UNIVERSITAT POLITÈCNICA DE VALÈNCIA

TELECOM ESCUELA TÉCNICA VLC SUPERIOR DE INGENIERÍA DE TELECOMUNICACIÓN

DEVELOPMENT OF AN AUGMENTED REALITY MOBILE APPLICATION FOR MUSEUMS

Zhixian He

Tutor: Maria Carmen Bachiller Martin

Cotutor: Beatriz Rey Solaz

Trabajo Fin de Grado presentado en la Escuela Técnica Superior de Ingenieros de Telecomunicación de la Universitat Politècnica de València, para la obtención del Título de Graduado en Ingeniería de Tecnologías y Servicios de Telecomunicación

Curso 2019-20

Valencia, 30 de Abril de 2020

Abstract

The objective of the project is to develop an augmented reality Android application which could enhance the visit experience of an open-air sculpture museum in Universitat Politècnica de València. The application could not only provide the guidance for visitors on campus, but also could bring augmented reality experience to them when visiting specific sculptures. The application leverages various kinds of multimedia technologies and modern mobile development technologies to provide best experience from all aspects.

Contents

List of Figures

List of Tables

List of Code

Part I

Report

Chapter 1

Introduction

Augmented reality(AR) is an interactive experience of a real-world environment. Computer-generated content in the real world virtual objects, and sometimes across multiple sensory modalities. Augmented reality could be defined as a system which has the following three basic features:

- A combination of real and virtual worlds
- real-time interaction
- accurate 3D registration of virtual and real objects

Usually, the overlaid information could be constructive or destructive. The experience is seamlessly integrated with the physical world so that this could be perceived as an immersive aspect of the real environment.[1] That is to say, comparing with the well-known virtual reality, which usually provides an immersive experience with a simulated space to users augmented reality alters one's ongoing perception of a real-world environment.

With the development of mobile devices, including mobile phones and smart eyewear, augmented reality technology becomes more and more reachable during these days. Augmented reality could be easily achieved on the majority of smartphones.

1.1 Objectives

The overall objective of our project is to develop an augmented reality mobile application of an open-air sculpture museum. The museum would like to complement its visiting experience with augmented reality information.

The breakdown objectives of this project are the following:

- To research the development of Augmented Reality and its implementations, especially for the development of mobile devices.
- To develop the proof of concept application. The application could demonstrate the functionalities going to be used in the final version of the app.

- To do the digitalization of the sculptures. The digitalization includes modeling, scanning, and even photogrammetry. With the help of digitalization, we could detect sculptures with AR cameras. Besides, through modeling and photogrammetry, we could show the sculptures virtually through our application.
- To design a customized mark. Some of the sculptures have an irregular shape and could not easily be detected by the AR camera. Thus, we need to use marks for the AR camera to recognize, and based on that, and we could provide AR experience. So we need to design the mark. Marks need to be easily identified, and it should integrate with a unified look yet encoded different data.
- To carry out the test to real visitors on campus. They should use the app as general guidance, and they could also experience the AR events through it.
- To analyze the results and feedbacks. We would give out surveys to the visitors who used our application and based on their feedback, and we would identify which the application should improve.

1.2 Work Plan

This project began in mid-October 2019, and we established the initial objectives we wanted to carry out. Objectives have undergone modifications over time to better adapt to the problems and needs that have been arisen in the development process. The completion of the project would be in April 2020. Thus it lasts about five months for its resolution, including all phases. According to the school's requirements, we divide the project into four tasks, and each task has four objectives.

Since this is a project which needs to be carried out in Valencia, Spain and we are not available in the city before January, the work plan before January was majorly getting familiar with the technologies. Even though we keep weekly meetings so we could constantly evaluate the progress of the project. The meetings enable us to react to newly arising problems and obstacles quickly.

The four phases established for the project are as follows:

- **Revision of technologies and the state of the art**: This phase majorly targeting learning tools and programming languages that are going to be adopted in the future development works and collecting information about related topics.
- **Proof of concept app using Vuforia and Unity**: After learning the tools, we could start to develop some proof of concept apps to make sure this technology stack could be used to provide the desired user experience. Another reason we would like to do this is that we were not in Valencia at the moment, and we would still like to pursue the progress. Thus, developing a PoC app is the best way.
- **Fully functional application and validation with real museum visitors**: After we arrive in Valencia, we could start to work with real sculptures. At the moment, we could do some scanning, measurements, even modeling the sculptures if needed. Based on our proof of concept application, we could easily integrate the models and data of the sculptures into it, which could be used as the final shipping version of the application. After the development

process, we would like to provide this application to real visitors on campus to get some feedback.

- **To summarize the results into a meaningful set of publishable papers**: This phase targeting the compose of the final project paper. The paper would include everything we've done so far.

Table 1.1: Gantt chart of the work plan

	Nov 1-15	Nov 16-30	Dec 1-15	Dec 16-31	Jan 1-15	Jan 16-31	Feb 1-15	Feb 16-29	Mar 1-15	Mar 16-31	Apr 1-15	Apr 16-30
Task 1[Revision of technologies and state of the art]												
Get familiar with Unity	X	X										
Get familiar with Vuforia	X	X										
Get familiar with Blender	X	X										
Get familiar with sculptures	X	X	X									
Task 2[Proof of concept app using Vuforia and Unity]												
Implement the overall framework of the app	X	X										
Implement the media showcase		X	X									
Implement the interactive map		X	X									
Implement the basic AR functionality			X	X	X							
Task 3[Fully functional application and validation with real visitors]												
Scanning the sculptures						X	X					
Modelling the sculptures						X	X					
Application development							X	X	X			
Testing and publish								X	X			
Task 4[To summarize the results into meaningful set of publishable paper]												
Collecting user feedback									X			
Minor improvement and bug fix									X	X		
Composing the thesis									X	X	X	
Checking the thesis											X	X

Chapter 2

Background

Although this project is designing a specific application for a specific museum, there are still many related topics. For example, we would like to know how are augmented reality provided and what is the state of the arts of augmented reality used in museums.

2.1 Augmented Reality technologies

With the increasing computing power on mobile devices, there are more and more SDKs which could provide augmented reality experience. There are majorly three SDKs which has been adopted in commercial use.

2.1.1 ARKit

ARKit is the newly introduced Augmented Reality API developed by Apple, used on iPhone/iPad devices. This API is only available to users of devices with Apple A9 and later processors. The ARKit API provides functionalities like face tracking, image tracking, world tracking, object detection, orientation tracking, positional tracking, and etc.[2]

Comparing with other AR SDK, ARKit could leverage the TrueDepth camera[3] on some of the iOS devices. The TrueDepth camera provides much more accurate face recognition and tracking functionality when compared with traditional cameras. The built-in Animoji[4] is implemented using this feature.

2.1.2 ARCore

ARCore is Google's platform for building augmented reality experiences. 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.[5]

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

- **Motion tracking**: allows the phone to understand and track its position relative to the world.
- **Environmental understanding**: allows the phone to detect the size and location of all types of surfaces: horizontal, vertical, and angled surfaces like the ground, a coffee table, or walls.
- **Light estimation**: allows the phone to estimate the environment's current lighting conditions.

2.1.3 Vuforia

Vuforia is a cross-platform augmented reality SDK that enables the creation of augmented reality applications. The SDK uses computer vision technology to recognize and track 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, along with 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.[6]

2.2 Augmented reality in the Museums

Currently, augmented reality has been widely used in museums. Due to the difference in the exhibitions, different museums have a different approach to leverage this kind of technology.

