UKRAINIAN CATHOLIC UNIVERSITY

BACHELOR THESIS

Street Art AR

Author:
Maxym KOMARENSKYY

Supervisor:
Oles DOBOSEVYCH

A thesis submitted in fulfillment of the requirements for the degree of Bachelor of Science

in the

Department of Computer Sciences
Faculty of Applied Sciences

Lviv 2020

Declaration of Authorship

I, Maxym KOMARENSKYY, declare that this thesis titled, "Street Art AR" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed:

Date:

"Life wouldn't be so interesting if it wasn't so hard."

Me

UKRAINIAN CATHOLIC UNIVERSITY

Faculty of Applied Sciences

Bachelor of Science

Street Art AR

by Maxym KOMARENSKYY

Abstract

Augmented Reality is a new way of world perception. In the next 10 or 20 years, AR will make a revolution in every world market: Education, Advertising, Videogames, Military, Healthcare, Engineering, etc. But I do not want to wait all these years I want to start the future now.

Contents

List of Figures

List of Abbreviations

AR	**A**ugmented **R**eality
SDK	**S**oftware **D**evelopment **K**it
DB	**D**ata**B**ase
DAO	**D**ata **A**ccess **O**bject

Dedicated to parents

Chapter 1

Introduction

Street Art AR is a platform where designers, 3d modelers, and just creative people of all around the world can share their best works. It is no matter where you are and who you are now if you can create something cool, something that other people must-see, this platform is for you.

So how it works? All that you need is your phone. You should download an app, open it and the entire world of breathtaking 3d and 2d models is in your hands. But let me describe it more detailed. If you open our app for the first time we need to identify you, so you should go through the small registration process and you are in. Our platform can use three types of users.

First type:

You are the person who never has experience of the creating 3d or 2d models but you are interested in what people can create and you want to check this out. You should not be a painter to go to the Gallery and get satisfied with what you see there. The same with our platform, you can never create some 3d model by yourself but you want to see what other people can do. However, we always will be happy if our platform is going to inspire you to create your own art.

Second Type:

You are a person who has experience in creating 3d models. This can be a hobby or even a job. You want to say something to the world in your own specific way, share what you had done before, or even inspire other people to start to make as good art as you or even better.

Third Type (Mixed Type):

Basically the third type of users at the same time want to see what other people do and they uploading their own works.

As I mentioned above my app has two main flows. This is when the user wants to see the works of other people and when he/she wants to upload his/her work to our platform.

1. **Want to upload**

The users need to open the "Upload" section of the app, click on the button "I want to upload my work", choose the file (of course this step would be much easier if the user did it on his laptop or PC, but to reduce the number of people who

should work on this project and amount of time that we need to spend, I have decided to make everything in the app), put your model on the right place (using AR technology) and click "Done button".

2. **Want to see**

The user needs to open the "Camera" section and the system will automatically download all near models and show you on the screen (also using AR technology, the user to be able to see the model from each side).

Chapter 2

Market Analysis

This section I would like to start with projects that inspired me to do "Street Art AR". This year I have faced three brilliant platforms that designers use to share their finished works and find inspiration for the future ones. These are Pinterest, Behance, and Dribbble. In general, they all are very similar and do the same job but in a bit different approaches. So here I will tell only about the most famous platform of these three - Pinterest. First of all, Pinterest is a social network with more than 200 million users. There you can share all possible visual content such as images, videos, infographics, etc. Pinterest has a very good implemented search structure using which you can easily find what you are looking for. Another project which I want to mention here is the greatest and the most popular AR game of all time - PockemonGo. Better about the popularity of this game will tell this infographic.

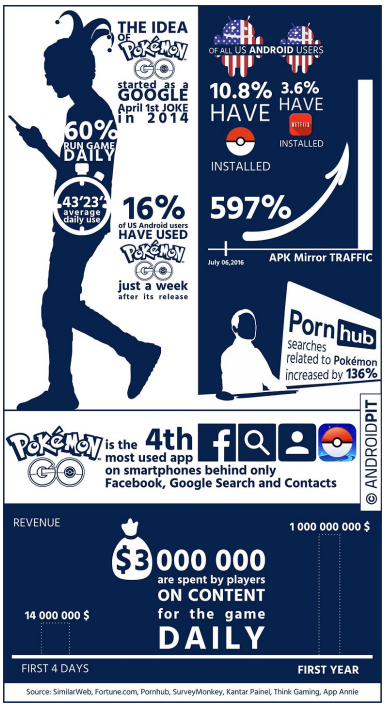

FIGURE 2.1: Infographic shows us a level of PokemonGo popularity

And also we can see how attention to AR technology has changed after this game

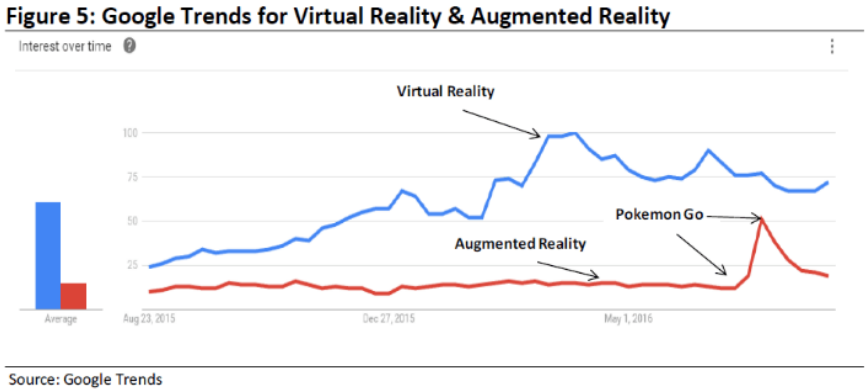

FIGURE 2.2: Popularity Increasing of ar technology

So we see that audience is interested in live interaction via AR technology.

And at the intersection of these products, my idea of "Street Art AR" started to form

2.1 IKEA Place

IKEA is a very useful app for people who want to buy furniture. Using AR technology IKEA Place app allows users to visualize how different furniture designs would look at their homes. Advantages of "Street Art AR": I understand that IKEA place has a totally different goal if we compare it to my app. But technology is quite similar. Both apps should put 3D models in chosen places and allow the users to look at it from all sides. But my app goes much farther.

