings. Thus, the app can add 3D models which interact with the physical objects, for example, a virtual walking cat can avoid the furniture. In a similar way, plane detection can only identify the flat surfaces of its surroundings like a doll sitting in a chair or hitting a wall. Other features are the light estimation, QR code scanner and virtual buttons. These will be explained in the SDK paragraphs.

I) ARcore

ARCore [15] is a software development kit developed by Google that provides augmented reality applications. This SDK is relatively new as the initial release was in March of 2018. It provides a free AR SDK, free Unity plug-in and it supports only Android devices. ARCore involves several AR features which are: 2D recognition, real-time image target creation, plane detection, SLAM technique and light estimation. The last one allows you to capture the current light conditions in the physical world and apply them to the virtual objects. However, there are some major limitations. Firstly, It requires Android 7.0 or later. ARcore is also compatible with a limited number of smartphones [16]. Finally, the learning material for someone who wants to begin developing with ARcore is relatively limited. More specifically its forum has very limited topics and existed tutorials are just enough to learn the basics.

II) Catchoom

Catchoom [17] offers a free trial and Premium for Startups, Professionals, and Enterprises. It also provides a free Unity plug-in and supports iOS and Android devices. There is only Image Recognition which it can be local or on the cloud server. The learning material covers a basic level. Forum topics started in 2015 and tutorials for beginners are available.

III) EasyAR v2.2.0

There are two editions of the EasyAR SDK [18]. The “Basic” edition is free and the “Pro” requires payment. It also offers free Unity plug-in and supports Windows 7 (and above), Mac OS X, Android 4.0 (and above) and iOS 7.0 (and above). EasyAR has a variety of AR features. In the free edition, it involves 2D recognition (in the Cloud), Multi-target recognition and QR code scanning. QR codes are recognized as image targets but they can

also provide additional data to the app like id, name, item description etc. Moreover, in the “Pro” edition, SLAM technique and 3D recognition are offered as extra. Although, the forum topics have started in 2017 and the existed tutorials are very limited thus makes it difficult for beginners to start using it.

IV) Kudan

Kudan [19] offers free AR SDK (with free basic license) and free Unity plug-in. Business and enterprise have a paid license option too. Kudan provides 2D, 3D recognition and also the SLAM technique. Its learning material is limited as the forum topics started in 2017 and the existed tutorials are few.

V) Maxst 3.5

Maxst [20] offers free AR SDK with a free basic license and free Unity plug-in. There are also three paid editions for professionals and enterprises. It has a variety of AR features which are 2D, 3D recognition (in the Cloud), SLAM technique, instant tracking which allows contents to be laid on a horizontal plane without mapping the surrounding space. It provides also a QR/Barcode scanner and Marker tracker (with special Maxst images) which actually belongs to the 2D recognition feature. Finally, its forum has topics started in 2017 and there are also several basic tutorials available.

VI) VoidAR v0.1.h Beta

VoidAR [21] provides a free basic license (free AR SDK) and paid license for business or enterprise. It has free Unity plug-in and supports iOS, Android, MacOS and Windows platforms. The AR features offered by VoidAR are 2D recognition (in the Cloud), Multi-tracking and SLAM technique. Unfortunately, there is not an available forum and the existed tutorials cover only the basic level.

VII) Vuforia 7

Vuforia [22] offers free AR SDK and free basic license (non-commercial purpose). There is also a paid license for companies with three editions: Classic, Cloud and Pro. Furthermore, it provides a free Unity plug-in and supports Android, iOS and Windows 10 platforms. Vuforia has numerous AR features including 2D, 3D recognition, extended tracking, multiple targets, marker tracker (called VuMarks), plane detection and virtual buttons.

The last one allows the user to press a virtual button by hiding a defined part of the image target from the camera with his/her hand. Vuforia has forum topics started in 2015 and also there are many available tutorials for all the AR features it supports.

	2D Recognition	3D Recognition	Documentation/ Forum/Tutorials Rating	Environment Tracking	Unity plug-in	Pricing - License	Main Limitation
ARcore	Yes	No	4/5	Yes (planes)	Yes	Free	Compatible with a limited number of devices
Catchoom	Yes	No	3/5	No	Yes	Trial/Paid	-
EasyAR	Yes	No	2/5	Yes (SLAM)	Yes	Free/Paid	-
Kudan	Yes	Yes	3/5	Yes (SLAM)	Yes	Free/Paid	-
Maxst	Yes	Yes	2/5	Yes (SLAM)	Yes	Free/Paid	-
VoidAR	Yes	No	3/5	Yes (SLAM)	Yes	Free/Paid	-
Vuforia	Yes	Yes	4/5	Yes (planes)	Yes	Free/Paid	No real-time target capturing

Table 3.1 Comparison table for AR SDKs

3.2 Utilized Tools

In this subchapter, we present all utilized tools during the app development including both software and hardware. The Vuforia 7 SDK was finally selected because of the available learning material, the excellent 2D image tracking and the variety of other AR features.

- **Vuforia Cloud Recognition:** On the website of Vuforia, there is a Developer Portal. There, in the Develop tab, there are the Target Manager and the License Manager.
  - Target Manager: Here we can upload images of our targets and manage our targets database. There is a rating for every uploaded image informing us on how much detectable that image is. Finally, we can download our database as unity package and import it into Unity Editor.
  - License Manager: In this window, a license for our app can be created and all of its info and usage data can be displayed.

- **Vuforia 7 Augmented Reality SDK:** It offers Unity prefabs i.e. prefabricated entities like Image Target, Multiple Target, AR Camera, Virtual Button etc. which are explained in more detail in the subchapter 4.2 called “Unity Project”. In addition, it provides libraries and other functionalities used for trackable objects and the AR camera.
- **Unity 2017.3.1f1:** Unity is a cross-platform game engine created by Unity Technologies, which is primarily used to develop 3D and 2D video games and simulations for computers, consoles, and mobile devices. It is also suggested for AR app development as it can be expanded with Unity packages and SDKs. Unity offers a feature-rich and highly flexible editor called Unity Editor with which we can develop and manage our project.
- **3D models:** Free or paid 3D models can be found in the Unity Asset Store or in related websites.
- **MonoDevelop:** MonoDevelop is an open source integrated development environment for Linux, macOS, and Windows. All scripts of this project were written with MonoDevelop.
- **C#:** Unity supports three scripting languages, C#, UnityScript, also known as JavaScript for Unity, and Boo. All project’s scripts were written in C# as it is the most widespread in unity documentation and tutorials.
- **Smartphone camera:** Sony Xperia Z3 Compact was used to capture images which were utilized as image targets during the development and testing process of this project. Its camera has 20.7 megapixels.
- **CamScanner app:** In order to best capture a flat surface, an image scanner app was used through a smartphone. CamScanner utilizes smart cropping and also enhances both saturation and contrast of captured images.
- **Operating System Windows 8.1:** Unity Editor and MonoDevelop were installed and running in Windows 8.1.
- **Smartphone:** The application was being tested throughout the development process using a Sony Xperia Z3 Compact.

CHAPTER 4

THE APPLICATION

4.1 Design

This application was implemented in order to prove the concept of providing museum visitors with multimedia and interactive educational games via their personal smartphone. The application design is simple and user-friendly. It includes button icons for convenient exploration and helping texts for user guidance. When the app is launched the main menu is displayed and the user has two options (modes) to choose from, as also figure 4.1 shows.

Figure 4.1 Main Menu

4.1.1 Multimedia

The first option in the main menu is called “Visit Mode” and provides multimedia for the user. Figure 4.2 shows the main screen of this mode. In the upper section of the screen, a text called “Help Text” is displayed which indicates to the user the purpose of the camera.

In the lower section of the screen, there are two buttons, the “Back Button” and the “Map Button”. The user can return to the main menu by pushing the first one. The “Map Button” shows the floor plan of the museum. All augmented exhibits are indicated with yellow color and they blink. On the other hand, grey spots are exhibits that user has already aimed at with the camera. These data remain even when the user closes the app, so the tour can be continued in another visit. The map is shown in figure 4.3.

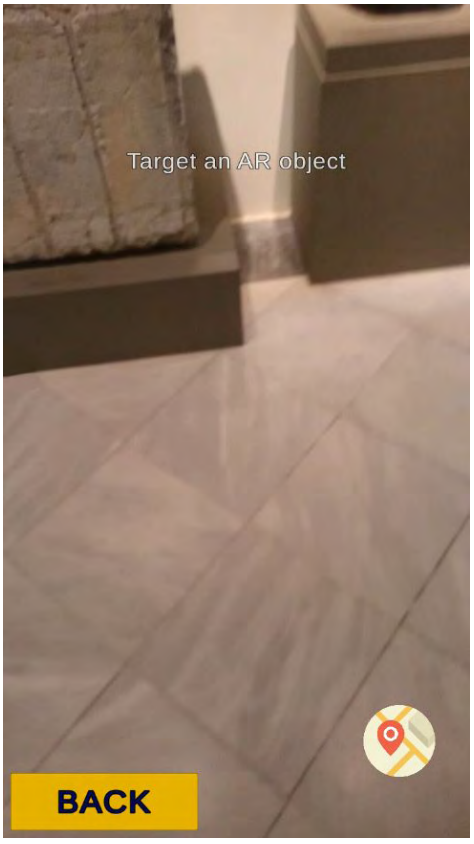

Figure 4.2 Main Screen in Visit Mode

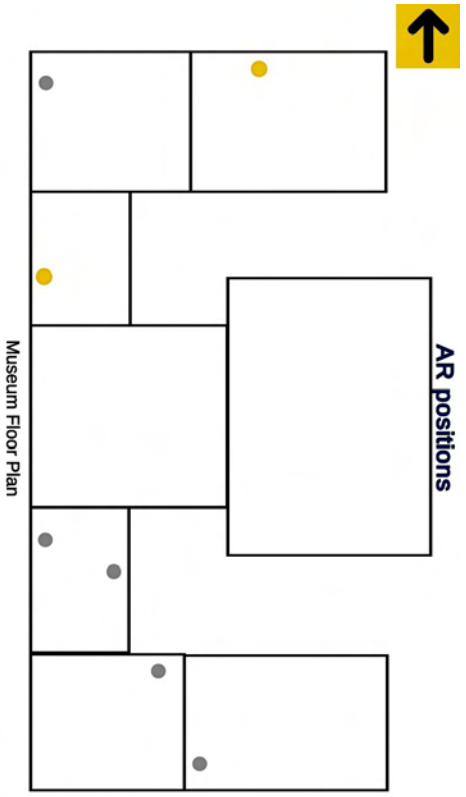

Figure 4.3 Museum Floor Plan in Visit Mode

Now when the user is aiming with the camera to an augmented object a multimedia window with related content shows up as shown in figure 4.4. There are four options there. For each button, we provide a brief explanation and an app screenshot. The first button (far left) refers to text, which shows the title and description of the exhibit.