2.2.1 Art Museums

Many art museums have been adopting augmented reality technologies. In 2018, the New York's Museum of Modern Art has already used the AR technologies to enhance its visiting experience. A group of artists has co-opted the brightly-lit Jackson Pollock gallery on the museum's fifth floor, turning it into their personal augmented reality playground. To normal visitors, the gallery remains unchanged, but to those that have downloaded the MoMA Gallery application, the paintings are merely markers—points of reference telling the app where to display the guerilla artists' works. Viewed through the app, Pollock's paintings are either remixed beyond recognition or entirely replaced.[7]

Augmented reality gives modern arts more possibility. Artists could use this technology to create more immersive experience ever than before. On the other hand, when it comes to traditional art museums, augmented reality could also uplift the interest when people are visiting. The Gaudi's famous building Casa Batllo is an excellent example of this. Visitors could have a more interactive visiting experience and could have a look of some of the original form of the architecture.[8]

2.2.2 Historical Museums

Back in 2015, the Natural History Museum of London has already tested the possibility of augmented reality technology. Instead of building a theater to play some scientific documentaries,

they represent the film through a handheld device. The film, which looks at evolution, uses the handheld display with a webcam to give each audience their window on the past. Animated 3D models of extinct creatures are combined with live video from the webcam in the handheld. These help to create the appearance that the animals are wandering around the studio in front of the audience. The camera tracking technology allows the system to render a view of the creature that matches the handheld's camera view by accurately calculating the camera's position and orientation within the studio. This allows the audience to use their handheld display to track the creature as it moves around them.[9]

Chapter 3

Design and Implementation

In this chapter, we would explain in detail how we leverage modern mobile development technologies to create the application and the augmented reality experience.

3.1 Museum

The museum is called Campus Escultòric Museum of UPV. It is an open-air sculpture museum. Different from traditional indoor museums, this museum is more like the campus itself. All the sculptures are scattered on campus, and naturally becomes an indispensable part of the campus life.

In this collection of more than 75 works, art, science, and technology come together. National and international artists feature in an examination of artistic expression that encompasses abstraction, kinetics, organics and geometry.[10]

Figure 3.1: MUCAES-UPV

3.2 Motivation

As we've mentioned in the previous chapter, currently, the augmented reality technologies have been widely adopted in museum tourism. Thus, the museum staff would also like to have this kind of augmented reality experience in their museum as well. However, there isn't any unified augmented reality solution for museums. The augmented reality experience needs to be explicitly designed according to the need of the museum. That's the reason why we are doing this work.

Aside from the augmented reality part, the museum also wants an application that could provide general guidance and introduction. For example, they would like to have an interactive map on the application. Visitors could view the location of each of the sculptures and could have a brief look at the sculptures, including the name, image, author, the year they made, etc.

The application we are creating will be installed directly onto the tablets of UPV. These tablets would be handed out to people when they are visiting the campus. Thus, we need to develop an Android application that could run on these kinds of low-end mobile devices.

3.3 Techonologies

We firstly introduce all the technologies we've leveraged in the whole development process in this section.

3.3.1 Vuforia

Different from ARCore and ARKit, Vuforia does not need hardware support. This SDK is built upon software, which means it could be used on any mobile device if they run iOS or Android. One more advantage of Vuforia is that it could still leverage the advantage of ARCore and ARKit. On devices that have native AR functionality support(ARCore/ARKit), Vuforia will automatically use these API as the backend, which provides better AR performance comparing to the pure software implementation.

3.3.2 Unity

Unity is a cross-platform game engine developed by Unity Technologies. The engine offers a primary scripting API in C#, and provide an easy to use drag and drop functionality. For 3D games, Unity allows specification of texture compression, mipmaps, and resolution settings for each platform that the game engine supports, and provides support for bump mapping, reflection mapping, parallax mapping, screen-space ambient occlusion (SSAO), dynamic shadows using shadow maps, render-to-texture and full-screen post-processing effects.[11]

Unity is a cross-platform engine, and currently, it could build games for more than 25 different platforms, including mobile, desktop, game consoles, and virtual reality.

The major reason we choose Unity is that it has great integration with Vuforia SDK. Comparing with Vuforia SDK used in native mobile development(iOS, Android and UWP), Unity could make Augmented Reality experience from a more visualized approach and could be easily deployed on multiply platforms, which extremely reduced the complexity of the development.

3.3.3 Blender

Blender is a free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, motion graphics, interactive 3D applications, and computer games. Blender's features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, rendering, motion graphics, video editing, and compositing.[12]

We majorly use Blender as a tool to create 3D models. Besides modeling, we also use Blender to do the retopology of the photogrammetry scanned model. The original model has too many vertices and faces that are unable to be rendered on mobile devices.

3.3.4 Android Studio + Kotlin + Android Jetpack

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on Windows, macOS, and Linux based operating systems. It is a replacement for the Eclipse Android Development Tools (ADT) as the primary IDE for native Android application development.[13]

Kotlin is a cross-platform, statically typed, general-purpose a programming language with type inference. Kotlin is designed to interoperate fully with Java, and the JVM version of its standard library depends on the Java Class Library, but type inference allows its syntax to be more concise. Kotlin mainly targets the JVM, but also compiles to JavaScript or native code (via LLVM). Language development costs are borne by JetBrains, while the Kotlin Foundation protects the Kotlin trademark.[14]

On 7 May 2019, Google announced that the Kotlin programming language is now its preferred language for Android app developers. Developing using Kotlin makes us write less boilerplate code.

Android Jetpack is a suite of libraries, tools, and guidance to help developers write high-quality apps more easily. These components help you follow best practices, free you from writing boilerplate code, and simplify complex tasks, so you can focus on the code you care about.[15]

We used several libraries from Android Jetpack: Room, Data Binding, LiveData, ViewModel, Navigation, AndroidKTX, and many other components. This set of libraries not only solved the compatible problem yet provide a great code structure, which makes the code easy to maintain and modify.

3.3.5 Adobe Illustrator

Adobe Illustrator is a vector graphics editor developed and marketed by Adobe Inc. In our project, we use this tool to create VuMark. VuMark is a bar code design and used by Vuforia. It allows the freedom for a customized and brand-conscious design while simultaneously encoding data and acting as an AR target. VuMark designs are completely customizable so that you can have a unique VuMark for every individual object.[16]

3.3.6 Agisoft Metashape

Agisoft Metashape is a stand-alone software product that performs photogrammetric processing of digital images and generates 3D spatial data to be used in GIS applications, cultural heritage documentation, and visual effects production as well as for indirect measurements of objects of various scales.[17]

3.4 Application

The curator of the museum wants us to create an application not only include augmented reality functionalities, but also could be used for guidance. Hence, we need to show some location of each of the sculptures on the map, and demonstrate useful information like the author, year they made, material, and even some description is introducing the background story of the sculpture.

At the very beginning, we were planning to implement the whole application using Unity, as using only one tool reduced the complexity of the overall development process. Yet, it is much more complex to create the general guidance experience in Unity rather than using some native Android development techniques. Thus, we decide to switch the implementation of this part to native Android development, for the following reasons:

- Unity's UI library majorly targets video game GUI, which has different rendering logic comparing with the traditional GUI libraries.
- It is quite difficult to implement the general map feature in Unity rather than native Android since native Android has better Google Map SDK integration.

Here, Figure 3.2 shows the structure of our application, and we will explain each part in detail in the following parts.