1. The user can share them with the entire world
2. The user can upload his own objects.

In the end, I want to emphasize that IKEA Place is a great app and it does its work very well.

2.2 HOLO

Next app that attracted my attention was HOLO. On a first look, it is a very basic AR app which can do not really complex things. Choose an object, place it, film a video, or take a photo. But during this process you can event not to notice that this app has great made functionality and this is adjusting an object by resizing, rotating, and moving. In all other things my app has the same advantages as with IKEA Place

2.3 IngressPrime

IngressPrime - the first AR game on my list. IngressPrime uses the user's location data to create a real-time AR game around the user. The mission of the user is to find portals in the real world and capture them for your side. The perfect thing, in this app, it is a world generating. All "portals" or place of interaction is the same for all

people and placed in the same spots. The incredible amount of work has been done to synchronize all interactions between users and those portals. But for my app, I don't need such a complex structure. The only one that I need, is to find a way to generate the same world for all users. It means that each user should see the object in the same place that other users see it.

2.4 WallaMe

WallaMe is an AR app that allows users to share secret messages with other users. All that you need to do is to point a smartphone's camera at the wall and draw your secret message and only people with whom you had shared it before and who stand on the same location, can see it. The advantage of my app is that I work with more complex files (3D model) and our users can share your work not only with close friends or people you know but with the entire world.

2.5 Zome

After WallaMe I have found a similar but much complex app - Zome. This is my real competitor. Until that moment I was downloading and investigating only similar features that will be in my app but not the idea. This app has really helped me to ask myself the right questions. What I can afford people that Zome cannot? Why they should use my app instead of Zome? But before I answer these questions let me explain the idea of Zome. Using Zome users can drop messages in any place in the open world. When another user is close to your message he can reveal it.

Now we are going to go through the disadvantages of Zome, which help me to understand which part of the project I should focus on

1) **UX**

Zome has a really bad user experience. When you open the app for the first time you just don't understand what is happening. There wasn't even an on-boarding tutorial. If your app is complex and you don't have the possibility to simplify it, at least what you can do it is add a tutorial. But, of course, the most important is good and intuitive UX which you can not create for one day

2) **Design**

Another not good thing is Zome's design. Everyone has his own taste and it is always hard to judge design but Zome doesn't follow any rules of modern design. One grey main color and low-quality icons never were in the trend

3) **Content**

Here is the main difference between my idea and the idea of Zome. Using Zome you as a user can see through your phone only dots that show you that on that place is some message and only when you click on this dot app will let you open an image or text and not using AR but just on another screen. My app will automatically upload all near objects. It will not be just a text or an image but a 3D model that the user can look at from all sides.

2.6 Summary

My app should have the possibility of an easy object placing as IKEA Place has, should have object adjusting by resizing, rotating, and moving as HOLO has, should have well-synchronized world between all users like IngressPrime has, but at the same time it must have good UI/UX, people not to think how it works but just express themselves in a spectacular way of modeling

Chapter 3

Technology Investigation

3.1 ArCore

ArCore is made by Google. The most important is that you do not need to add any additional devices to use it. Three key technology of ArCore is Light Estimation, Environment understanding, and Motion Tracking.

- Light Estimation allows your phone to estimate the condition of the light.
- Environment understanding allows your phone to detect the sizes of objects that are captured by the camera.
- Motion Tracking allows your phone to understand the position in the world.

A lot of other AR SDK requires you to move your camera around the place where you are to detect all surfaces. And ArCore is no different. When the user moves his phone around the room, ArCore memorizes all surfaces to be able to put other virtual objects in the future. ArCore builds his own virtual world to allow you to place some object on the surface, move the camera to another place, come back to the previous place and you will see that the object is still there. It uses motion tracking for this purpose

3.2 Vuforia

Another very popular AR SDK is Vuforia. Tree main features of this SDK are:

- object recognition
- image recognition
- additional features

Let me explain these features in more detailed way.

3.2.1 Object recognition

One of the most useful things that Vuforia can do, is 3D object identifying (Model Target) in real-time.

But to use this technology we need to follow some rules

- The object should not be monotonous.
- The object should not be transparent or shiny.

- The object should be static.
- The object should be complex. If the object will be more complex, it will be easier for Vuforia to identify it.

3.2.2 Image Recognition

Other useful thing of Vuforia is image recognition in real-time (Image Target). Like with object recognition, image recognition has some rules.

- The size of the image should not be more than 2MB
- The image should not be monotonous. The same rule as object recognition has. If the image has more detail, it will be much easier for Vuforia to identify it. After uploading an image, Vuforia will estimate this image from 0 until 5. Where 0 is a hard detected image and 5 is a good image for detection.
- The image should have .jpeg or .png extension.

3.2.3 Additional features

- Extended tracking If image or object is no longer in the field of view of the camera, the Vuforia continue to show information about it.
- Ground plane Detecting of horizontal planes and placing content. But we can use this technology on a limited amount of devices because it is using data from ArCore and ArKit.

3.2.4 Disadvantages

As I understood Vuforia is very hard to learn how to use it. For the beginner, it is almost impossible.For example, you have to write about 1000 lines of code to make a marker tracking.

Another big disadvantage is that Vuforia has good cooperation with Unity but not with native modules. For Android, they have released a new version of the library which made connection much easier but iOS has the only library which is written on C++, there is a wrapper on Objective-C but if your project is written on Swift you must code a bridge between Swift and Objective-C and it is always not stable approach.

3.2.5 Advantages

1. It is very simple to develop with Unity.
2. Useful functionalities which are described above.
3. A free version, but with some limitation.
4. Supporting of many devices.