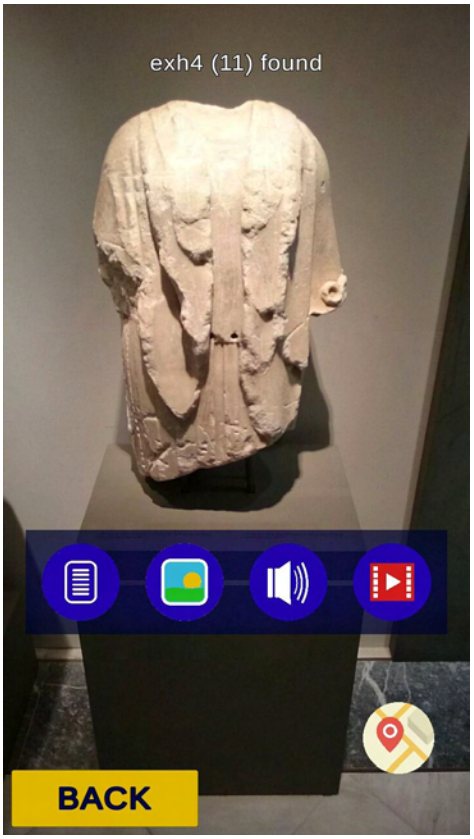

Figure 4.4 Multimedia Window

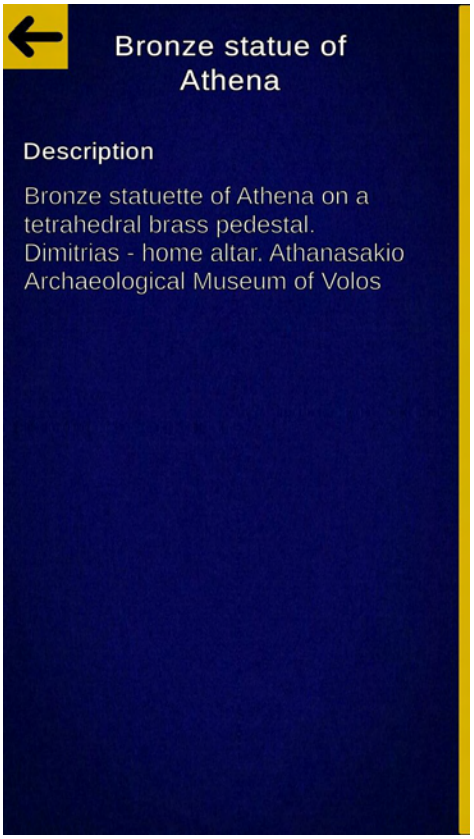

Figure 4.5 Text Panel

Secondly, the “Images Button” shows exhibit-related photos from many different angles or informative posters. In future work, it may include photos from the excavation site or images of related artifacts in museum storages. After that, the audio includes recorded speech with relative information for the exhibit. These are depicted in figures 4.6, 4.7 and 4.8 respectively.

Figure 4.6 Images Panel

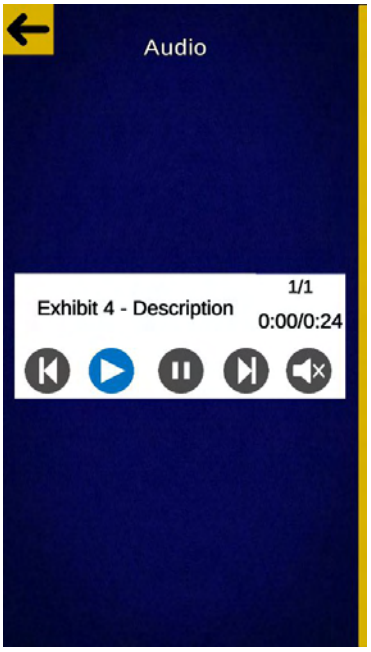

Figure 4.7 Audio Panel

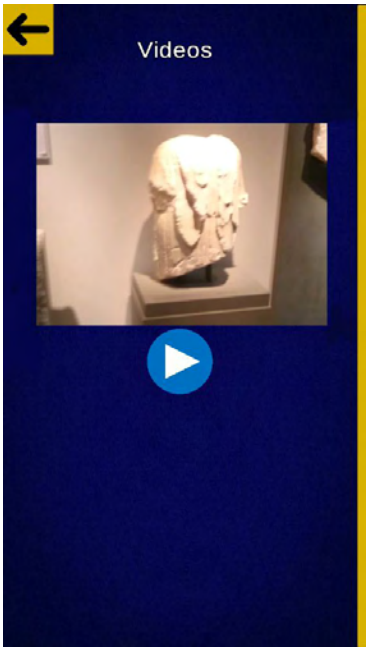

Figure 4.8 Videos Panel

4.1.2 Interactive Game

The second option is called “Game Mode” and it’s a treasure hunt game. Several exhibits have been augmented with a "coin" (rotating cube) near them. The user collects the "coins" using the touchscreen. The museum’s rooms represent the different levels of the game. When the "coins" of a room are all collected, then the next level (room) will be unlocked. This is the main concept of the game. By selecting the Game Mode and after passing the initial guide text, the camera is enabled as shown in Figure 4.9.

In the upper section of the screen, we see a help text for the user stating the purpose of the camera. Moreover, in the upper left corner, there is a counter that indicates the number of collected coins. In the lower section of the screen, there are two buttons, the “Back Button” and the “Map Button”. The user can return to the main menu by pushing the first button. The “Map Button” enables a panel with the museum’s floor plan which in this mode is a little bit different. As previously stated, each room represents a level. Thus, only the room 1 is unlocked at the first time the app is launched as figure 4.10 depicts.

In this map, we can see how many "coins" are included in each room and how many of them we have collected. Each yellow circle is a button which gives some information about the exhibits that are inside this room and also have a "coin" attached to them. In

figure 4.11, an example of the list of these exhibits in room 1 is presented. It is worth saying that these data are preserved even if the user closes the app. Thus, the user can resume his/her game another time. However, there is a “Reset Button” colored grey in the middle room in case the user wants to reset his/her score.

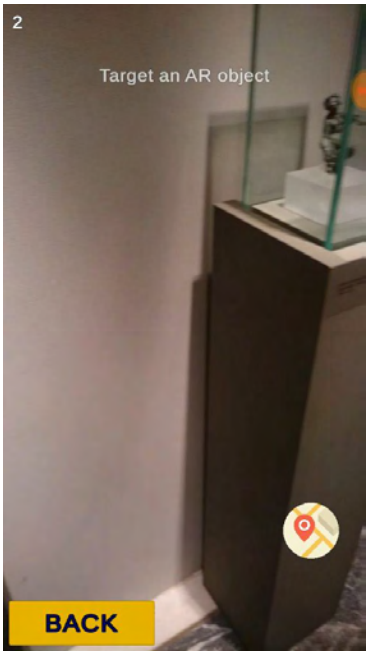

Figure 4.9 Main Screen in Game Mode

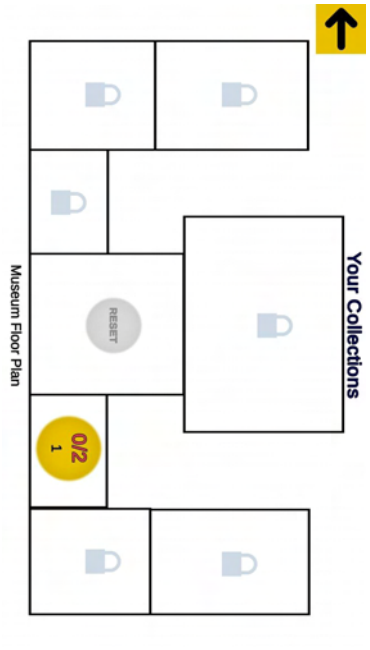

Figure 4.10 Map in Game Mode

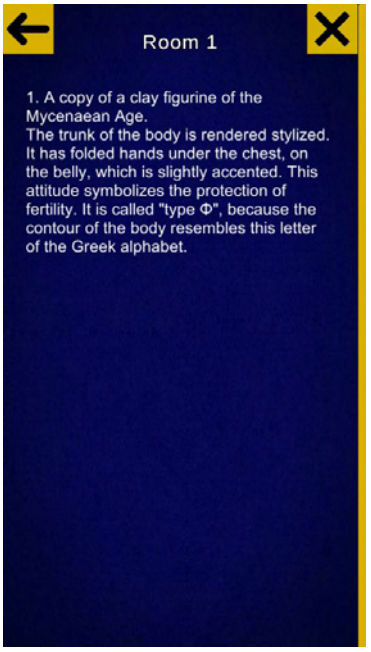

Figure 4.11 Room 1 - List of exhibits

Now when the user aims at an AR object, a rotating "coin" will appear as shown in figure 4.12. The user can pick up this "coin" by tapping on it via the touchscreen.