Figure 3.2: Structure of the application

3.4.1 Icon Design

As a mobile application for the museum, we need to show the logo on both the launcher icon and the splash screen. We were provided with a bitmap file of the logo of the museum, yet this couldn't satisfy our needs. We are not sure about which size of the logo we are going to use, so it would be much better if we could have a vectorized file of the logo. Since the shape of the logo is really simple, we were able to recreate this one using Adobe Illustrator(Ai). After we create the file in Ai, we could export it as an SVG file. Then we could import the SVG file into Android Studio as a vector asset. By doing this, we could scale the same logo to any size we want. For example, for the app icon, we would prefer a smaller size; To display the icon on the splash screen, we would prefer a bigger size.

To make the launcher icon of the application suite different device models, we adopted an adaptive icon. For example, an adaptive launcher icon can display a circular shape on one OEM device, and display a squircle on another device. Each device OEM provides a mask, which the system then uses to render all adaptive icons with the same shape.

To create the adaptive icon, we use the vector asset we created as the foreground and we use the primary color of the museum logo as the underlay background. Thus, in Figure 3.3(a) you could see that our icon being adaptive to four kinds of different masks, and in Figure 3.3(b) you could see the splash screen of the application.

(a) Adaptive icon (b) Splash screen

Figure 3.3: How we use icons

3.4.2 Main Menu

To make the development process well organized, we decide to adopt the brand new architecture components called "Navigation". In traditional Android development, developers would implement an "Activity" class for each of the screens they would like to demonstrate, and they handled the navigation relationship by creating Intents between each of the Activities. When developers have created plenty of Activities and Fragments, it may become challenging to manage the navigation relationship between these screens efficiently. However, navigation components are the

answer to this chaotic situation.

Navigation component asks developers to create one single Fragment and set it as the host, and then developers could create a navigation graph simply using one XML file. The XML file describes the navigation relationship between each of the Fragments. At this moment, we are not jumping between Activities anymore. Instead, we use only one main Activity and one fragment as the container, and each time we navigate between different screens, the navigation component would help us substitute the Fragment with the one we described in the XML file. In Android Studio, we could have a visual view of the navigation relationship between each of the fragments and easily modify them. Thus, the overall navigation of the application become well organized.

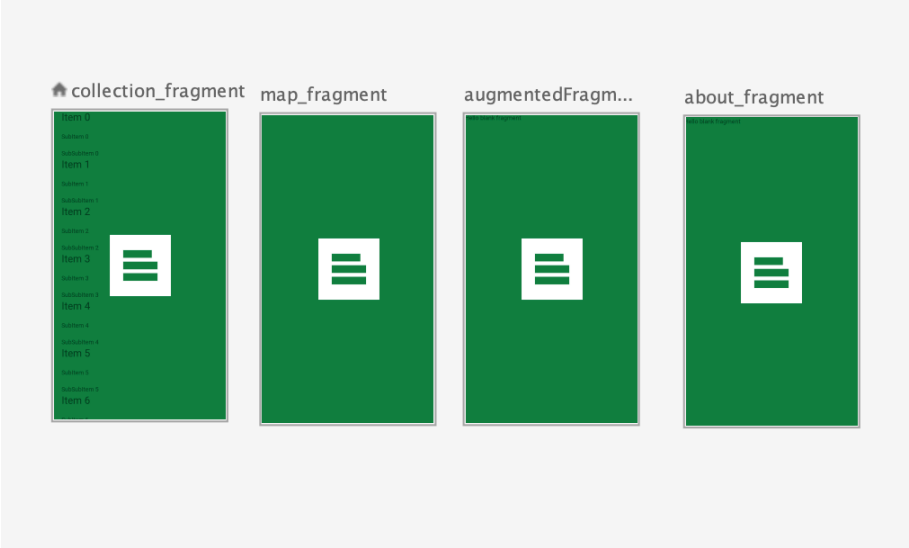


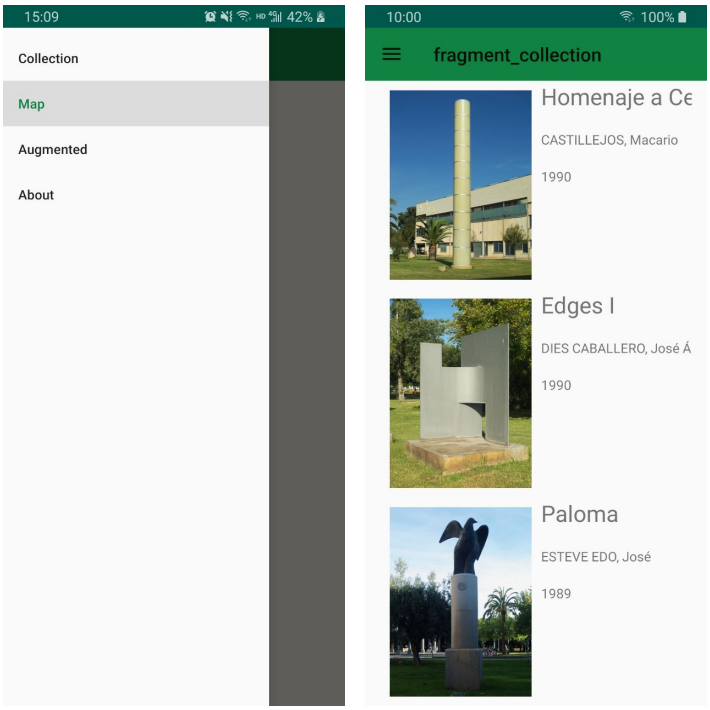
Figure 3.4: Navigation view

According to the museum's requirements, they would like to have these following functionalities aside of the AR camera:

- A map showing the location of the sculptures
- A list showing all the sculptures, and users could view the details of each sculpture.
- An about page showing basic information about the museum and the application

Following the Material Design guideline, we decide to implement the overall navigation using the "Navigation drawer". In Figure 3.5(a), you could have a view of how the main menu looks like.

You could see, that we have four screens in all. Each of them provides different functionalities. The collection view is shown in Figure 3.5(b). The list includes all the sculptures, and users could view brief introduction of each sculptures. They could also click on each entry to view more detailed description. The map and the augmented reality view would be introduced in the following sections, and the about page shows some information about the museum and the developers.



(a) Navigation drawer (b) Collection

Figure 3.5: Application screenshots

3.4.3 Database

In traditional Android development, developers would use SQLite as the default choice of the database. Although Android provides the SQLite API natively, the API is quite low level and difficult to use for modern mobile application development. For example, developers need to write plenty of boilerplate code to implement the conversion between SQL query and object. Besides, there isn't any verification for the original SQL query data. Once the table of the database changed, developers need to update all the manually related queries in the code.

The Room persistence library is another component in Android Jetpack, and it is an abstraction layer over SQLite to allow for a more robust database access while harnessing the full power of SQLite.

There are majorly three components in Room:

- Database: Contains the database holder and serves as the main access point for the underlying connection to data.
- Entity: Represents a table within the database.
- DAO(Data Access Object): Contains the method used for accessing the database.

Figure 3.6 demonstrates the relationship among the three components of the Room database.

Thus, with the help of Room, we could easily manipulate data in the database, with a more object-oriented approach. And when the tables in the database get modified, we could easily update our

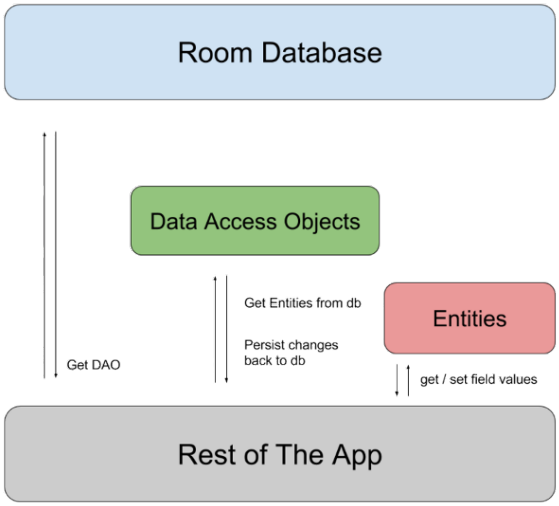


Figure 3.6: Room Architecture

code by creating a migration.