3.3 ArKit

That was two SDK for Android which I find really good implemented and useful to know, about their existing, for any Android developer who wants to work with AR. But there is still one more player who cannot be used by Android developer but it has a big part of Ar market - ArKit

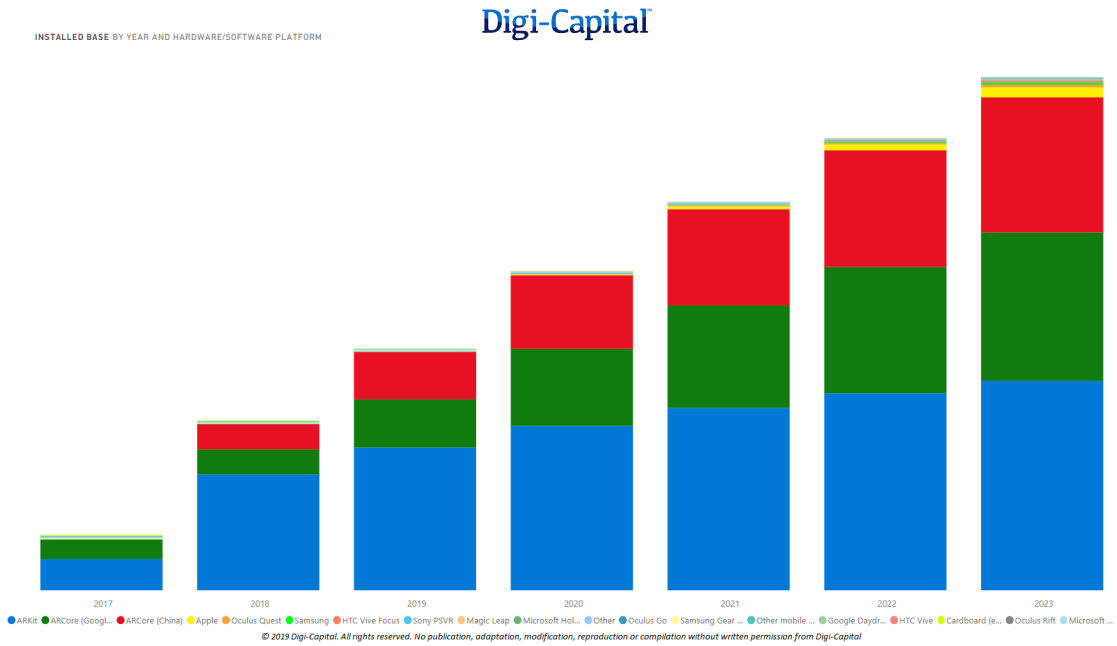

FIGURE 3.1: Technologies using

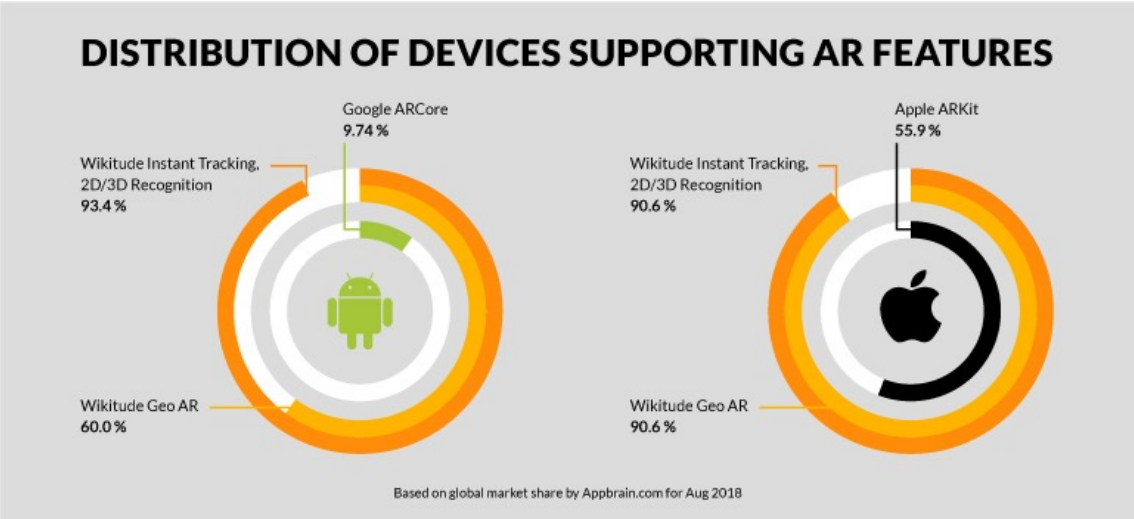

FIGURE 3.2: Distribution of devices supporting ar features

ArKit is Apple's official toolkit for developing augmented reality applications for iOS.

The main features of ArKit are :

- Object and image detection in the real-life
- Object and image tracking in the real-life
- Feature detection in real-time
- Physics engine for AR objects. With this technology, users can interact with objects and their environment.

With the new update to ArKit3, there will be even more great features.

3.3.1 People Occlusion

This feature lets people walk around in the AR app in realtime. Before it was made with some glitch, the system was not able to detect a person who is standing in front of the object and there was an effect like an object is always in front of a person even if a person is standing farther than objects. But now experience has become much more realistic

3.3.2 Motion Capture

One of the main new feature of ARKit3 is the motion capture of the human. Now the system sees human movement as bonded together lines and dot.

3.4 Firebase

What is Firebase and why I will use it for my app? So, Firebase is Google's platform which helps you to build, grow, and improve your app. It is like a bunch of useful services in one place that you should not create by yourself because they are already created by Google. These services are including file storage, push notification, database, analytics, authentification and even more.

FIGURE 3.3: Motion Capture Example

The main point of all is that when we are creating an app using Firebase we don't need to write back-end which should communicate with the database. Firebase gives us SDK which replaces all API calls. Of course, it doesn't mean that we do not need back-end developers in our world, there are a huge amount of tasks that Firebase is not able to do, but in the same moment, it really helps people and companies who want to create a simple app (or even not so simple).

For which purposes will I use Firebase in my project? Let's start from the beginning.

3.4.1 Authentication

Firebase really helps with identifying users. Now I do not need JVT and refresh tokens to identify a user who sent an API request. So it is all about security, with Firebase Authentication you should not worry about it, Firebase will do everything for you. Even if you need to give some special access for different people you always can to configure it using Firebase Security rules.

```
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if false;
    }
  }
}
```

```
service firebase.storage {
  match /b/{bucket}/o {
    match /{allPaths=**} {
      allow read, write: if false;
    }
  }
}
```

FIGURE 3.4: Example of Firebase security rules

Also, Firebase helps you to use different services of authentication, such as Google, Twitter, Facebook, Github, or via your phone number.

3.4.2 Cloud Firestore and Firebase Realtime Database

The most important part of these databases is the real-time updating of your data. In the app, you set up a listener that sends you data on which you have a subscription and afterward each time when this data will be changed in the database you will receive this update. It is something similar to sockets and R sockets where you open a connection and listens for notifying. Using this service your app will always show the user fresh data.