Figure 4.12 "Coin" appeared

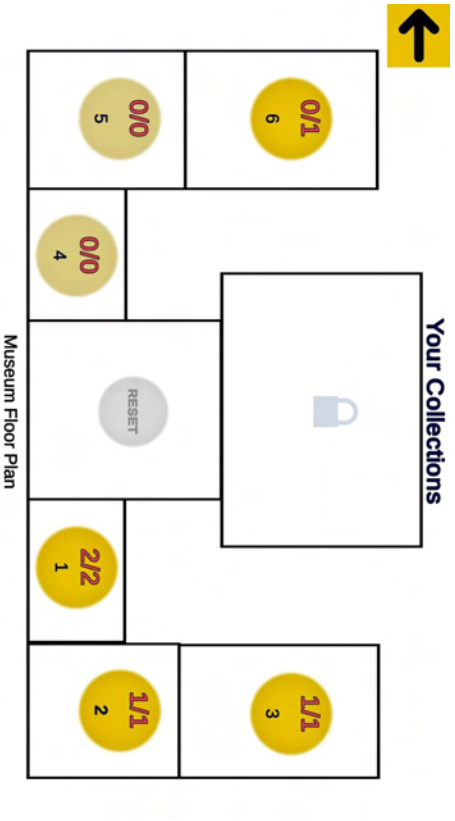

Figure 4.13 Until room 6

After collecting the "coins" from the first room, we unlock the next one. If a room includes no "coins" then we will unlock the next room that contains at least one. For example, figure 4.13 shows this case where room 4 and 5 include no coin. At the end where all "coins" are collected, a winning text shows up in the middle of the screen. This process of treasure hunt increases the user's motivation and it helps him/her to enjoy the tour.

4.2 Unity Project

4.2.1 Project Setup

The first part of the setup is downloading and installing the required software. Several software components are needed for the setup of this project. The following table provides their names with each download page.

Unity Engine (Version 2017.3.1f1)	https://unity3d.com/unity
Vuforia 7 SDK	https://developer.vuforia.com/downloads/sdk
Android SDK	https://developer.android.com/studio/#command-tools
JDK	http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

Table 4.1 Software and download pages

After downloading and installing all the necessary software they must be connected with Unity. Vuforia 7 has been integrated into Unity so it needs no further action. However, if there is no Vuforia component in the Unity Editor we can import it by clicking on the menu “Assets -> Import Package -> Custom Package” and selecting the downloaded Vuforia package. In order to connect Android SDK and JDK with Unity we must click (in Unity Editor) on the menu “Assets -> Preferences -> External Tools”. In the lower section under the title Android, we can fill the paths where these components are installed.

The second step is the configuration of Build Settings. With Unity Editor opened, in “File->Build Settings”, there are several settings that need to be configured before building the application. First of all, in the upper section called “Scenes In Build”, we must drag and drop our scenes. In one of the next paragraph, we will see in more detail the three scenes used in this project.

Now in order to build an Android app, we have to make sure that the appropriate module is loaded. We can find it by clicking the Android option on the left list. Then, we click the button to load the module if we have not already. We should also switch to the Android platform by having selected the Android option from the left column and then we click the “Switch Platform” button.

Furthermore, we click the “Player Settings” button for more changes. In the upper section, we can change the company name and product name. After expanding the other setting tab, in the section with title Identification, we fill the package name, version, and the minimum API level. In Player settings, we can also change the app icon and the screen orientation.

Before completing the setup process it would be helpful for us to import the “TextMesh Pro” package from the Unity Asset Store. This provides us with more detailed and clear UI Texts.

Last but not least, we have the Vuforia Developer Portal [23]. In the License Manager, we have to register in order to create a license key. After that, we can create a license key by clicking “Get Development Key”. Now we need to copy and paste the license key to Unity Editor. In the upper menu of the editor, under “Vuforia -> Configuration”, a field called App License will appear in the Inspector window. We must paste our key there. Figure 4.14 shows the webpage of the License Manager after login.

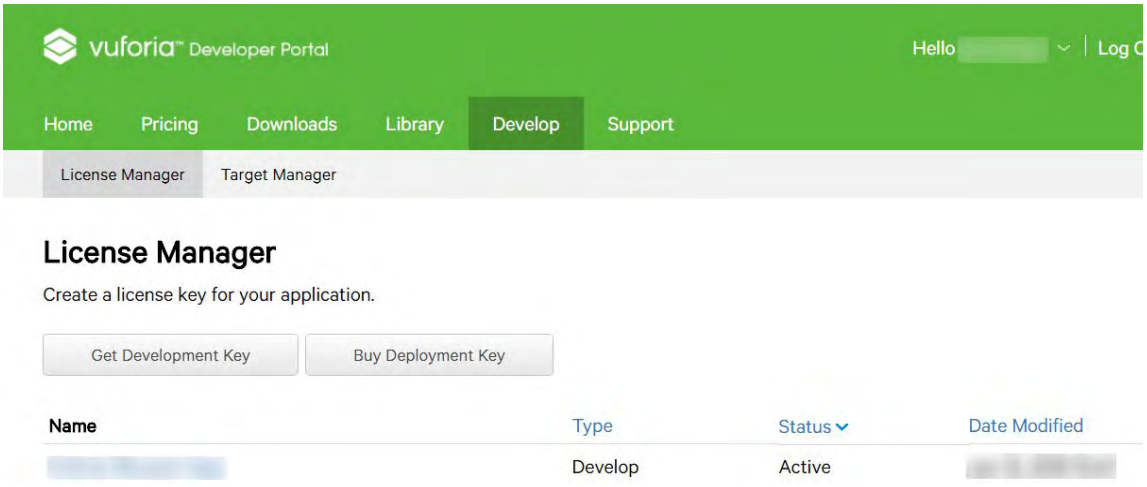

Fig 4.14 License Manager

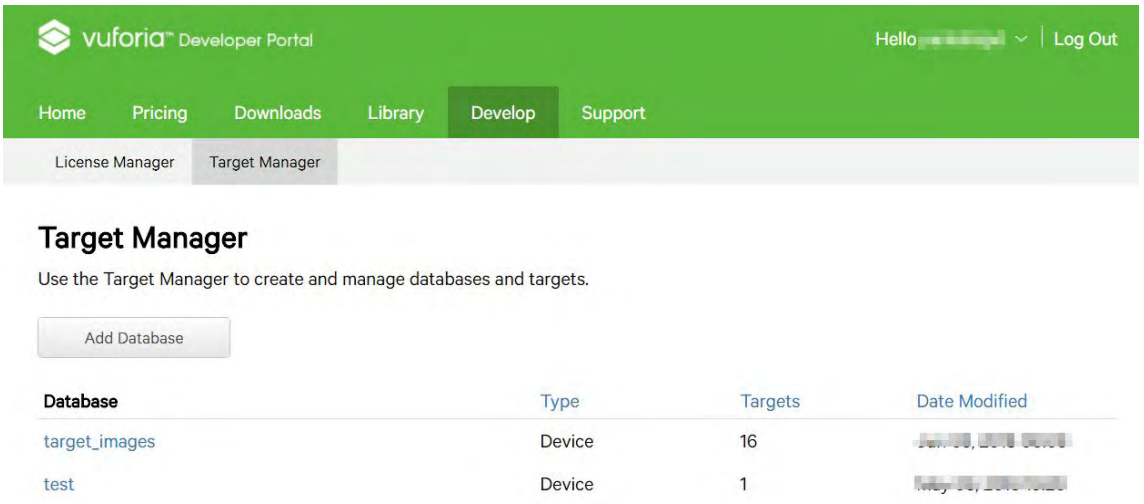

Figure 4.15 Target Manager

In the Target Manager, we click “Add Database” for creating a new image target database. Then we fill the name and also select “Device” type. For adding new targets we click the button “Add Target”. Now we need to select image type and upload our image. (For our project, we used only single images. This type is very useful even for simple 3D objects.) In addition, a very helpful feature is the rating for each image. This shows you how much detectable the uploaded image is. By clicking on a target you can also see the button “Show features” below the image. This button depicts the main points that are used for the recognition of that image. The main webpage of the Target Manager is displayed in figure 4.15.

Finally, the database with the image targets can be downloaded by clicking the button “Download Database” and choosing “Unity Editor” in the popped up window. Now we can import the database to the project by clicking “Assets -> Import Package -> Custom Package” and selecting the downloaded database. One last step is to check that the new database is on. To do this, go to “Vuforia -> Configuration” and in the Inspector Window make sure that your dataset is activated under the title Datasets.

4.2.2 Scenes

Unity scenes are different environments of our app with a separate hierarchy of objects. In this project, there are three scenes located in “Assets/Scenes”. They are the following: 1) Menu: It includes the main menu where the user chooses one of the two modes and goes to the corresponding scene. 2) VisitScene: The main goal of this mode is to provide multimedia for the user. More specifically, text, images, audio, videos are displayed when the user aims with his/her camera a museum exhibit. 3) GameScene: The main goal of the game is to motivate the user to search for specific exhibits in each room. The user collects virtual objects like "coins" attached to the exhibit and in this way he/she can continue to the next room.

4.2.3 Objects

Figures 4.16, 4.17 show the Hierarchy window with all created objects (i.e. entities of our app) of Visit and Game scene respectively. In the following paragraphs, each object will be explained in more detail about its purpose and its main components i. e. attached scripts.

Figure 4.16 Objects in Visit Mode

Figure 4.17 Objects in Game Mode

I. AR Camera (In Visit & Game Mode)

This is a Vuforia Prefab the location of which is the folder “Assets/Vuforia/Prefabs”. These are prefabricated assets and can be dragged and dropped to the Hierarchy window to create corresponding objects. The AR Camera object is attached with all necessary scripts in order to enable camera and track targets. In this project, the AR Camera has been attached with two scripts called “TouchInput.cs” which relates to Pick-Up Objects and “CameraFocusController.cs”. Both will be explained in more detail in the main scripts paragraph.

II. UI Panels (In Visit & Game Mode)

This category consists of several panels which are full-screen windows. They include multimedia panels (Text, Images, Audio, Videos), the guide, the museum floor plan, the exhibit list for each room and of course the main UI panel which has been laid above the camera. Each panel is a child of a Canvas object. Let's elaborate on each one to understand more about them. Firstly, the Guide Panel shows an explanatory text about the game. The Main UI Panel is laid above the camera and contains the Back button, the Map Button and the helping texts like the "coin" counter. Another important panel is the Museum Floor Plan Panel. Here the user can have an overview of the museum and his/her collected "coins" (in the Game Mode) or the augmented exhibits in each room (in the Visit Mode). In the Game Mode, the user can click on an unlocked room to see a list with the exhibits that have a coin. These lists are displayed in the seven Exhibits List Panels, one for each room. On the other hand, the Visit Mode has its unique panels too. These are the Multimedia Panels. There are four different types of multimedia panels which are Texts Panels, Images Panels, Audio Panels and Videos Panels. Each exhibit has a different set of multimedia panels related to its content. More specifically, a Text Panel is shown in figure 4.18 and the related objects in figure 4.19. There is the exhibit's title and a brief description which are inside of a UI Scroll View Object.