We have a PDF file from the museum, and it listed the following information of each sculpture: name, author, the year they made, material, dimension, and location. Firstly, the code for creating the Entity of the sculptures is shown in Listing 3.1.

Listing 3.1: Sculpture.kt

```
@Entity(tableName = "sculpture")
data class Sculpture (
    @PrimaryKey var id: Int = 0,
    var title: String,
    val year: Int,
    val author: String,
    val material: String,
    val lat: Double,
    val lng: Double
    ) {}
```

Just as the table in SQLite, we need to have a primary key. Here we use the internal ID of the sculpture as their primary key, as this value is different from each other. Then we have some additional columns, which are the data we would like to present in the application.

Listing 3.2 shows the DAO we designed.

Listing 3.2: SculptureDAO.kt

```
@Dao
interface SculptureDao {
    @Query("SELECT * FROM sculpture")
    fun getAll(): LiveData<List<Sculpture>>
    @Insert
    suspend fun insert(sculpture: Sculpture)
    @Insert
    suspend fun insertAll(sculptures: List<Sculpture>)
}
```

This defines the SQL queries we would like to perform. We would like to read the whole table

and draw the location of sculptures on the map, with corresponding information, we would like to insert entities into the database when the app runs the first time on the mobile device.

Finally, Listing 3.3 shows the database.

Listing 3.3: SculptureDatabase.kt

```
@Database(entities = arrayOf(Sculpture::class), version = 2)
abstract class SculptureDatabase: RoomDatabase() {
    abstract fun sculptureDao(): SculptureDao

    private class SculptureDatabaseCallback(
        private val scope: CoroutineScope
    ) : RoomDatabase.Callback() {

        override fun onCreate(db: SupportSQLiteDatabase) { ... }
    }

    companion object {
        @Volatile
        private var INSTANCE: SculptureDatabase? = null

        fun getDatabase(context: Context, scope: CoroutineScope)
                : SculptureDatabase {
            val tempInstance = INSTANCE
            if (tempInstance ≠ null) return tempInstance
            return INSTANCE ?: synchronized(this) {
                val instance = Room.databaseBuilder(
                    context.applicationContext,
                    SculptureDatabase::class.java,
                    "sculpture_database"
                )
                    .addCallback(SculptureDatabaseCallback(scope))
                    .build()
                INSTANCE = instance
                instance
            }
        }
    }
}
```

The database class is built upon the Sculpture entity class and the SculptureDAO class. To make it easier to access the database, we create a singleton of the database. So each time when we would like to perform some action on the database in the code, we could call the SculptureDatabase.INSTANCE.

3.4.4 Map

The map is implemented using Google Maps SDK for Android.[18] The SDK provides plenty of useful features such as adding markers, adding styled maps, hiding map features with styling, displaying info windows, drawing shapes on the map, drawing overlay tiles, etc. Developers could add any tweaks onto the real-world geographical data.

In our case, we would like to mark all locations of sculptures onto the map, so that visitors would gain a more intuitive view of the distribution of the sculptures on campus. Listing 3.4 shows how we inflate the MapView with the GPS data of the sculptures.

Listing 3.4: MapFragment.kt

```
override fun onMapReady(googleMap: GoogleMap) {
    mMap = googleMap
    sculptureViewModel.allSculptures.observe(this, Observer { sculptures →
        for (s in sculptures) {
            // var str = gson.toJson()
            mMap.addMarker(
                MarkerOptions()
                    .snippet("Author: " + s.author)
                    .position(LatLng(s.lat, s.lng))
                    .title(s.title))
        }
    })
    val upv = LatLng(39.4808, -0.342823)
    mMap.moveCamera(CameraUpdateFactory.newLatLng(upv))
    mMap.moveCamera(CameraUpdateFactory.zoomTo(18f))
}
```

This part of code works after the MapView has been loaded. It first tries to access the sculpture ViewModel we created previously, and generate a marker object containing the corresponding data. Then the marker would be inserted into the map. After inserting the data, we create another GPS location for our "camera" to target at. At last, we set our camera to look at the campus, and adjust the zoom level so that users could have a better experience by default.

3.4.5 Augmented Reality Camera

The Augmented Reality is implemented using Unity and Vuforia SDK, and this part is the core of our application. As requested by the curator of the open-air sculpture museum, they've selected three sculptures which they would like to apply augmented reality experience. To demonstrate the possibility of augmented reality technology, we were planning to create three augmented reality events for three sculptures from different aspects.

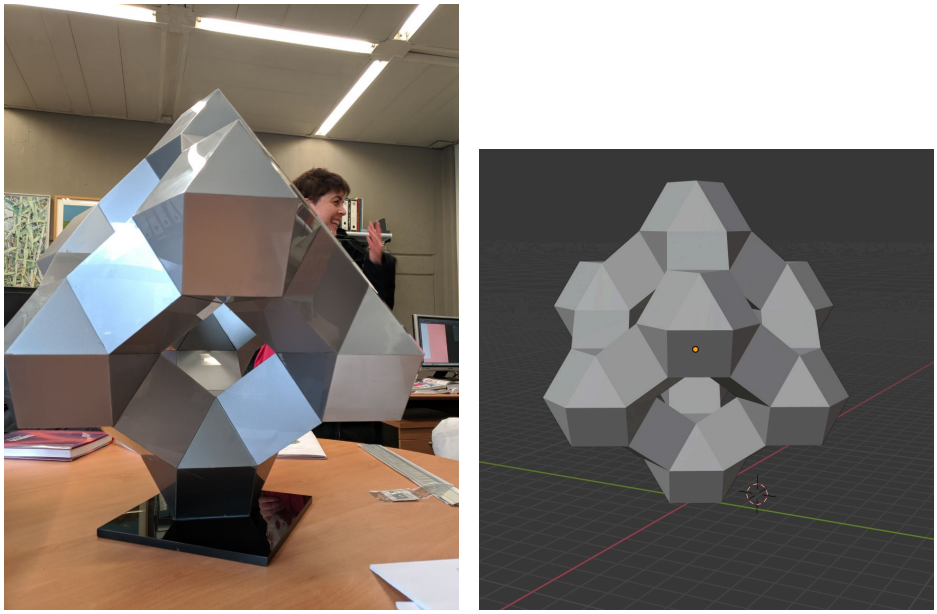
3.4.5.1 Image Recognition + Model Event

This sculpture is called "Yturralde". It's currently not placed on campus yet. As you could see from Figure 3.7(a), it is quite a small one. The curator of the museum showed us this sculpture in their office and explained what they would like to have. As this is a small sculpture, the museum wants to create a virtually larger version of it. Thus, visitors on campus could have a view of the enormous sculpture using the augmented reality application.

Based on the requirements, we started to design the implementation method. The first problem we encountered is how to represent the sculpture virtually. Although Vuforia is an augmented reality SDK, it doesn't have such functionality built-in. The only way we could show a virtual object in augmented reality application is to have a 3D object file of it, which means we need to create a 3D model of the sculpture.