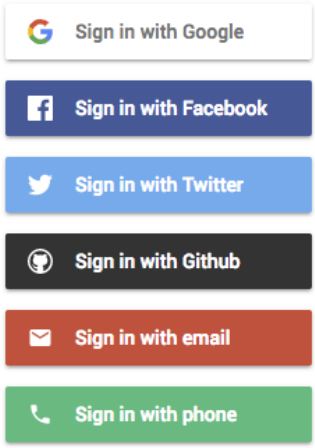

FIGURE 3.5: Possible authorisation methods

3.4.3 Cloud Storage

With could storage, you can upload and download files directly from the app and you should not worry about running out of space.

3.4.4 Firebase cloud messaging

Basically, there are two ways of sending push-notification to users. The first one is when we are using Firebase console and sending it directly to the users and the second one is when back-end sends them. For the first versions, I will not have a back-end so it means that I am interested only in the second approach. If I used the Firebase Analytics and Firebase prediction, I would be able to send target messages, to a specific amount of people with specific parameters.

3.4.5 Firebase Crashlitics

This is the tool that helps you with crashes or exceptions. Crashlitics really a well-done product for mobile development. You can track the statistic of the crashes, which phones produced these crashes, which version of the app they used. Long story short, it is a must-have service in every app.

3.5 Sceneform

Sceneform is a Framework that Google company realized in the 2018 year. The main goal of Sceneform is to help developers to easily work with the ArCore platform without a good knowledge of OpenGL.

Now let see what Sceneform can do.

- ArSeceneView is using camera data from the session to render an image. Planes are highlighted to show us that they are detected by ArCore.
- There is such class as ArFragment. It is as usual Fragment which you can easily add to your .xml file. The main job of this Fragment to let developers not worry about versions of ArCore, permissions that users should accept, etc. So if the user doesn't have the latest version, ArFragment asks him to do it, if

the user did not accept camera permissions before, ArFragment asks to accept them.

- Processing of 3D model is asynchronous
- There are three ways to create Renderable (textures and 3d objects that Sceneform is able to render) 1) from obj, fbx, gltf (3D asset files) 2) from Android widget 3) from shapes

3.6 Location tracker

In my project, I need to track the location of the user because my system needs to know which 3D models it needs to receive to show them on the screen. For example, I am a user of "Street Art Ar" and I want to get to a certain art object. The user has the life area around him and each object has a location. If the location of the object is in the area of the user, the system should start to download this object to show it immediately when the user will reach the location of this art object.

For the location tracker, we need to make a few steps before coding. I will use the Firebase database for keeping all locations of the objects and for this I need to create a Firebase project and get a google-services.json file to get access to Google services.

Now I will explain which method I should use to implement the location tracker.

createLocationCallback - so this function we are calling to get the user's current location. After this, we should update data in the Firebase Realtime Database.

requestLocationUpdates - we need to check if the user has installed the latest version of Google Play Services on his smartphone and this method will help us with this. Inside of this method, we also should check if the user gave permission to receive his location. After this we need to check if the location provider is enabled and then we finally can receive updates from his phone about his location.

onRequestPermissionResult - I have already described above that we need to check the permission status of the location and exact this callback will help us to know what user has decided. If he has denied location access we will show him pop-up which will explain to him that without this permission you as a user of the app can not use this app.

3.7 Google Map

After the location tracking of the user, I need to implement an animated market of this user and dots on the map where he/she should go to see the certain art object. I need to use Google Map API to show the map and markers on it. I will use the following methods for this:

animateCamera - this method will be called only once when the system will receive the location of the user (when the user will open the page with google map). animateCamera will make smooth zooming to the user location.

animateMarkerToGB or addMarker - When the user opens the app I need to use the addMarker method to show marker on the map but if the marker is already on the map I should use animateMarkerToGB method to move the marker to next location. In such an approach, I will create an animation of the moved marker on the map. For static markers that represent Art objects I will receive information about location from the Firebase database and use addMarker method.