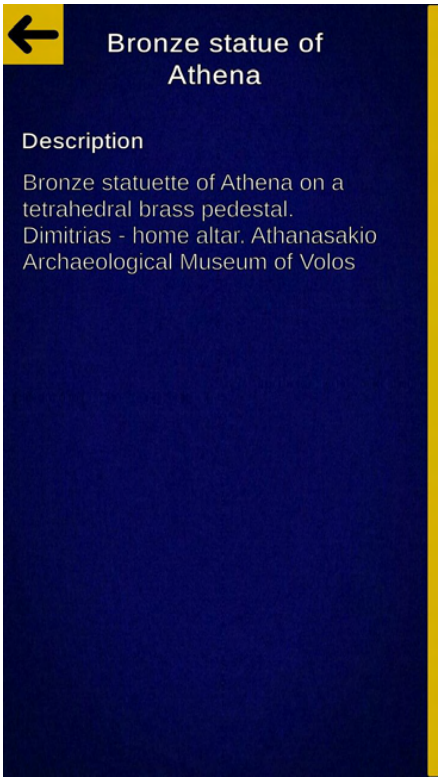

Figure 4.18 Texts Panel

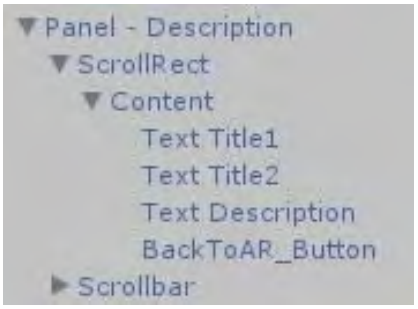

Figure 4.19 Texts Panel Objects

Images Panel consists of different UI Images. These images are inside of a UI Scroll View object. Figures 4.20 and 4.21 show respectively the Image Panel and its objects.

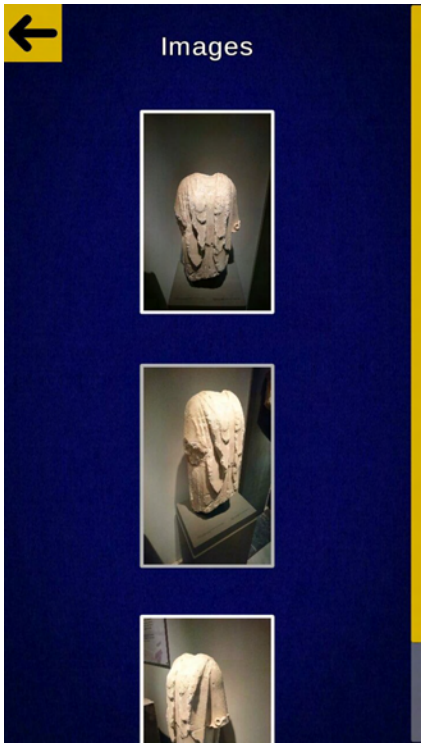

Figure 4.20 Images Panel

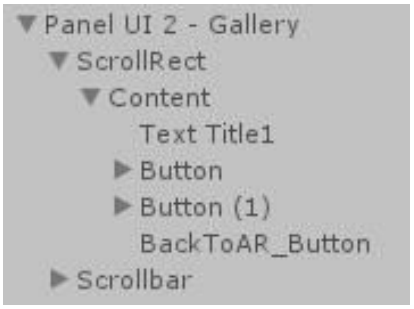

Figure 4.21 Images Panel Objects

Audio Panel includes an Audio Player. The two most crucial objects are the AudioManager and the AudioPlayer. The first one has a script component of the same name that takes the necessary audio clips and UI Texts like Time, Track Title and Track Number. It also has the Audio Source Component. The AudioPlayer object has many children (objects) which are either UI Buttons like Play, Pause, Next, Mute or UI Images showing the track info. All these objects are shown in figures 4.22 and 4.23.

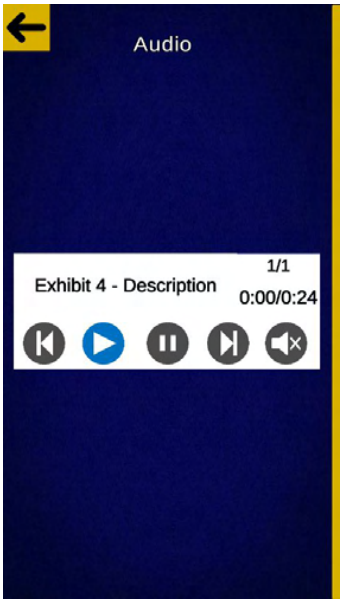

Figure 4.22 Audio Panel

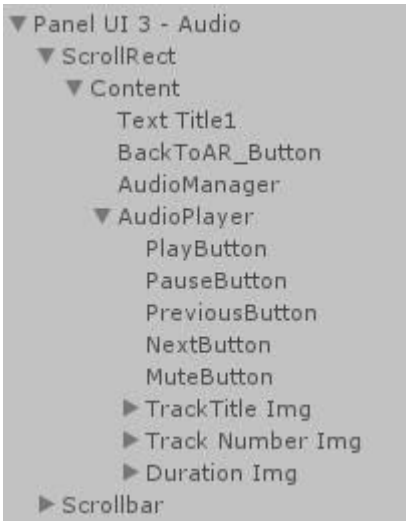

Figure 4.23 Audio Panel Objects

Videos Panel has one or more videos. Each video has its own Video Player. Thus, every video object has four main children. These are a Camera, a UI Raw Image, a UI Image for a video thumbnail and a Play/Pause Button. The camera object must have a Video Player component which takes the video file. This component should have the Render Mode selected to the “Render Texture” option. As you see, a Render Texture is needed here and so does in the UI Raw Image in the component of the same name. This Texture can be created in the project window with a right click and select “Create -> Render Texture”. This texture must be dragged and dropped to the two components mentioned above. In figures 4.24 and 4.25, we can see the objects in a Videos Panel.

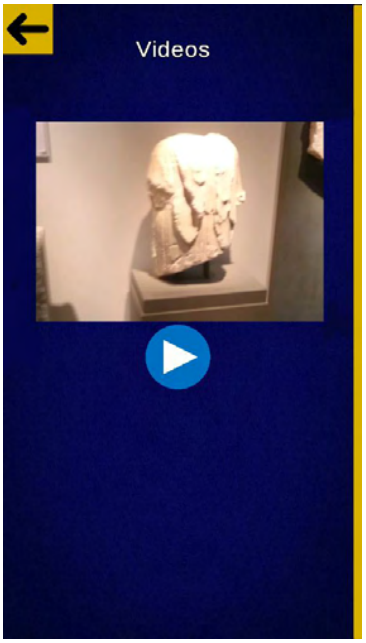

Figure 4.24 Videos Panel

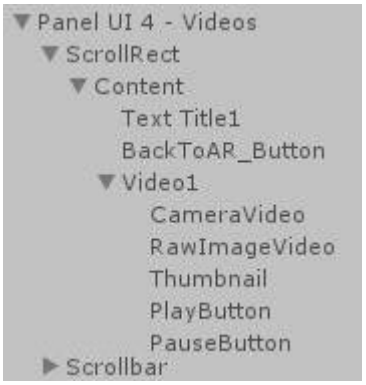

Figure 4.25 Videos Panel Objects

III. UI Texts (In Visit & Game Mode)

All main panels include UI Texts. These can be exhibit's titles or descriptions (in Multimedia Text Panel). They can also be indicators like guide text (in Guide Panel), help text, the "coin" counter (both in UI Panel) or the number of collected "coins" in a room (in Main UI Panel).

IV. Multimedia Buttons (In Visit Mode)

There are four Multimedia Buttons: Text, Images, Audio, and Videos. Each one triggers the corresponding Panel. Only when the camera is aimed at an AR object, the multimedia buttons are activated as figure 4.26 displays.

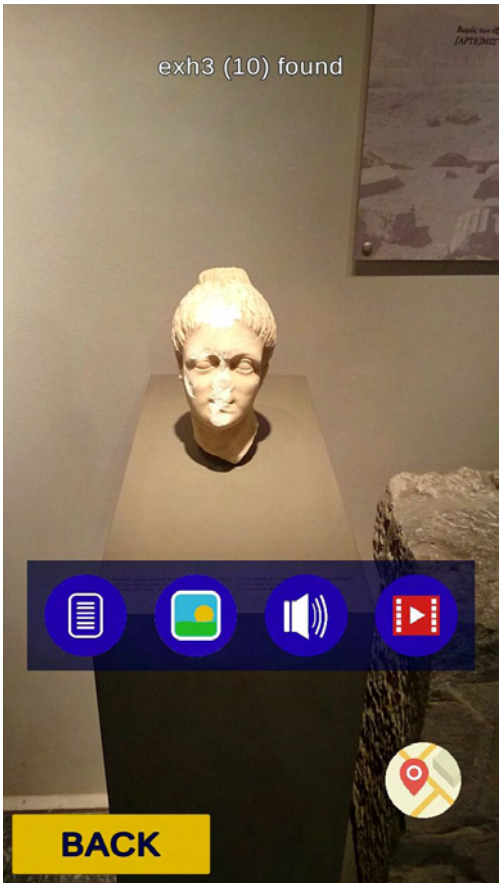

Figure 4.26 Multimedia Buttons

V. Other UI Buttons (In Visit & Game Mode)

In both modes, there is a Map Button in the main UI Panel. It enables the Museum Floor Plan Panel. Similarly, there is a Back Button in the main UI Panel in both modes. By clicking it, the Main Menu shows up closing the previous scene. Most of the UI Buttons utilize the Unity script “Button” which has an option to add an action when the button is clicked. In some instances like the Audio Buttons, scripts were written to implement the “on click” complex actions.