Luckily, the sculpture has a regular shape. From our measurement, all of its surfaces are equilateral triangles or squares, and the sculpture is symmetric. These indicate that it could be implemented from scratch in 3D modeling software.

From our observation, we found that the sculpture consists of two parts: one is the cube, and another one is a polyhedron. The polyhedron is consists of squares and equilateral triangle, with



(a) Yturralde (b) 3D model of Yturralde

Figure 3.7: Sculpture and 3D model

each edge of a square extends as an equilateral triangle, and four of the equilateral triangles encircle a square.

Thus, we would like to split the polyhedron into two symmetric parts, the upper part and the lower part, which are precisely the same. What we need to do then is to determine the angle between the triangles and the square. There is a simple calculation. As we could see from the diagonal line of the encircled square is perpendicular with the top and bottom square, and as they share the same width, we could easily calculate the distance between the top and bottom squares.

Then imagine an orthogonal view from the side of the polyhedron. We could draw a right triangle. One right-angle side is half the length of the diagonal line, and the hypotenuse is the length of the square. Based on the Pythagorean theorem, we could easily calculate the length of another right-angle side. Thus, we have all the building blocks to create the polyhedron.

By combining the upper and lower parts, we get the polyhedron, and by combining multiple cubes and polyhedrons, we get the 3D model of the sculpture.

Although the model of the sculpture is created, we need to "place" it on campus. As we are not going to use the original sculpture, we couldn't leverage any of the tracking functionalities provided by Vuforia SDK. Luckily, there is still one solution we could use, which is the VuMark.

After importing the VuMark into Unity, we could then add the model we just made as a child object of the VuMark. Then we could adjust the related size, position, and angle of the virtual sculpture. To make it looks more like the original sculpture, we apply a metallic texture onto the model.

Initially, Vuforia would lose track once the VuMark leaves the camera view. Yet by enabling the "Extended Tracking", the application would remember the location of the mark, and even if the mark moves out of the camera sight, we could still have a view of the 3D model freely.

3.4.5.2 Image Recognition + Image Event

(a) Pleyades (b) Particles in Unity

Figure 3.8: Sculpture and visual effects

This sculpture we were working on is the one called “Pléyades”. Emilio Martínez made the sculpture in 2000. It is more like a relief on the ground. The sculpture was intended to be camouflaged to better integrated with the main road of the campus. The sculpture symbolizes the The Pleiades, which is also known as the Seven Sisters, an open star cluster.[19] The steel squares on the group resemble the massive blue stars in the cluster.

Although the sculpture was intended to be integrated with the the surrounding environment, the curator of the museum, wants visitors to have a sense of reward when they notice the sculpture on the ground, and that is the the reason why we are making this augmented reality experience.

At the very beginning, we were provided with two photos of Pleiades from NASA’s website. The museum wants us to show an overlay image above the sculpture when the user is looking at it using the application. Thus, even without any explicit explanation, visitors would gain knowledge that this sculpture represents cosmic stuff, and more experienced visitors may recognize that it is the Pleiades.

We started to design the implementation method of this kind of experience. The initial idea was based on Vuforia’s image tracking ability. Vuforia’s image tracking could attach virtual objects onto a predefined image, and by taking a picture of the sculpture, we could add an overlay image above it in Unity. Thus finally, when using the application to look at the sculpture, we could get a view of the star cluster on the ground.

Based on the image tracking ability, we proposed two solutions:

- The first one is that we set a fixed observation point. We take a picture of the sculpture from that point, and by tracking the picture, we could implement the augmented reality

Figure 3.9: Pleiades

event. However, the difficulty is that visitors are required to stand in the same position as to where we take the picture, and they need to face the same angle. Otherwise, the perspective of the view in the visitor's camera is different from our predefined image. The difference could make the application unable to recognize the picture. Thus, the augmented reality event wouldn't be triggered. Besides, we need to adjust the perspective of the overlayed star cluster image; otherwise, the image may have a different perspective in the visitor's view.

- The second solution is that we use a drone to take several pictures of the whole sculpture from a perpendicular angle. By concatenating all the images together, we could gain a monolithic image of the sculptures. The whole sculpture is about 3 meters wide and 12 meters long, so it is challenging for us to take a monolithic picture without aid from other tools.

However, after our evaluation, none of these solutions could work. The first one, as we've mentioned, it is challenging to have the same position and angle. Even if we marked the location on the ground, a slight difference could lead to an unexpected performance of the augmented reality event. And about the second solution, we do not have a drone to use. Additionally, from the experience of our later progress, we found out that the outdoor environment is usually strongly affected by the sunlight, and Vuforia is not good at dealing with the different light conditions of the same object.

Another point that we were concerned about is that the sculpture may not strictly be aligned with the image we have. Although the sculpture is called "Pleyades" and it gained its inspiration from the star cluster, it is just a symbolization of the star cluster.

The experience we would like to provide to our visitors is that we want them to know this is a cosmic thing. Thus, we do not need to provide the exact image of the Pleiades. To leverage the advantage of Unity as a game engine, we proposed the following solution:

Instead of displaying an image, we would like to create some particles above each of the steel squares virtually, indicating that these are the stars. And we would like to add a transparent overlay

background of the cosmos. The visitors would know what this represents from these kinds of visual stuff.

Another difficulty we're facing is how to recognize the sculpture, or in other words, how do we trigger the event. The VuMark of the the museum was initially designed for another sculpture to use, yet it turns out that this could be used as an anchor point for the virtual world in this scenario.

As the whole sculpture is a rectangle, we selected one corner as the anchor point to position the VuMark. From that corner, we measure the distance from that point to each steel squares on the field. Although the real-world data doesn't matter, the ratio of distances to the size of the VuMark could be used to set the position of particles in the Unity engine.

By adjusting the VuMark's angle in the engine and adjusting the position of the particles according to our measured data, we could then show this cosmic look on the ground. To note here that, when I was doing the measurement work, I was about leaving Spain due to the breakout of Coronavirus so that I couldn't do a further field test. Yet, by implementing the experience using VuMark, we could test the outcome anywhere. One shortage is that we could not check if the particles are aligned with the steel squares on the ground.

3.4.5.3 Object Recognition + Model Event

(a) Homenaje a Manolo Gil (b) Photogrammetry of Homenaje a Manolo Gil

Figure 3.10: Sculpture and photogrammetry model

The sculpture is called "Homenaje a Manolo Gil". Oteiza Jorge made this sculpture in 1989. This sculpture was going to be renovated in a few months, by the fine art students on the campus. The curator of the museum wants us to show the original form of the sculpture after the renovation, using the augmented reality technology. Thus, what we need to do is to somehow virtually keep the current state of the sculpture, both the shape and the texture.

Also, we need to find a way for the application to recognize the sculpture. Luckily, Vuforia SDK

already provided a way to extract features from real-world objects, so that later on, we could use these features to recognize real-world objects.

Vuforia provides an Android application called "Vuforia Object Scanner". The application needs to work with a piece of paper provided by Vuforia. The left and the bottom side of the paper consists of irregular pattern, and this helps the application to locate this paper in real-world environment. On the top right corner, there is a coordinate system. That coordinate system is where we need to put our scanning object.