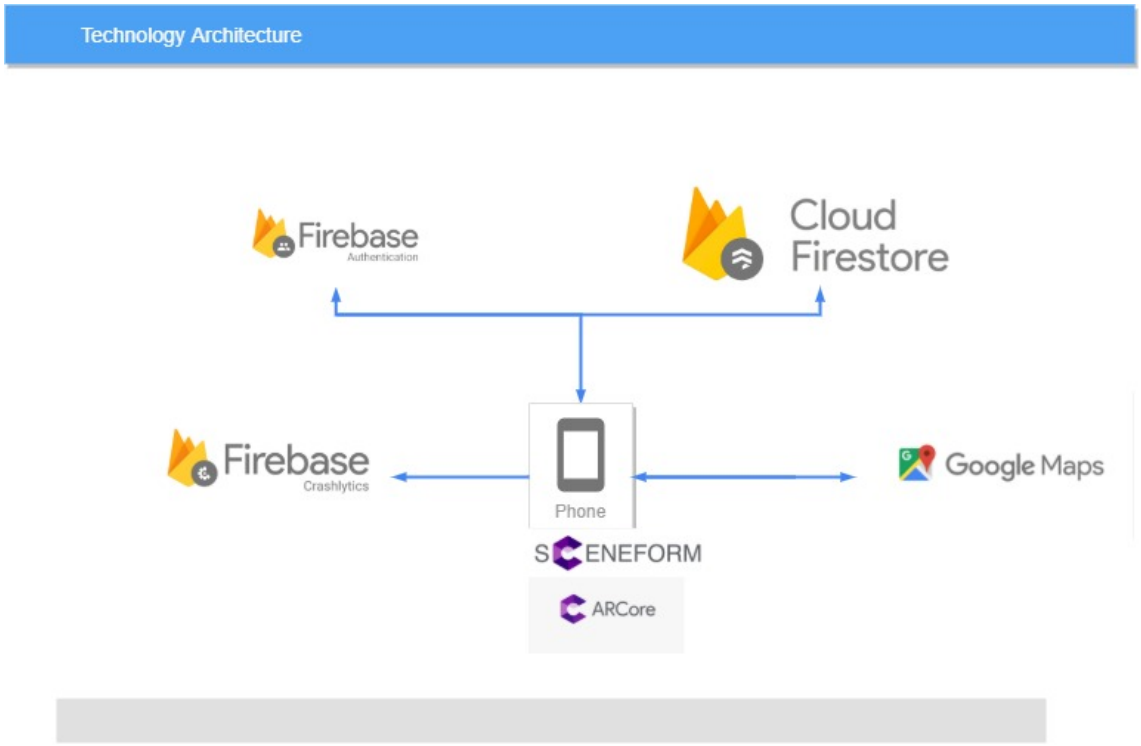

FIGURE 3.6: Technology Architecture

Chapter 4

Versions

1. The user can authenticate via Gmail. The application will have already a downloaded list of 3D objects, which the user can place via ArCore technology.
2. The user can upload his own 3D objects to the Firebase and place them where he wants. The system will remember this place for this particular object.
3. The user can see the Google map and markers on it which will show him the place of other art objects. When the user will be close to 3d object he/she can click on a marker on the Google Map and open the screen which will show him/her this art object in real-world through the smartphone.
4. The user can see the implemented design of this application. Until this step, I can not call my application as ready for users but after this, we can go live.
5. After registration, the user can see the on-boarding tour of the app that will explain to him/her the main features of this application.
6. When the main functionality will be implemented we need to add more authentication methods such as via Facebook, Twitter, or via the phone number.

Chapter 5

Implementation

5.1 Architecture

Long time for Android Application there was no architecture approved by Google. It means that there was no one common approach for developers and people wrote architecture of their application as they wanted. And on IO17, Google has changed this. They presented to the world Android Architecture Components.

So what are the "Architecture Components"? This is the set of libraries that helps Android developers to build the testable, robust and maintainable application

- LifeCycle Aware
- ViewModel
- LiveData
- Room
- Navigation Component

Each of these components is created to solve some problems that have been existing for years until Google stop it.

5.1.1 LifeCycle Aware

Let's discuss the problems that Android developers had before. For example, we want to create an application that tracks the location of the user as soon as he/she opens the app. Before LifeCycle Component was released you as an Android developer always needed to add lifecycle callbacks to your Activity or Fragment to check lifecycle state. But the disadvantages of such an approach are the following: hard maintaining, a lot of errors that can occur, and badly written code. Developers needed to prevent each state of the app and call the right callback. It was very hard to implement in a good and clean way.

LifeCycle Aware component is created to help the developers with the handling of operations that have dependencies on the lifecycle. Lifecycle component can react on each changing of lifecycle status of other components and help us with the organization of the code, maintenance and what is the most important takes some part of handling lifecycle status which developers had to write by themselves that in turn reduces the number of application crashes

5.1.2 ViewModel

ViewModel is a class where the android developer should write all business logic and then pass this processed information to UI elements. ViewModel can be shared and it helps to pass information between different fragments and activity.

ViewModel is always bound with the lifecycle of class where it is initialized (Fragment or Activity) and will be cleared as soon as fragment will be destroyed or activity will be finished. And one of the most useful features, viewModel will not be clear when the user rotates the phone. The new instance of the activity will just connect to this viewModel.

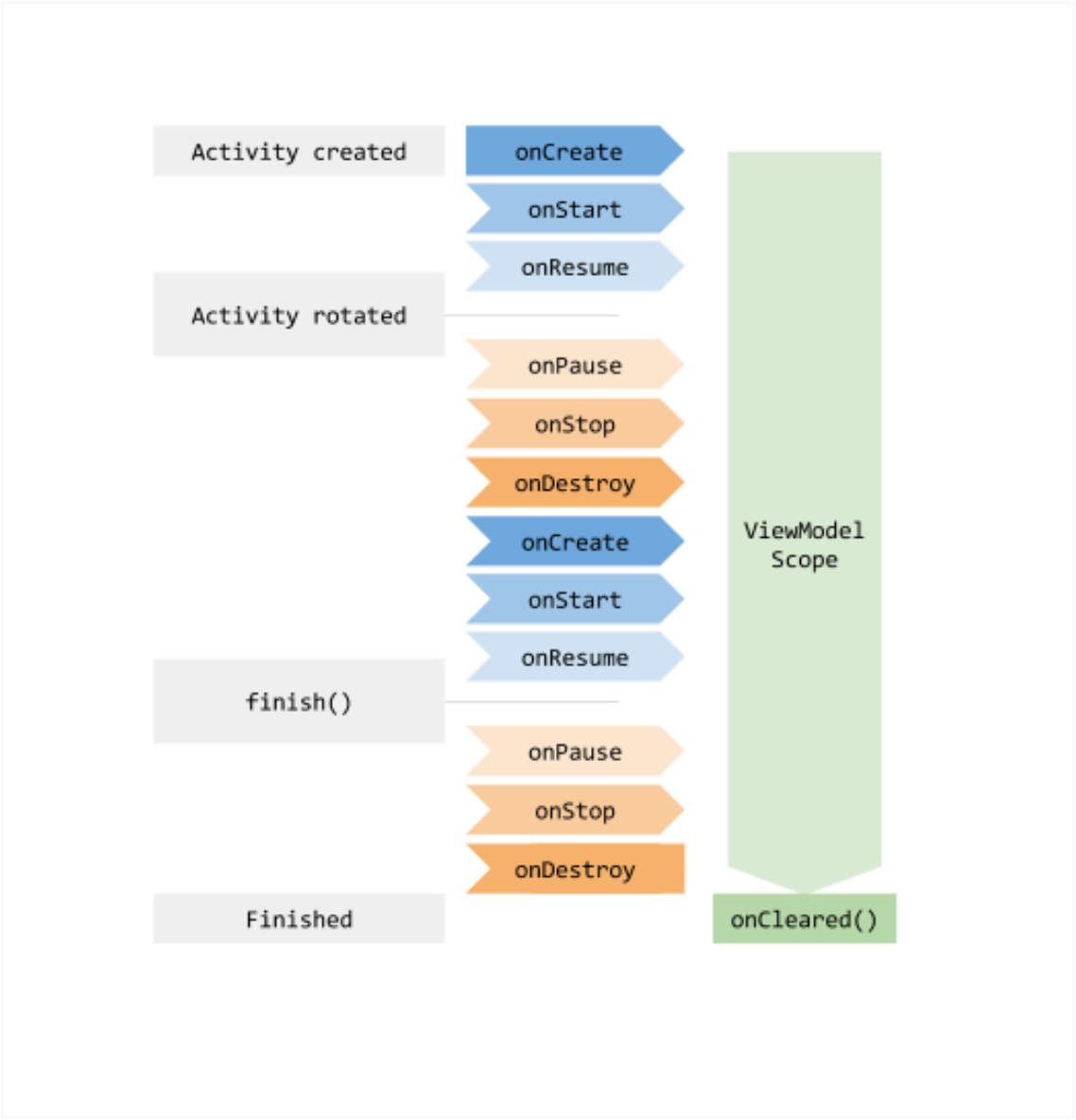

FIGURE 5.1: ViewModel Lifecycle

So the main task of viewModel is receiving the information, processing, and notifying the fragment or activity about any changes. For notifying I will use the LiveData but it will be described a bit later.