VI. Score Controller (In Visit & Game Mode)

This object is an empty object the purpose of which is to host the script with the same name. Thus, it controls the score, meaning the collected "coins" and the unlocked rooms. More details are included in the script’s paragraph of the subchapter 4.2.4.

VII. Pick-Up Objects (In Game Mode)

These are cubes which represent the collectible "coins" and they exist only in the Game Mode. Each one is child of an image target. This enables them to be displayed in the user's screen when he/she targets the corresponding exhibit. In addition, we created a layer called "Touch Input" from the upper right corner of the inspector window of a pick up object. After that, we set the layer to "Touch Input" in each pick up object. This facilitates the search of touchable objects from the "touchInput" script. They have two script components as shown in figure 4.27

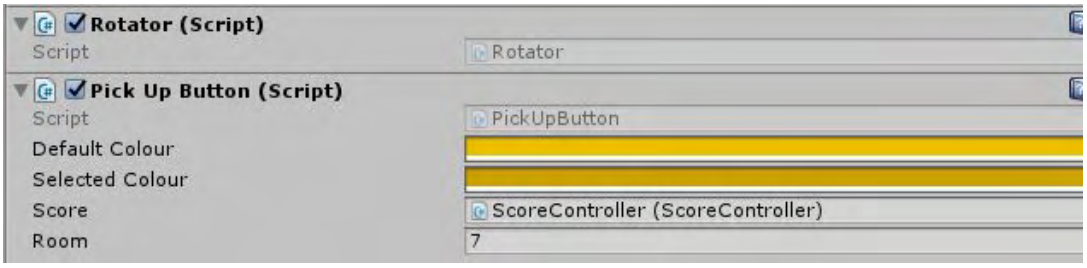

Figure 4.27 Pick-Up Object - script components

The first component is called "Rotator.cs" which enables them to rotate. The second one is called "PickUpButton.cs" and renders them collectible via the touchscreen. The last parameter of this script is the public variable "Room" which determines the room that this "coin" belongs to.

VIII. Image targets (In Visit & Game Mode)

The targets are grouped using an empty object as shown in figure 4.28 for better management. Each image target has as child the objects that we want to appear when this image is targeted with the smartphone camera. These objects will appear in the user's screen, according to their position relative to the image target in the Scene View. We have moved the children objects in Scene View just above their image target.

Figure 4.28 Image Targets

Vuforia has a prefabricated asset called Image Target which is located in Assets/Vuforia/Prefabs. Generally, a prefab works like a template. By dragging this asset to the hierarchy window you can create an image target object. Having selected this new object, in the Inspector window, in the component Image Target Behaviour you can define your dataset and your image target. In this project, many target objects have a Pick-Up object as a child in order to be displayed whenever the image target is detected on the camera. Figures 4.29 and 4.30 show this case.

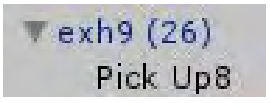

Figure 4.29 Image Target and Pick-Up Object

Figure 4.30 Pick-Up Object in camera

4.2.4 Scripts

In this subchapter we explain some of the main scripts used for the unity project. All of the source code is written in C#. First of all, it is necessary to mention that unity scripts contain two functions from the time they are created: The first one is “Start” and the other one is “Update”. “Start” is called on the frame when the script is enabled just

before any of the Update methods are called the first time. “Update” is called in every frame. Therefore, our code as we will see below is mainly written inside these two functions. In the following paragraphs, we explain the functionality of the main scripts presented in an order that facilitates the reader’s understanding.

- TouchInput.cs :

  This script is attached to the AR Camera object and its purpose is to let the user interact with the virtual objects via the touchscreen. Let’s examine the code snippet given in the figure 4.31. In lines 27 until 48, for each touch input, we find the touched object which must belong to the touchInputMask layer. Depending on the touch phase, we call the corresponding function from the “PickUpButton” script which is explained in the following paragraph. Moreover, there is one extra case of touch phase to consider. That is when the user presses down the button, moves the finger outside the object’s region and then he releases it. Thus, we have created two lists which the first contains the old touches working like a buffer and the second the new ones. By checking if a touch is inside both lists (lines 49 - 53), we can solve this issue.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class TouchInput : MonoBehaviour {
6
7    public LayerMask touchInputMask;
8
9    private List<GameObject> touchList = new List<GameObject> ();
10   private GameObject[] touchesOld;
11   private RaycastHit hit;
12
13   // Use this for initialization
14   void Start () {
15
16   }
17
18   // Update is called once per frame
19   void Update () {
20
21     if (Input.touchCount > 0) {
22
23       touchesOld = new GameObject[touchList.Count];
24       touchList.CopyTo (touchesOld);
```

```
25        touchList.Clear ();
26
27        foreach (Touch touch in Input.touches) {
28          Ray ray = Camera.main.ScreenPointToRay(touch.position);
29
30          if (Physics.Raycast (ray, out hit, touchInputMask)) {
31
32            GameObject recipient = hit.transform.gameObject;
33            touchList.Add (recipient);
34
35            if (touch.phase == TouchPhase.Began) {
36            recipient.SendMessage("OnTouchDown",hit.point,SendMessageOptions.DontRequireReceiver);
37            }
38            if (touch.phase == TouchPhase.Ended) {
39              recipient.SendMessage ("OnTouchUp", hit.point, SendMessageOptions.DontRequireReceiver);
40            }
41            if (touch.phase == TouchPhase.Stationary || touch.phase == TouchPhase.Moved) {
42              recipient.SendMessage("OnTouchStay",hit.point,SendMessageOptions.DontRequireReceiver);
43            }
44            if (touch.phase == TouchPhase.Canceled) {
45              recipient.SendMessage ("OnTouchExit",hit.point,SendMessageOptions.DontRequireReceiver);
46            }
47          }
48        }
49        foreach( GameObject gObj in touchesOld){
50          if (!touchList.Contains (gObj)){
51            gObj.SendMessage ("OnTouchExit", hit.point, SendMessageOptions.DontRequireReceiver);
52          }
53        }
54      }
55
56    }
57
58  }
```

Figure 4.31 TouchInput script

Its public variable of type “LayerMask” must be equal to the layer of the desired virtual object. For example, in this project, all the “Pick Up” objects belong to the layer “Touch Input”.

- PickUpButton.cs:

  It is attached to each of the “Pick Up” objects and it is utilized only in the Game Mode. As shown in the following figure 4.32, there are four functions for the touch handling and they are called from the “TouchInput” script. Firstly, there are four public variables

that should be initialized from the Unity Inspector window of the current object. The "ScoreController" variable refers to an empty object which has the "ScoreController" script attached to it. This object should be dragged and dropped in this public variable. Finally, the "PickUpButton" script has the public variable "room" from which we define the room number that this pickup object is located in the museum. Now regarding the functions, "Start" gets the current object's material in order to change the object's color when it is pressed and released. This is done by the three implemented functions. The fourth function called "OnTouchUp" increases the counter and updates the counter text when an object is collected thus deactivated. "OnTouchUp" function calls also the "SetCollected" function from the "ScoreController" in order to update it. More details, in the next paragraph.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4
5  public class PickUpButton : MonoBehaviour {
6
7          public Color defaultColour;
8          public Color selectedColour;
9          private Material mat;
10         public ScoreController score;
11         public int room;
12
13         // Use this for initialization
14         void Start () {
15                 mat = GetComponent<Renderer>().material;
16         }
17
18         void OnTouchDown() {
19                 mat.color = selectedColour;
20         }
21         void OnTouchUp() {
22                 this.gameObject.SetActive (false);  //object collected
23                 score.count += 1;
24                 score.SetCountWinText ();
25                 score.SetCollected (room);
26         }
27         void OnTouchStay() {
28                 mat.color = selectedColour;
29         }
30         void OnTouchExit() {
31                 mat.color = defaultColour;
```

```
32              }
33  }
```

Figure 4.32 PickUpButton script

- ScoreController.cs:

The current script is used only in Game Mode and implements two main functions of the museum floor plan panel. The first one is the locking and unlocking of levels (rooms). Figure 4.33 shows the code of “Scorecontroller”. It is important to explain its public variables. In the inspector window of Unity Editor, some of them should be related to objects. Specifically, these are the score texts (line 13), the buttons (line 19), the lock images (lines 20) and the target images (lines 22, 23).

After the necessary initializations in the lines 30 until 75, we update the locked levels using the functions “PlayerPrefs.SetInt” and “PlayerPrefs.GetInt”. In this way, the info of the last unlocked room is written to a file in order to be able to be read again even if the user closes the application. There is also the “SetCollected” function that updates the lists and the texts of the selected objects. Here, there is a special case when a room has no objects to be collected. The ”UnlockUntilNonEmptyRoom” function solves this problem using recursion. If an empty room occurs, then the next rooms are being unlocked until the one which includes at least one “Pick Up” object or until the last room is unlocked.