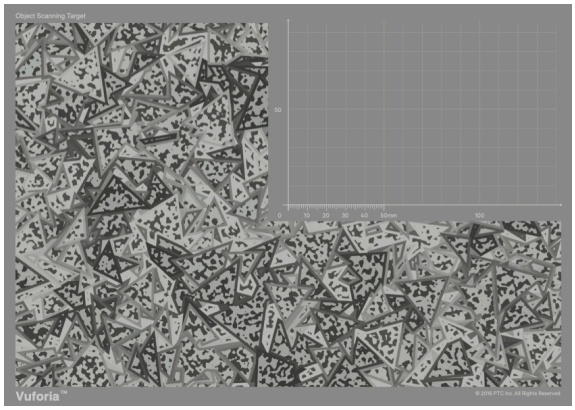


Figure 3.11: Vuforia Object Scanning Target

Figure 3.11 is the PNG image provided by Vuforia in the size of an A4 paper. Yet, the sculpture we were working on is about 50cm to 50cm. Thus, the A4 paper is not big enough for this scanning task. Luckily, someone has already converted the PNG image into a vector PDF file, which means we could print the image at any size we want, without losing resolution.

So the museum staff prepared a big scanning sheet for us. The sheet is about 100cm to 60cm, along with two grey boards, which would be used to block the background when scanning.

As we were concerned about the interference of the sunlight, we planned to do the scanning around the dusk. At that time, the sunlight would be blocked by the building, so there would not be direct sunlight.

After setting up the scanning environment, we started to do the scanning. We have two museum staff to assist us during the scanning process. They need to hold up the greyboard so that the scanning application would not extract details from the background. The scanning work didn't go very well. It took us nearly two hours to get a satisfying scanning result. During the scanning, we found out that the sculpture is hollowed out, so it was challenging for the application to extract details. And sometimes, we just accidentally extracted features from the background, which means we need to discard the scanning result and start over.

After scanning, we started to test the scanning result. Based on the data we obtained, the Vuforia Scanner can recognize the sculpture from most of the angle. Yet, it was enough for visitors to use, as they majorly would see the sculpture from the front side, which the recognition works pretty well.

However, we got an unexpected result when we were testing the scanning result in the afternoon the next day. When we were doing the test, the application failed to recognize the sculpture. The reflection of its metallic surface causes this, and we couldn't do anything to make the recognition

happen under direct sunlight. Thus, we need to discard this object tracking method, as this is not a reliable one. From our previous experience, the reliable solution would be using the VuMark for tracking. At that time, I was about leaving Spain, so we didn't have a chance to test the VuMark recognition result in this sculpture. But due to the portability of the VuMark, we could still develop the recognition remotely.

Aside from tracking, as we mentioned before, we still need to keep a virtual version of the presence of the sculpture. At the very beginning, we were planning to model the sculpture from scratch, yet it was a tough job. What we need to do was to model this sculpture as accurately as possible. The sculpture has an irregular shape, which is hard for us to model, even with reference.

So we were thinking about using photogrammetry to obtain the model. However, the biggest obstacle is that we didn't have the equipment to do the photogrammetry. Usually, photogrammetry requires a setup stage with many cameras and specialized software controlling all the data captured by the cameras. Besides, there should be tons of works even after the scanning, because usually, the raw model we gained contains many noise data, and we need to clean up the model.

Luckily, one museum staff found an approachable way to do the photogrammetry. With the help of Agisoft Metashape, we were able to implement the photogrammetry with only one camera. The software provides a scanning manual instructing how to do the scan.

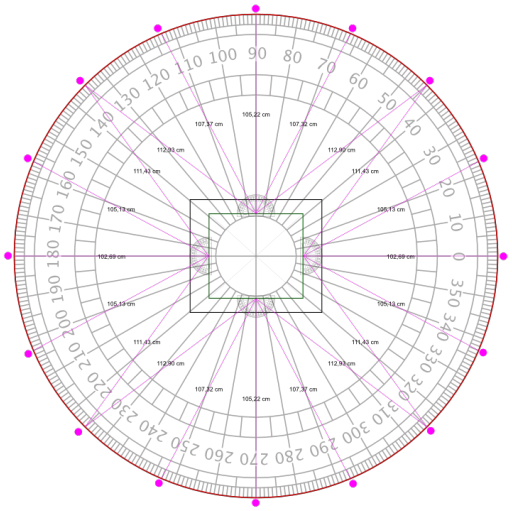


Figure 3.12: The sheet we use to coordinate the scanning point

This is the tool we used when we were doing the scanning. By measuring the distance and angle, we put 16 small cone barrels evenly spaced on the ground, to mark the horizontal position we would use to take the photo of the sculpture. Then, by setting a fiducial horizontal height, we adjust the tripod of the camera to that height. Each time we move the camera, we make sure that the height of the camera keeps the same.

In all, we took pictures from 3 different height and 16 different horizontal position, so, at last, we would get 48 pictures of the sculptures, and that's all for the "scanning" part.

Later on, we need to do the post-processing. Before importing the image into the Metashape, we need to first erase the background from the image, to make sure that we would not process the

background data.

After that, we could then import all the 48 photos into Metashape. The software would automatically align all the pictures to its corresponding position and angle. Then the software would generate something called “dense cloud”. It’s like a cloud of many color points. All these points consist of the shape and color of the sculpture. Yet, it is not a mesh.

Then, we need to convert the dense cloud into a mesh, which could be manipulated using other 3D modeling software. In our case, we would like to use Blender to do a retopology of the generated mesh, since the directly generated mesh contains too many faces. It was hard to process even on a high-end PC, and our target platform is low-end mobile devices. Thus we need to reduce the redundant faces to make sure the mesh could be rendered in real-time even on low-end devices.

After the retopology, we would then re-import the mesh into Metashape. This time, we would like to bake the texture onto the mesh. The software would automatically bake the texture onto the mesh since the overall shape keeps similar to the original mesh generated by the software.

And that’s how we obtained the 3D model of this sculpture.

3.4.6 Combining Unity and Android Application

The Unity Engine is built upon Mono, an open-source implementation of the .NET Framework. The C# code we wrote in Unity would be compiled to CIL bytecode, and these codes could run on Mono runtime. However, Android applications run on Android Runtime, a runtime similar to JVM. These two are entirely different bytecode and runtime, yet now we want to put them together.

Luckily, the Android SDK provides another development kit called NDK. These allow native applications, which could compile to machine code, directly run on devices. On the Unity side, they provide a tool called IL2CPP, which could compile the CIL bytecode to the C++ code.

So, the way to integrate Unity application and Android application is shown as the figure below.

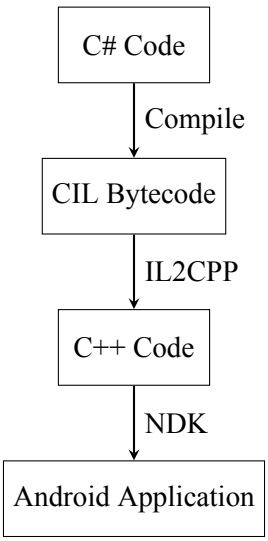


Figure 3.13: How Unity code integrate with Android

The generated C++ project contains a Java class which works as UnityPlayer, an Android Activity

class which functions as the entry point of the application. By creating an Intent to this Activity from our previously built Android native application, we could then start the augmented reality program from the main application.

Chapter 4

Result and Discussion

In the beginning, we were planning to make a test for real visitors on campus. The application is supposed to work on the tablets of school, and it would be provided to the visitors on campus. We've made a survey in advance, which would be handed out to the visitors who used our application, and this could help us to obtain useful feedback. However, due to the breakout of COVID-19, we were unable to carry out this plan. To demonstrate our work, we still append the survey form into the annexes.