So the advantages of the ViewModel are:

- It keeps running when activity is in back-stack

- It is lifecycle aware (look the previous section)
- It is easily combined with LiveData using which we can easily notify fragment about data changes and according to this, change the design
- It keeps living after screen rotation.
- It can be bind with Navigation Graph (It is another Architecture component which I am going to describe in the next sections)

It is very easy to work with ViewModel when you do not need to pass any information through the constructor. This can be the only one a bit complex moment when you work with ViewModels

In case if we need to pass the data through constructor we need to use the ViewModelFactory. If to be precise we need to implement our own factory to create an instance of ViewModel with the custom constructor.

And the last case that I want to describe in the scope of ViewModel is Shared ViewModel.

Shared ViewModel is viewModel which is bound with Activity or navigation graph to pass information through the fragments.

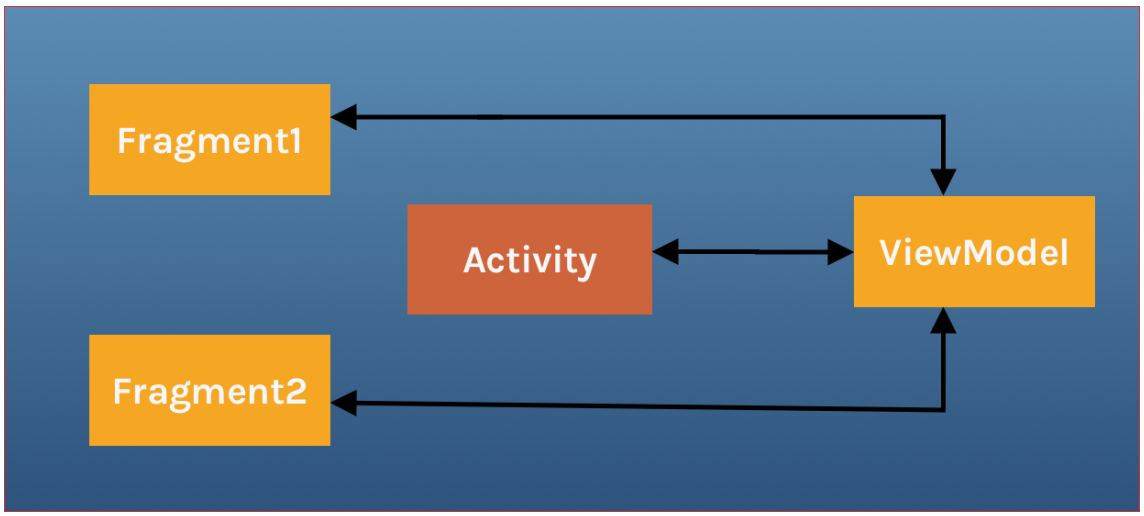

FIGURE 5.2: Shared ViewModel Architecture

5.1.3 LiveData

LiveData is the observable data holder by subscription on which you can get updated value when it changes. LiveData is lifecycle aware so it means that we do not need to worry about any subscription when our application goes to the foreground.

If you understand how the Observer pattern works it means you easily can understand how LiveData works. In MVVM architecture LiveData is used for communication between business logic which is written in ViewModel and UI element which you can change in Fragment or Activity according to the data that you receive.

From Fragment or Activity side you just need to subscribe to the particular LiveData, pass the viewLifeCycleOwner and Observer, as soon as the value of LiveData will be changed Observer will receive it.

There are two methods to update LiveData:

- setValue (update value from the main thread)

- postValue (update value from other thread)

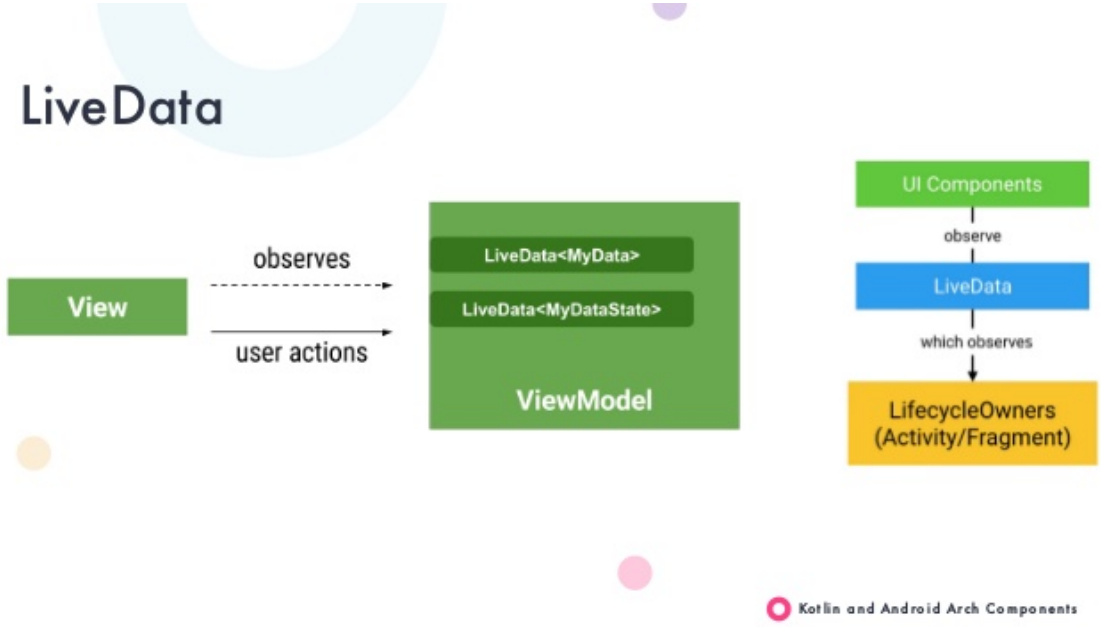

FIGURE 5.3: Live Data Architecture

5.1.4 Room

This other library from the architecture components. Basically, Room is a wrapper on SQLite, not to use all power of it. The benefits of this library are

- It is very easy to use with LiveData
- It is easy to implement

There are three main components of Room such as Entity, Dao, Database. I have experience with writing back-end side on Java so for me, it was quite easy to understand the concept of these components.