The second function of this script includes the handling of images (i.e. Locks and Buttons) and the text updates for each room. The lock images are activated covering the button when a room (level) is locked.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using TMPro;
5   using UnityEngine.UI;
6
7   public class ScoreController : MonoBehaviour {
8
9    public TextMeshProUGUI countText;
10   public int count;
11   public TextMeshProUGUI winText;
12
```

```
13    public TextMeshProUGUI[] ScoresTextsList;
14    private int[] ObjectsInRoomsList;
15    private int[] CollectedInRoomsList;
16
17    private int targetsNum;
18
19    public GameObject[] ButtonsList; //For each room
20    public GameObject[] LockImgList; //For each room
21
22    public GameObject[] MyImageTargetsList;
23    public GameObject[] MuseumImageTargetsList;
24
25    private int lastUnlockedRoom;
26    private PickUpButton pickUpButton;
27
28    // Use this for initialization
29    void Start () {
30      count = 0;
31      SetCountWinText ();
32      winText.text = "";
33
34      ObjectsInRoomsList = new int[7];
35      CollectedInRoomsList = new int[7];
36      targetsNum = 0;
37
38      for (int i = 0; i < ObjectsInRoomsList.Length; i++) {
39        ObjectsInRoomsList [i] = 0;
40      }
41
42      // Check my targets for room numbers
43      for (int i = 0; i < MyImageTargetsList.Length; i++) {
44        pickUpButton = MyImageTargetsList [i].GetComponentInChildren<PickUpButton> ();
45        if (pickUpButton==null) {
46          Debug.Log ("**" + MyImageTargetsList[i].name + " has no Pick Up Button**");
47        }
48        else if (pickUpButton.room >= 1 && pickUpButton.room <= 7) {
49          ObjectsInRoomsList [pickUpButton.room - 1] += 1;
50        }
51        else {
52          Debug.Log ("**" + MuseumImageTargetsList[i].name + " has room number out of my range**");
53        }
54      }
55
56      // Check museum targets for room numbers
57      for (int i = 0; i < MuseumImageTargetsList.Length; i++) {
58        pickUpButton = MuseumImageTargetsList [i].GetComponentInChildren<PickUpButton> ();
59        if (pickUpButton==null) {
60          Debug.Log ("**" + MuseumImageTargetsList[i].name + " has no Pick Up Button**");
61        }
62        else if (pickUpButton.room >= 1 && pickUpButton.room <= 7) {
63          ObjectsInRoomsList [pickUpButton.room - 1] += 1;
```

```
64        }
65        else {
66          Debug.Log ("**" + MuseumImageTargetsList[i].name + " has room number out of my range**");
67        }
68      }
69
70      for (int i = 0; i < ObjectsInRoomsList.Length; i++) {
71        targetsNum += ObjectsInRoomsList [i];                //total targets
72        ScoresTextsList [i].text = "0/" + ObjectsInRoomsList [i].ToString ();   // Initialize Score Texts
73      }
74
75      // Update Locks
76      lastUnlockedRoom = PlayerPrefs.GetInt ("LastUnlockedRoom", 1);
77      // maybe not assigned.
78      if (lastUnlockedRoom == 1){
79        PlayerPrefs.SetInt ("LastUnlockedRoom", lastUnlockedRoom);
80      }
81      UpdateLocks (lastUnlockedRoom);
82    }
83
84    public void SetCountWinText(){
85      countText.text = "Count: " + count.ToString ();
86      if (count >= targetsNum){
87        winText.text = "You win!";
88      }
89    }
90
91    //It's called from pick up button script
92    public void SetCollected(int room){
93      CollectedInRoomsList [room - 1] += 1;
94      ScoresTextsList [room - 1].text = CollectedInRoomsList [room -1].ToString ()
95                                + "/" + ObjectsInRoomsList [room - 1].ToString ();
96      if (ObjectsInRoomsList[room -1] == CollectedInRoomsList[room - 1]) {
97        PlayerPrefs.SetInt ("LastUnlockedRoom", room + 1);
98        UnlockRoom (room + 1);
99        UnlockUntilNonEmptyRoom (room + 1);
100     }
101   }
102
103   private void UnlockRoom (int roomNum) {
104     ButtonsList [roomNum - 1].gameObject.SetActive (true);
105     LockImgList [roomNum - 1].gameObject.SetActive (false);
106
107     if (ObjectsInRoomsList[roomNum - 1] == 0) {  //if there's no objects in room disable button
108       ButtonsList [roomNum - 1].GetComponent<Button> ().interactable = false;
109     }
110   }
111
112   private void LockRoom (int roomNum) {
113     ButtonsList [roomNum - 1].gameObject.SetActive (false);
114     LockImgList [roomNum - 1].gameObject.SetActive (true);
```

```
115     }
116
117     private void UpdateLocks (int lastRoom) {
118       for (int i = 0; i < 7; i++) {
119         if (i+1 <= lastRoom) {
120           UnlockRoom (i+1);
121         }
122         else {
123           LockRoom (i+1);
124         }
125       }
126     }
127
128     // Clear all Player's data
129     public void ResetAllPlayerPrefs () {
130       PlayerPrefs.DeleteAll ();
131       for (int i = 0; i < 7; i++) {
132         LockRoom (i+1);
133       }
134       Start ();
135       Debug.Log ("All PlayerPrefs are deleted.");
136     }
137
138     public void UnlockUntilNonEmptyRoom (int nextRoom) {
139       if (ObjectsInRoomsList.Length == nextRoom - 1) {
140         //We surpass rooms number. Don't do anything.
141       }
142       else if (ObjectsInRoomsList[nextRoom - 1] == 0) {
143         UnlockRoom (nextRoom);
144         UnlockUntilNonEmptyRoom (nextRoom + 1);
145       }
146       else {
147         UnlockRoom (nextRoom);
148       }
149
150     }
151
152   }
```

Figure 4.33 ScoreController script

- MainMenu.cs:

  In order to transition to another scene, we used the following function in figure 4.34.

```
1   public void GameMode () {
2       SceneManager.LoadScene ("Scenes/GameScene");
3   }
```

Figure 4.34 Call of LoadScene

This function needs the 'UnityEngine.SceneManagement" collection and it's imported by the command "using ..." as shown in previous scripts. Other scenes saved in the folder "Assets/Scenes" can be loaded with the call of similar implemented functions.

- UI scripts:

All of the screen displays and windows that pop up in our application are implemented with the "UI Buttons" (for the trigger) and its "Button" component (for the handling). New windows can be displayed by activating them. That means that their objects have been already created and we make them visible calling the function in figure 4.35.

```
1  public void EnableQuitVerification () {
2      VerifPanel.SetActive (true);
3  }
```

Figure 4.35 Window pop-up

- CameraFocusController.cs:

The camera detects better the image targets when they have more details. Thus, having the camera focused on the right object is crucial. Figure 4.36 shows the script that sets the camera focus on auto mode. In lines from 9 to 20, we make sure that "Vuforia AR Camera" has been started and then the autofocus is enabled in the function "SetAutofocus". This script is attached to the "AR Camera" object.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using Vuforia;
5
6  public class CameraFocusController : MonoBehaviour {
7    private bool mVuforiaStarted = false;
8
9    void Start () {
10     VuforiaARController vuforia = VuforiaARController.Instance;
11
12     if (vuforia != null){
13       vuforia.RegisterVuforiaStartedCallback (StartAfterVuforia);
14     }
15   }
```

```
16
17    private void StartAfterVuforia () {
18      mVuforiaStarted = true;
19      SetAutofocus ();
20    }
21
22    // this script will switch automatically to Auto focus continous mode
23    private void SetAutofocus () {
24      if (CameraDevice.Instance.SetFocusMode
25      (CameraDevice.FocusMode.FOCUS_MODE_CONTINUOUSAUTO)) {
26        Debug.Log ("Autofocus set");
27      }
28      else {
29        Debug.Log ("This device doesn't support auto focus");
30      }
31    }
32  }
```

Figure 4.36 CameraFocusController script

- ImageTargetReaction.cs:

The current script is attached to each image target object in the “Visit Mode”. When an image target is detected in the camera, the multimedia buttons and helping texts are immediately displayed. In more details, figure 4.37 shows that in the "Start" function the “TrackableBehavior” component was utilized to register a trackable event handler. Thus, in the function “OnTrackableStatusChanged” we activate the helping texts, the multimedia buttons and also update the map status. This means that the detected exhibits are shown as grey spots without blinking. Therefore, we need to drag and drop the corresponding circle image from the map panel to this script component in every image target object.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using Vuforia;
5   using TMPro;
6   using UnityEngine.UI;
7
8   public class ImageTargetReaction : MonoBehaviour, ITrackableEventHandler {
9
10    private TrackableBehaviour mTrackableBehaviour;
```