The one big obstacle we met during the development process is that it turns out the Vuforia SDK is not good at dealing with objects with different light conditions. In the beginning, we were planning to implement the augmented reality event of "Pleyades" using image tracking. However, image tracking is worked like extracting features from the image. The features are usually some points which have significant greyscale difference, as we've introduced in the technology part. When the light condition changed, the features which were introduced in the original image may no longer exist in the new scene. Thus, the Vuforia SDK could not parse the image in different light conditions, which may lead to the malfunctioning of the image tracking.

The biggest challenge we faced is in the "Oteiza" sculpture. In the beginning, we were planning to implement the tracking using the Vuforia's object tracking functionality. However, as we mentioned before, the whole scanning process didn't work well as we expected. This mainly caused by three reasons:

- The first reason is that it is challenging to provide a distraction-free background for the scanning. The sculpture is in the open-air environment, and we are not allowed to move the sculpture indoor. Even if we've already prepared a gray board in advance, the scanning process required us to constantly move the board to hide the background. Yet, it was not very easy to achieve this. Thus, when we were doing the object scanning, we constantly encountered unexpected feature points from the environment. Once this happened, we had to do the scanning start over.

- The second reason is that the sculpture is not a good object for scanning. The sculpture is majorly black. Although it doesn't have a regular shape as the one we modeled, it still has a simple outline. The most severe problem is that the sculpture is hollowed out, which makes it even more difficult to extract features. When we were doing the scanning, we spent so much time trying to extract enough details, yet it just can't.

- However, the real reason which makes us give up on this method is that after we've successfully done the scanning, the next day, when we would like to test the tracking result in the noon, it couldn't work.

Later on, when we reviewed the object scanning tutorial, we found out that the best scenario for the object scanning method is with small indoor objects. The Vuforia SDK couldn't deal with direct lighting, especially when the surface of the object is metallic. For small indoor objects, the tutorial suggests setting up a rotatable stage and an environment without direct lighting. However, none of these requirements could be achieved in our situation, as we were dealing with an open-air sculpture with quite large size.

Based on these two cases, we finally realize that the Vuforia SDK is not suited to the outdoor environment. The SDK majorly targets small indoor objects. Other native SDKs may provide better tracking functionalities comparing to Vuforia. Yet, Vuforia's pure software implementation makes it able to run on almost any kind of device. In contrast, those native SDKs usually could only run on a handful of newly released devices.

But the VuMark is indeed something that we could rely on. The VuMark itself is highly customizable yet could embed data into it. At last, all three augmented reality events would be triggered using VuMarks, yet it provides a sense of consistency when visitors are using the application on campus. Each time when they saw the VuMark on the ground, they would know that there is an augmented reality event waiting for them to explore.

So at last, even though we didn't strictly follow our initial design, we still provide a quite convincing result.

Chapter 5

Conclusion and Future Work

The museum staffs are quite satisfied with our outcome. Even though we only implemented augmented reality event for a small part of the sculptures, we've already proven that it is feasible to provide this kind experience in an open-air the museum, even if we encountered some problems.

To conclude, our application successfully satisfied the needs of the museum. It could provide guidance, provide useful information about the corresponding sculptures, and could provide different types of augmented reality events.

From another point of view, I've been dealing with various software during the whole development process. From game engine to mobile application development toolchain, from 3D modeling software to graphic design software, from photogrammetry software to augmented reality SDK. It's such a precious opportunity for me to learn and use all these different kinds of technologies.

Even though some of our plans didn't go well, we gained knowledge of the short come of the Vuforia SDK. For example, Vuforia SDK could not handle direct sunlight, and could only work with indoor objects. Due to the current limitation of the augmented reality SDKs, we need to carefully investigate what the SDK could implement and what couldn't. Based on this kind of knowledge, we then could start to design the related augmented reality experience.

Currently, the three augmented reality events are triggered by scanning the VuMark. These provide a unified yet extendable framework for the identification of the augmented reality event on campus. If the museum would like to extend the application so that it could work with more sculptures, they could easily generate more VuMarks and add augmented reality experience onto them. This could keep users' cognition yet extending the scope of application.

References

[1] Wikipedia contributors. *Augmented reality — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Augmented_reality&oldid=952477093. [Online; accessed 22-April-2020]. 2020.

[2] Apple. *ARKit*. https://developer.apple.com/documentation/arkit. [Online; accessed 7-April-2020]. 2020.

[3] Apple. *Tracking and Visualizing Faces*. https://developer.apple.com/documentation/arkit/tracking_and_visualizing_faces. [Online; accessed 22-April-2020]. 2020.

[4] Apple. *How to use Animoji on your iPhone and iPad Pro*. https://support.apple.com/en-us/HT208190. [Online; accessed 22-April-2020]. 2020.

[5] Google. *ARCore*. https://developers.google.com/ar/discover. [Online; accessed 7-April-2020]. 2020.

[6] Wikipedia contributors. *Vuforia Augmented Reality SDK — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Vuforia_Augmented_Reality_SDK&oldid=915325078. [Online; accessed 7-April-2020]. 2019.

[7] Miranda Katz. *Augmented Reality Is Transforming Museums*. https://www.wired.com/story/augmented-reality-art-museums/. [Online; accessed 22-April-2020]. 2020.

[8] Casa Batllo. *Useful Information*. https://www.casabatllo.es/en/visit/. [Online; accessed 22-April-2020]. 2020.

[9] Graham Thomas. *Augmented Reality film launches at the Natural History Museum*. https://www.bbc.co.uk/rd/blog/2010-12-augmented-reality-film-launche. [Online; accessed 22-April-2020]. 2015.

[10] UPV. *Campus Escultòric Museum of the UPV (MUCAES-UPV)*. http://www.upv.es/entidades/FPA/info/1018519normali.html. [Online; accessed 22-April-2020]. 2020.

[11] Wikipedia contributors. *Unity (game engine) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Unity_(game_engine)&oldid=949330256. [Online; accessed 7-April-2020]. 2020.

[12] Wikipedia contributors. *Blender (software) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Blender_(software)&oldid=948897687. [Online; accessed 7-April-2020]. 2020.

[13] Wikipedia contributors. *Android Studio — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=949424141. [Online; accessed 7-April-2020]. 2020.

[14] Wikipedia contributors. *Kotlin (programming language) — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Kotlin_(programming_language)&oldid=949103207. [Online; accessed 7-April-2020]. 2020.

[15] Google. *Anrdoid Jetpack*. https://developer.android.com/jetpack. [Online; accessed 7-April-2020]. 2020.

[16] Vuforia. *VuMark*. https://library.vuforia.com/articles/Training/VuMark.html. [Online; accessed 7-April-2020]. 2020.

[17] Agisoft. *Agisoft Metashape*. https://www.agisoft.com/. [Online; accessed 22-April-2020]. 2020.

[18] Google. *Google Maps SDK for Android*. https://developers.google.com/maps/documentation/android-sdk/intro. [Online; accessed 7-April-2020]. 2020.

[19] Wikipedia contributors. *Pleiades — Wikipedia, The Free Encyclopedia*. https://en.wikipedia.org/w/index.php?title=Pleiades&oldid=951097072. [Online; accessed 17-April-2020]. 2020.

Part II

Annexes

Appendix A

User Manual

The application has four views, and each of them has the corresponding ability. We want to introduce them separately. Firstly, after you opened up the application, you could click on the top left icon or swipe right from the left side of the screen to open up the navigation drawer. From the navigation drawer, you could see the four views in a list. They are “collection”, “map”, “augmented” and “about”.

The default view is the collection view. In the collection view, you could view all the sculptures in a list. In the list view, you could have a look at the picture, the name, the author, and the year they made of each sculpture. Once you clicked on one entry, you would navigate to its detail view. In the detail view, you could see more information such as material, dimension, and even the description of the sculpture.