- Entity - this is a table in the database.
- DAO - interface for getting data from database and writing data into database.
- Database - should be an abstract class that extends RoomDatabase and it is the place where all DAOs are grouped.

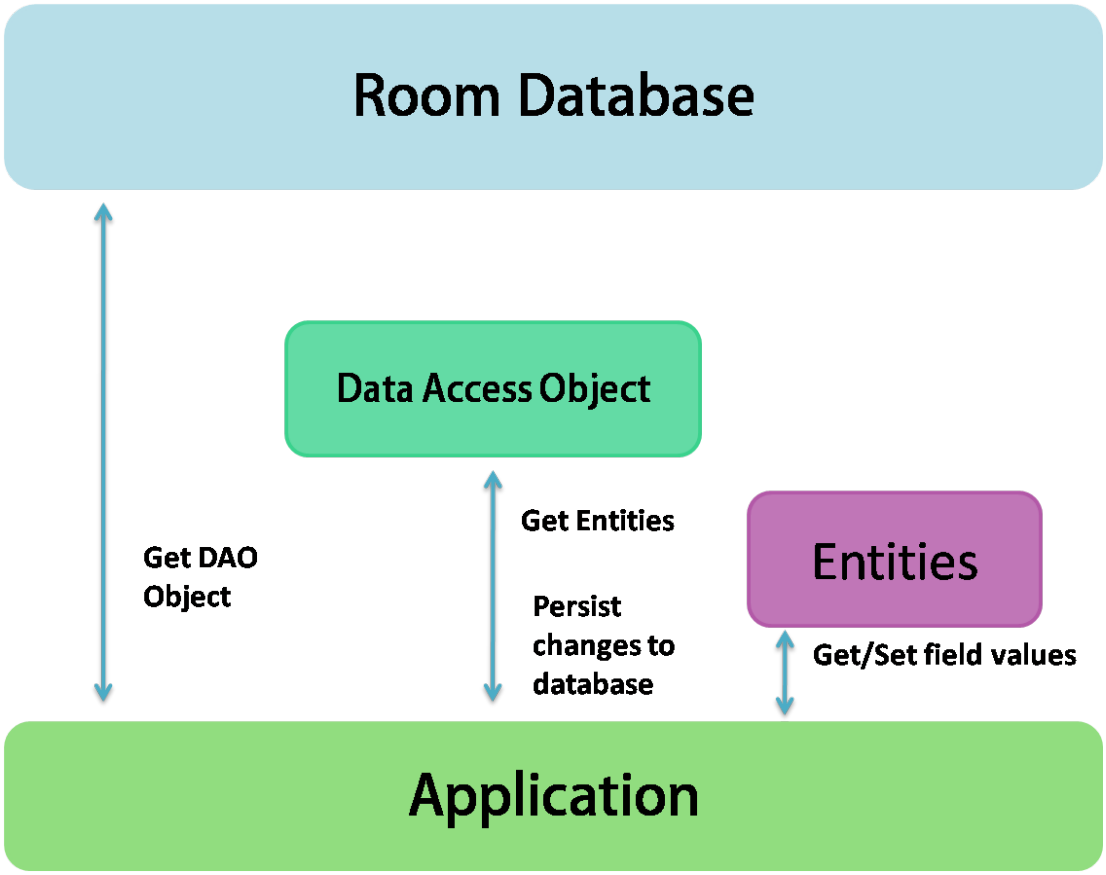

FIGURE 5.4: Room architecture

5.1.5 Navigation component

As you can understand from the name "Navigation Component" simplify android app navigation and also visualize your flows. "Navigation Component" has the following benefits:

- You as a developer can set up default animation and transition
- Easier implementation of deep linking
- Handling of fragment transaction
- Handling of the back stack
- And Android Studio has a tool where developers can see and edit the flows of app

Android Navigation Component has three key part:

- Navigation Graph - this is an XML file where we can describe all destinations and even establish the data that we can pass from one screen to another. But I personally prefer to use ViewModel for this. You can bind ViewModel with navigation graph or sub-navigation graph and it will exist until the user is in this graph. This is one of the most useful things that this library can offer.
- NavHostFragment - this is an XML view where you can specify your navigation graph
- NavigationController - this is an object that tracks the user's movement through the screens and graphs. This object lets you navigate the user to another screen and also return him to one of the previous screens that he had opened before.

For example, my application has the following navigation graph.