```
11
12    public GameObject MultimediaButtons;
13    public TextMeshProUGUI FoundText;
14    public TextMeshProUGUI HelpText;
15
16    public GameObject MapPosition;
17    private Blink blink;
18    private GameObject mat;
19    private Color disabledColor;
20
21    void Start() {
22      mTrackableBehaviour = GetComponent<TrackableBehaviour> ();
23      if (mTrackableBehaviour) {
24        mTrackableBehaviour.RegisterTrackableEventHandler(this);
25      }
26      disabledColor = Color.gray;
27    }
28
29    public void OnTrackableStateChanged(
30      TrackableBehaviour.Status previousStatus,
31      TrackableBehaviour.Status newStatus)
32    {
33      if (newStatus == TrackableBehaviour.Status.DETECTED ||
34        newStatus == TrackableBehaviour.Status.TRACKED ||
35        newStatus == TrackableBehaviour.Status.EXTENDED_TRACKED)
36      {
37        // When target is found
38        HelpText.gameObject.SetActive (false);
39        MultimediaButtons.SetActive (true);
40        FoundText.text = this.gameObject.name + " found";
41        FoundText.gameObject.SetActive (true);
42
43        MapPosition.GetComponent<Blink> ().enabled = false ;
44        MapPosition.GetComponent<UnityEngine.UI.Image> ().color = Color.gray;
45        MapPosition.gameObject.transform.localScale = new Vector3 (0.25F, 0.25F, 0.55F);
46      }
47      else {
48        // When target is lost
49        HelpText.gameObject.SetActive (true);
50        MultimediaButtons.SetActive (false);
51        FoundText.gameObject.SetActive (false);
52      }
53    }
54
55  }
```

Figure 4.37 ImageTargetReaction script

- AudioManager.cs:

  The “Audio Manager” script takes 3 kinds of parameters which are the AudioSource, the audio-clips and all the necessary text objects to display the current track information. In this project, the sound files are either “.mp3” or “.wav” located in the folder “Assets/Sounds”. They should be assigned to the table “audioClips” by drag and drop in the inspector window. Firstly, we assign the component “Audio Source” to a variable (line 27) and initialize its clip with our first sound (line 29). After that, three functions are called in order to display the title, the play-time, and the number of the current audio track.

  Several functions were written in order to run at the pressing of the corresponding audio player’s button. Some of them, for example, are Play, Pause, Mute etc. which are very simple and they don’t need elaboration. However, the “WaitForEnd” function is a little different and complex because it is a coroutine. This function updates the track time display and auto-plays the next song after the current’s completion. Its execution starts at line 42 when the play button is pressed. A coroutine can return a value as in line 49 which causes it to stop, although it resumes in the next frame from the exact same line. Generally, coroutines are used for continuous commands execution and they are preferred instead of the "Update" function in order to gain efficiency.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   using UnityEngine.UI;
6   using UnityEngine.Audio;
7   using TMPro;
8
9   [RequireComponent(typeof(AudioSource))]
10  public class AudioManager : MonoBehaviour {
11
12    public AudioClip[] audioClips;
13    private int currentTrack;
14    private AudioSource source;
15
16    public TextMeshProUGUI clipTitleText;
17    public TextMeshProUGUI clipTimeText;
18    public TextMeshProUGUI clipTrackNoText;
19
20    private int fullLength;
21    private int playTime;
22    private int seconds;
23    private int minutes;
24
25    // Use this for initialization
26    void Start () {
27      source = GetComponent<AudioSource> ();
28      currentTrack = 0;
29      source.clip = audioClips [0];
30      ShowCurrentTitle ();
31      playTime = (int)source.time;
32      ShowPlayTime ();
33      ShowTrackNumber ();
34    }
35
36    public void PlayAudio () {
37      if (source.isPlaying){
38        return;
39      }
40
41      source.Play ();
42      StartCoroutine ("WaitForTrackEnd");
43    }
44
45    IEnumerator WaitForTrackEnd () {
46      while(source.isPlaying){
47        playTime = (int)source.time;
48        ShowPlayTime ();
49        yield return null;
50      }
51      NextTitle ();
```

```
52    }
53
54    public void NextTitle (){
55      source.Stop ();
56      currentTrack++;
57      if (currentTrack > audioClips.Length -1){
58        currentTrack = 0;
59      }
60      source.clip = audioClips [currentTrack];
61      source.Play ();
62
63      //Show title
64      ShowCurrentTitle ();
65
66      //Show track number
67      ShowTrackNumber ();
68
69      StartCoroutine ("WaitForTrackEnd");
70    }
71
72    public void PreviousTitle () {
73      source.Stop ();
74      currentTrack--;
75      if (currentTrack < 0){
76        currentTrack = audioClips.Length - 1;
77      }
78      source.clip = audioClips [currentTrack];
79      source.Play ();
80
81      ShowCurrentTitle ();
82      ShowTrackNumber ();
83      StartCoroutine ("WaitForTrackEnd");
84    }
85
86    public void PauseAudio () {
87      StopCoroutine ("WaitForTrackEnd");
88      source.Pause ();
89    }
90
91    public void Mute () {
92      source.mute = !source.mute;
93    }
94
95    void ShowCurrentTitle(){
96      clipTitleText.text = source.clip.name;
97      fullLength = (int)source.clip.length;
98    }
99
100   void ShowPlayTime () {
101     seconds = playTime % 60;
102     minutes = (playTime / 60) % 60;
```

```
103       clipTimeText.text = (minutes + ":" + (seconds.ToString ("00")) + "/"
104         + ((fullLength / 60) % 60) + ":" + ((fullLength % 60).ToString ("00")));
105     }
106
107     void ShowTrackNumber () {
108       clipTrackNoText.text = (currentTrack + 1).ToString () + "/"
109         + audioClips.Length.ToString ();
110     }
111   }
```

Figure 4.38 AudioManager script

- Video Player functions:

We have not written additional scripts for the video player. The sole button which is the Play/Pause button is implemented with the existed Button Script Component. There, under the “On Click ()” frame we can add new features. By clicking the plus button and dragging there an object we can execute one of its functions when the button is pressed. In order to implement the “Play Button”, we dragged the object with the “Video Player” component. In the second option, we chose “VideoPlayer.Play” (or “VideoPlayer.Pause” for the “Pause Button”). Furthermore, the thumbnail image, the “Play Button” should be deactivated by selecting “Game Object -> Set Active” and uncheck the box while the “Pause Button” should be activated. However, in the case of “Pause Button”, we should deactivate the “Pause Button” itself and activate the “Play Button”. The figures 4.39 and 4.40 make it clear.

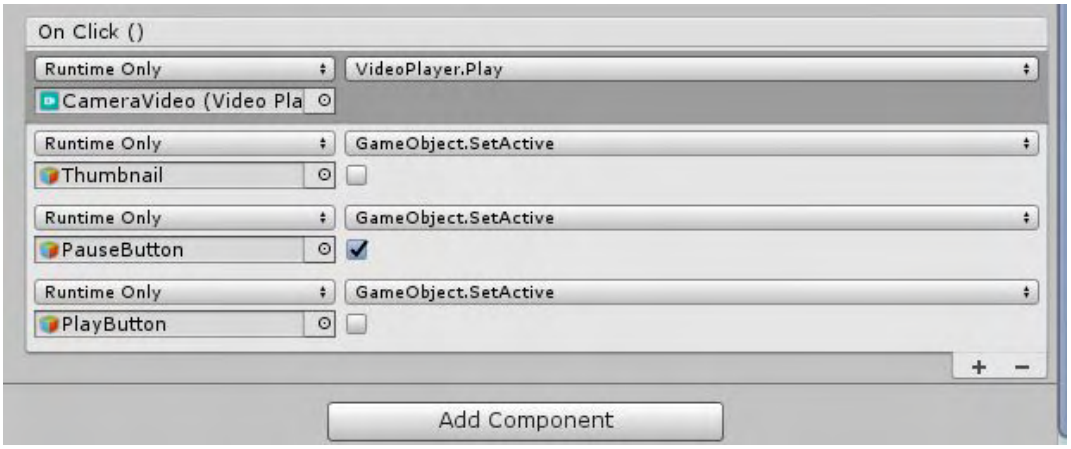

Figure 4.39 “Video Play Button”

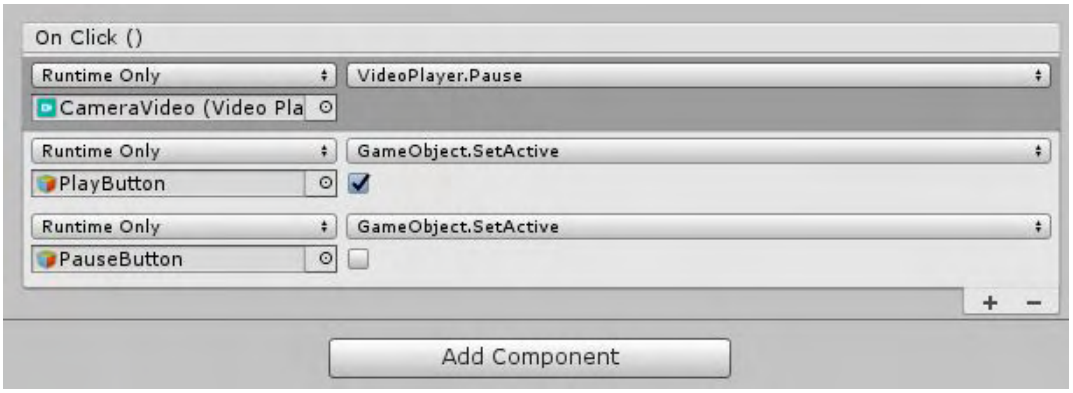

Figure 4.40 “Video Pause Button”

4.2.5 Extra Files

Image files (.png, .jpg) were used either for the exhibit’s photos or for design purposes like button icon and background image. These images are stored in the folder “Assets/Images”. In addition, sound files (.mp3) were also utilized only in the “Multimedia Panels”, for example, in the audio and video players. They are located in the folder “Assets/Sounds”. Video files (.mp4) were used only in the “Videos Panels”. They are stored in “Assets/Videos”.

CHAPTER 5

CASE STUDY: THE ARCHAEOLOGICAL MUSEUM OF VOLOS

The app has been tested as “proof of concept” in the Athanasakeion Archaeological Museum of Volos, which is a museum located in Volos, Greece. In this chapter, we explain briefly the integration process which consists of designing the museum’s floor plan, capturing images and test image recognition. We also present challenges and limitations that occurred alongside.

5.1 The Integration Process

First of all, the navigation of the user in the museum is facilitated by the museum’s floor plan. It is available inside the app at every moment. The utilized drawing was created with a simple image editor after studying the rooms of the museum and its existed floor plans. This is a customized feature of our app so it needs to be re-created in order for the app to be integrated into a different museum.

However, the main two challenges were about capturing images which we would use as image targets. The first one was the exhibit’s environment. We used only 2D tracking, so the figure of the exhibit was also a challenging factor. We wanted to examine the factors that help or incommode image recognition and therefore we examined each of the two factors in different stages. In the first stage we captured images from different museum rooms in order to test which conditions for instance lighting, background, exhibit cases like glass were prohibitive for the image recognition. Due to our sole use of only 2D tracking, it was more convenient for this stage to capture flat exhibits and exhibit-related posters to make sure that the shape of the exhibit was pretty much compatible. In the second stage, we collected images from a variety of exhibits with different shapes in the same room. This allowed us to have a general idea which exhibits and exhibit shapes were more convenient. We concluded that reflections or blur from glass cases, monotone exhibit colors, lack of distinctive patterns, and simple symmetric shapes like cubes, circles etc. were making the image recognition more difficult.

After the first two stages of capturing images, there was another one which aimed to test image recognition. In this stage, 11 image targets were selected by the archaeologist of

the museum with consideration to the chronological order. Therefore, these exhibits were from two rooms and they belong to six different kinds of exhibits: big objects, small objects in glass case, headstones, statuettes, statuettes in glass case and statues. Most of them were not enclosed, but some of them were in a glass case which may incommode image recognition. Exhibits which had been recognized successfully were 8 out of 11. That means the success rate for this trial was approximately 70%. The causes for the unsuccessful targets were probably the blur in a glass case and lack of distinguishable features like colors and patterns in some exhibits. The following figures show some of these image targets.