The map view consists of a map and many markers. Each marker marks the location of the sculptures. You could click on each marker to have a brief look at the sculpture. The map view is intended to guide visitors on campus. You could find the sculpture you like and headed to the location straight away.

The augmented view opens up the camera. If you see the VuMarks on campus, open up the camera and scan the mark. Then you could experience the corresponding augmented reality experience we’ve prepared for you. Currently, we’ve implemented three different augmented reality experiences for three sculptures. Try to find them on campus. There will be more in the future!

The about page shows information about the MUCAES-UPV museum and the developers of the application. If you are interested in any of us, feel free to reach out to us!

Appendix B

Survey

Aumented Reality App assessment questionaire

Your opinion is important to try to improve the application. The information collected here will be very useful to know your ratings and suggestions. The survey is completely anonymous and voluntary.

THANKS A LOT!

Smartphone/Tablet model	
Smartphone/Tablet brand	
OS (version)	

Indicate if the application has failed in any of the marks (during the visit).

1. Marks	Fail (Yes/No)	Comments
1. Pleyades		
2. Oteiza		
3. Yturralde		

Questions after the visits.

	YES	NO	Sometimes
The application works on my smarphone			
The video is visualized			
The video is visualized but the audio is not heard			
Audio is heard			
The application is slow and the contents are lagged			
I had trouble installing it			
The marks are detected well			

RATE THE EXPERIENCE (0 negative to 5 very positive):

	1	2	3	4	5	No opinion
Did you like the experience?						
Did you find the content of the museum more attractive with this application?						
Would you install the application if you went to a museum that had it to download?						
Have you learned more thanks to the application?						
Did you find its use easy?						
Did you find the installation easy?						

COMMENTS AND SUGGESTIONS:

Appendix C

VuMarks

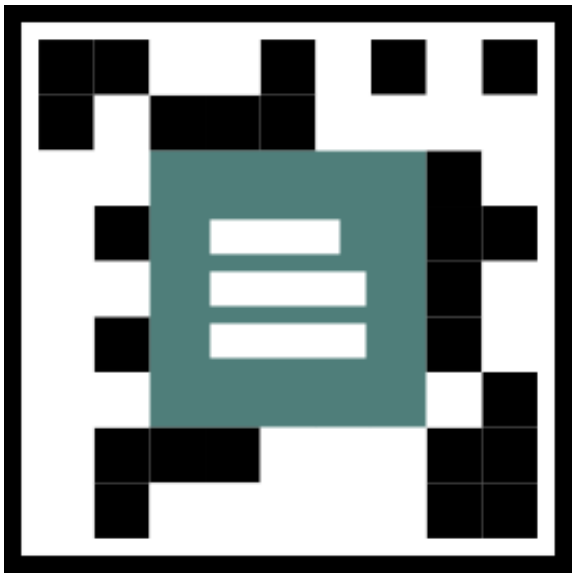


Figure C.1: VuMark for 303

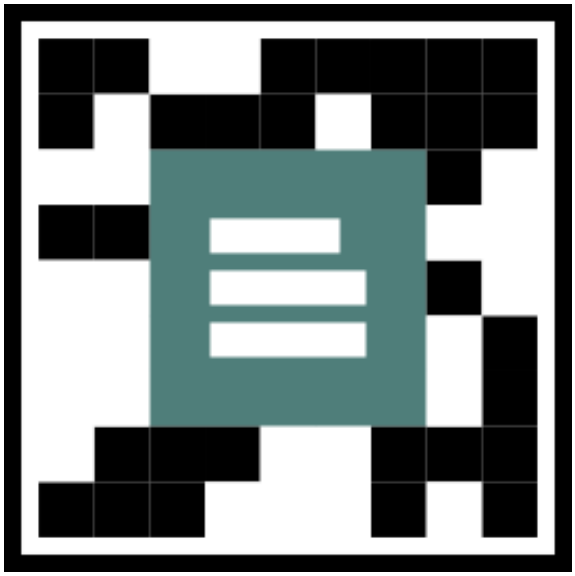


Figure C.2: VuMark for 731

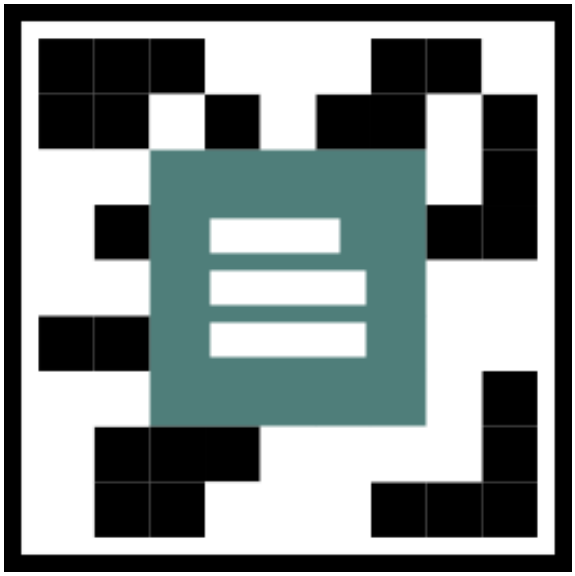


Figure C.3: VuMark for 1234

Appendix D

Image of the Usage

Because we do not have the chance to do the field test due to the breakout of COVID-19, the image of use, is just an image I took from my home when I was using the application. It demonstrates that the application could recognize the VuMark and add virtual objects in the camera view.

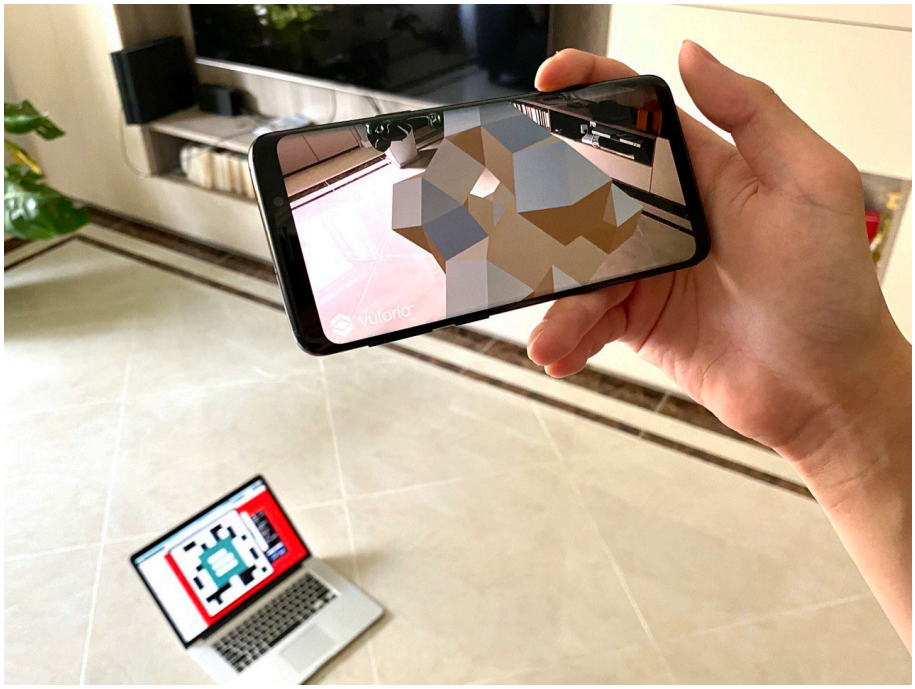


Figure D.1: Image of use