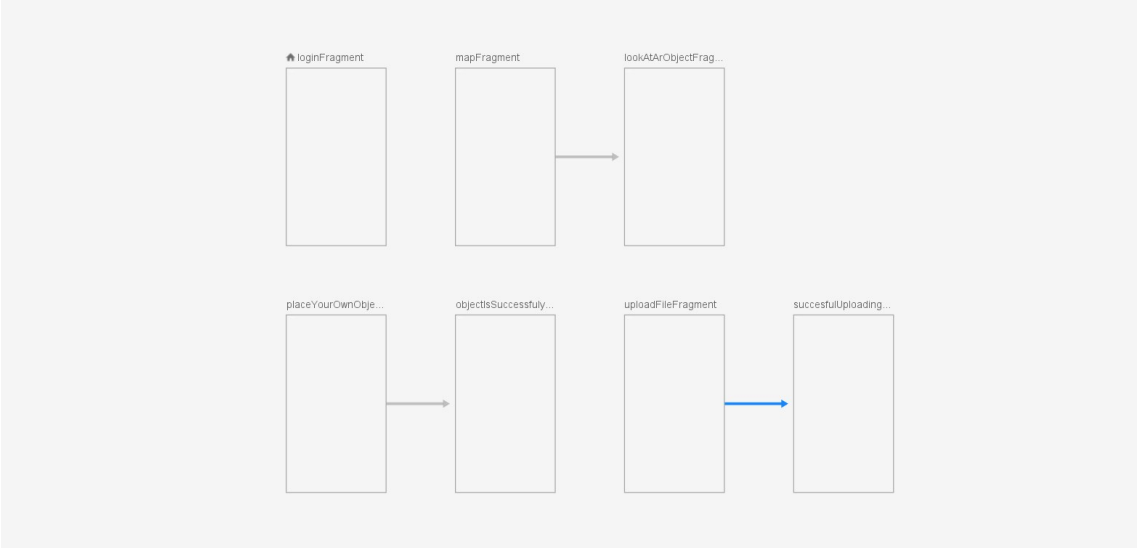

FIGURE 5.5: Navigation Graph

The first page that users will see it is "LoginFragment". There they authenticate themself and move to "MapFragment". "MapFragment" and "Loginfragment" are

not connected because they belong to different flows. If the user is already logged in, he will not go to the "LoginFragment" but direct to the "MapFragment".

On "MapFragment" we need to override the system back button because when the user clicks it, we should not go to the previous screen in the stack but should close the application.

```
91      val callback = object : OnBackPressedCallback( enabled: true) {
92          override fun handleOnBackPressed() {
93              activity?.finishAndRemoveTask()
94          }
95      }
96
97      requireActivity().onBackPressedDispatcher.addCallback( owner: this, callback)
98  }
99
```

FIGURE 5.6: overriding of system back button

If the user is on "MapFragment" and close to some art object he is able to open it in ArFragment, in my case it is the "LookAtArObjectFragment" screen. Using the bottom navigation view the user is able to move to other graph directions which starts from "PlaceYourOwnObjectFragment" and "UploadFileFragment" accordingly.

On "PlaceYourOwnObjectFragment" the user is able to choose the object from the list of uploaded ones and place it where he wants. If the placement was successful, it means that the system has written the coordinates of the art object into Firebase Database and afterward we bring the user to the next screen "ObjectSuccessfully-PlacedFragment" to notify the user that everything went good.

A similar flow has the next direction which starts from "UploadFileFragment". Users click on the button "upload", choose the file, via "Sceneform" they can see a preview of the object, and if everything is good, they approve uploading and system bring them to "SuccesfulUploadingFragment" screen.

5.2 ArCore

Before starting work on Ar with Sceneform we should install Google Sceneform Plugin which helps us with assets exporting. Also from this plugin, we will get 3d viewer where we are able to check how our model will look in the app.

Afterward, we need to add ArFragment to the XML file to make the user's Ar experience easier for the user. Using this fragment we can implement indicators for user's actions, for example, we can add an indicator that the user needs to move his phone to find planes, etc.

```
<fragment
    android:id="@+id/sceneform_fragment"
    android:name="com.google.ar.sceneform.ux.ArFragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
```

FIGURE 5.7: XML tag of adding ArFragment

To use the camera we need not forget to add permissions in applications manifest file. The manifest file is like a description of your application for Google Play, Android build tools and Android operating systems.

After detecting a plane by our ArFragment we need to place the object. After choosing the model, I am receiving a URI of it, looking for the center of the screen, and performing a hit test on that particular point. Sceneform allows us to load model asynchronously and have easy error handling way by using Renderable that I described earlier. Then I need to add this model to the node by using Renderable, Anchor, and ArFragment. The Anchor is the position which we have chosen to place our object. Afterward, I just need to create two nodes AnchorNode and TransformableNode. AnchorNode is the node on the plane where the object will stay even if we move our camera to another place. TransformableNode is the node for scaling our model. TransformableNode needs to be set with Renderable and assigned to the ArchorNode. It allows the model to be scalable and transferable within the place

Chapter 6

Results

6.1 General

During my workflow on this project, I tried to cover all stages of Ar application development from creating an idea until market analysis, from technology investigation until the implementation of this app. The hardest part of this project was to understand how AR works and how to use it because I never had experience with this before. The outcome of all this journey is impressive, I have an application where the users can place 3D models on the plane, upload them to the Firebase Storage and they can see on the GoogleMap where are the art objects of other people.

6.2 Technologies that were used

- Android SDK
- Firebase
- RxJava/RxKotlin
- Koin
- ArCore
- Sceneform
- ViewModel
- LiveData
- Room
- Navigation Component

6.3 What next?

There are two main things that need to be improved. First one is the development of improvements and new features such as:

- To develop a better tracking system of the users when they open the "Google Map" screen
- To develop a better downloading system of 3D objects which are around the user.

- To develop automatic placement of the 3D object when users come close to one of the art objects.
- To develop a gallery of user's art objects
- To develop the functionality of adding of art objects to favorites
- To develop a gamification system. If users visite more art objects, they will get a higher level.

The second thing that needs to be improved is design because for now, I have only basic screens with complex logic behind.

6.4 Summary

In the end, I want to say thank you to all people who helped me with research and understanding of all technologies that I have used in this project. I am sure that I have reached a new level of my knowledge in Mobile development. Especially I want to say thank you to my parents who gave me the opportunity to study in the best Ukrainian universities. Without them I would not have what I have now.

Thank you, for your attention

Best Regards

Maxym Komarenskyy

Bibliography

[1] AR technology for Android

[2] Vuforia

[3] ARKit 3

[4] How is ARCore better than ARKit?

[5] What is Firebase? The complete story, abridged.

[6] Firebase Security Rules

[7] Building a Poly browser using Sceneform

[8] Building ARCore apps using Sceneform

[9] Android Architecture Components

[10] Guide to app architecture

[11] Handling Lifecycles with Lifecycle-Aware Components

[12] LiveData Overview

[13] ViewModel Overview

[14] Room Persistence Library

[15] Android Architecture components walk-through

[16] How ViewModels Work on Android

[17] Navigation Component- The Complete Guide

[18] Cloud Firestore

[19] Firebase Authentication

[20] Cloud Storage

[21] Firebase Realtime Database

[22] Firebase Crashlytics

[23] Firebase Cloud Messaging