Successful targets

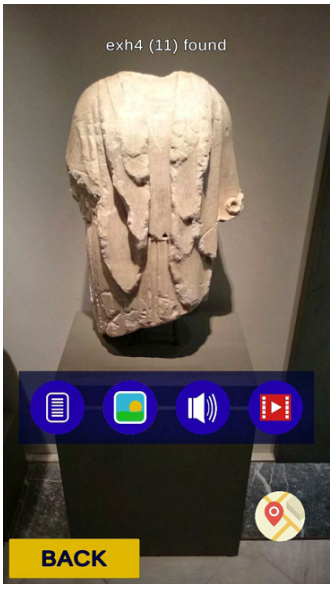

Figure 5.1 Statue

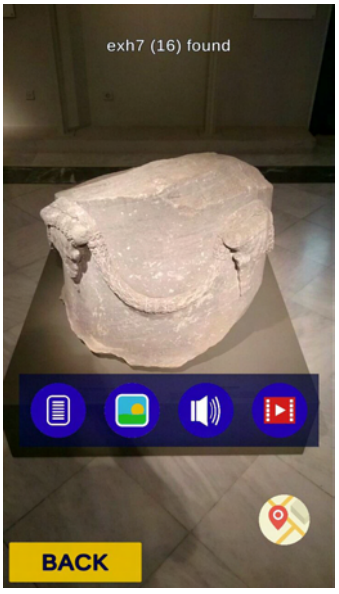

Figure 5.2 Large altar

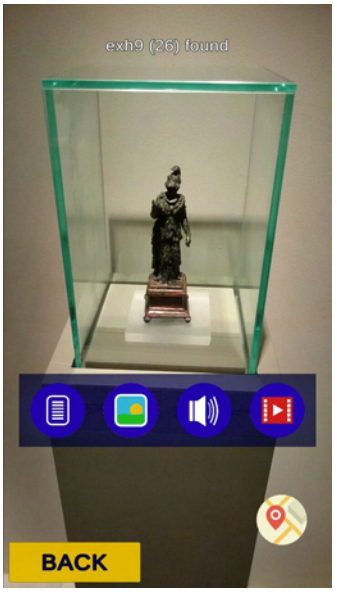

Figure 5.3 Statuette (in glass)

Unsuccessful targets

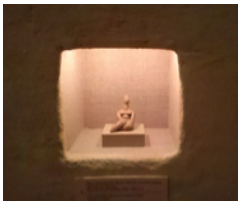

Figure 5.4 Statuette (in wall)

Figure 5.5 Large altar

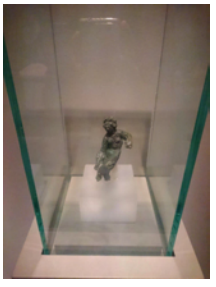

Figure 5.6 Statuette (in glass)

In addition, the space range of recognition is also very important. The distance required for the recognition of an exhibit was about 1-1.5 meters. The angle of targeting, however, was very limited. Because of the 2D tracking, the app was recognizing most of the exhibits only from in front. This can be counteracted by capturing multiple images of the exhibit from various angles and included them in the app.

CHAPTER 6

CONCLUSION

6.1 Discussion: Pros and Cons of the Implementation

The application's design has been based on our experience and research of available tools. The utilized technologies, for example, Image Recognition and Augmented Reality have offered several advantages to the final app. First of all, one of the most important benefits is that the only needed hardware was a smartphone. Thus, the visitor needs no extra hardware except their own smartphone device in order to use the museum app. Because each visitor is already familiar with his/her phone, the user experience becomes friendly and convenient. The lack of extra equipment also lowers the cost rendering this project of low budget. Another advantage is that this implementation does not require any kind of tags like QR or RFID tags. There is no intervention to the museum's decoration and aesthetics. Therefore, this facilitates the integration of the app to a new museum. The third benefit is about the internet connection. The app has no internet connection. Particularly, the image targets database is stored locally in the smartphone. In this way, there are no movement restrictions during the tour. The user can have a convenient navigation in all areas of the museum. The visitor will need an internet connection only once in order to download the app.

On the other hand, the implementation has its limitations. Firstly, the input of the app is static. This means that the multimedia content, the image targets and the floor plan image that we used, they cannot be changed or improved with more images after the app is built. Only an app developer can alter the content and rebuild the app through the Unity project. This may discourage museums from updating the app content. Moreover, the app depends strongly on the exhibit's current position and lighting. Variances in these factors can deteriorate the image recognition of the exhibit. Another consequence of the tracking technology and especially of 2D tracking is the limitation in the angle range of targeting. The user can target the exhibit only from in front with a small angle right or left; although multiple images from various angles can easily improve the angle range of the exhibit's recognition. Furthermore, the app has been implemented only for Android devices. Although Unity offers the option to build the app for iOS, we didn't have the

equipment to test it in this platform too. Last but not least, there is the limitation that Vuforia SDK which was used for Augmented Reality technology, requires a paid license in order for the app to be published and used commercially.

6.2 Future Work: Improvements and New Technology

Despite the limitations, the app can be improved in several ways. The first enhancement can be the utilization of 3D Tracking. Especially for tiny objects like jewelry and statutes, Vuforia offers 3D scanning which can increase the space range of recognition. Secondly, an internet connection can be used partially in order to enhance the efficiency of the app. For example, multimedia content like audio and videos can be streamed from a server, thus the size of the application will decrease considerably.

Moreover, new equipment like AR headset and AR glasses can enrich the user's experience significantly. The head movement can make the Augmented Reality more realistic. In this way, gesture recognition can be included as now the user's hands are free to move. This will increase the capabilities both of the user and the Augmented Reality system.

BIBLIOGRAPHY

[1] "Augmented Reality Wikipedia", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/Augmented_reality. [Accessed: 29- Jun- 2018].

[2] J. Normand and G. Moreau, "DoF-based Classification of Augmented Reality Applications."

[3] "The Ultimate Augmented Reality Technology Guide", *Reality Technologies*. [Online]. Available: http://www.realitytechnologies.com/augmented-reality. [Accessed: 29- Jun- 2018].

[4] P. Breuss-Schneeweis, ""The speaking celt"", *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing Adjunct - UbiComp '16*, 2016. DOI: 10.1145/2968219.2974044

[5] R. Wojciechowski, K. Walczak, M. White and W. Cellary, "Building Virtual and Augmented Reality museum exhibitions", *Proceedings of the ninth international conference on 3D Web technology - Web3D '04*, 2004. DOI: 10.1145/985040.985060

[6] A. Damala, P. Cubaud, A. Bationo, P. Houlier and I. Marchal, "Bridging the gap between the digital and the physical", *Proceedings of the 3rd international conference on Digital Interactive Media in Entertainment and Arts - DIMEA '08*, 2008. DOI: 10.1145/1413634.1413660

[7] C. Chen, B. Chang and P. Huang, "Multimedia augmented reality information system for museum guidance", *Personal and Ubiquitous Computing*, vol. 18, no. 2, pp. 315-322, 2013. DOI: 10.1007/s00779-013-0647-1

[8] A. Damala, T. Schuchert, I. Rodriguez, J. Moragues, K. Gilleade and N. Stojanovic, "Exploring the Affective Museum Visiting Experience: Adaptive Augmented Reality (A2R) and Cultural Heritage", *International Journal of Heritage in the Digital Era*, vol. 2, no. 1, pp. 117-142, 2013. DOI: 10.1260/2047-4970.2.1.117

[9] J. Keil, L. Pujol, M. Roussou, T. Engelke, M. Schmitt, U. Bockholt and S. Eleftheratou, "A digital look at physical museum exhibits: Designing personalized

stories with handheld Augmented Reality in museums", *2013 Digital Heritage International Congress (DigitalHeritage)*, 2013. DOI: 10.1109/DigitalHeritage.2013.6744836

[10] S. Rattanarungrot and M. White, "A service-oriented mobile augmented reality architecture for personalized museum environments", *2014 International Conference on Virtual Systems & Multimedia (VSMM)*, 2014. DOI: 10.1109/VSMM.2014.7136695

[11] A. Tillon, I. Marchal and P. Houlier, "Mobile augmented reality in the museum: Can a lace-like technology take you closer to works of art?", *2011 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities*, 2011. DOI: 10.1109/ISMAR-AMH.2011.6093655

[12] S. Yoon and J. Wang, "Making the Invisible Visible in Science Museums Through Augmented Reality Devices", *TechTrends*, vol. 58, no. 1, pp. 49-55, 2013. DOI: 10.1007/s11528-013-0720-7

[13] "Augmented Reality SDK Comparison | Comparison tables - SocialCompare", *Socialcompare.com*. [Online]. Available: http://socialcompare.com/en/comparison/augmented-reality-sdks. [Accessed: 01- Jul- 2018].

[14] "List of augmented reality software", *En.wikipedia.org*. [Online]. Available: https://en.wikipedia.org/wiki/List_of_augmented_reality_software. [Accessed: 01- Jul- 2018].

[15] "ARCore Overview | ARCore | Google Developers", *Google Developers*. [Online]. Available: https://developers.google.com/ar/discover/. [Accessed: 01- Jul- 2018].

[16] "Supported Devices | ARCore | Google Developers", *Google Developers*. [Online]. Available: https://developers.google.com/ar/discover/supported-devices. [Accessed: 01- Jul- 2018].

[17] "Catchoom - Image Recognition and Artificial Intelligence Solutions", *Catchoom*. [Online]. Available: https://catchoom.com/. [Accessed: 01- Jul- 2018].

[18] "EasyAR-Best engine for developing Augmented Reality", *Easyar.com*. [Online]. Available: https://www.easyar.com. [Accessed: 01- Jul- 2018].

[19] "Home | Kudan", *Kudan*. [Online]. Available: https://www.kudan.eu/. [Accessed: 01- Jul- 2018].

[20] "Home | MAXST Developer - The Right Choice for Your AR SDK", *Developer.maxst.com*. [Online]. Available: https://developer.maxst.com. [Accessed: 01- Jul- 2018].

[21] "VOID AR - Beyond Reality", *Voidar.net*. [Online]. Available: https://www.voidar.net/. [Accessed: 01- Jul- 2018].

[22] "Vuforia | Augmented Reality", *Vuforia.com*. [Online]. Available: https://www.vuforia.com/. [Accessed: 01- Jul- 2018].

[23] "License Manager | Vuforia Developer Portal", *Developer.vuforia.com*. [Online]. Available: https://developer.vuforia.com/license-manager. [Accessed: 01- Jul- 2018].