Huiwen Duan

3D Relics

A Standalone Augmented Reality Mobile Application

Helsinki Metropolia University of Applied Sciences

Bachelor of Engineering

Media Engineering

Thesis

01 May 2013

Author(s) Title Number of Pages Date	Huiwen Duan 3D relics, a standalone augmented reality mobile application 45 pages + 3 appendices 01 May 2013
Degree	Bachelor of Engineering
Degree Programme	Media Engineering
Specialisation option	Mobile Application Development, Hybrid Media

Instructor(s)
Supervisor(s)
	Kari Salo, Principal Lecturer

This project was carried out by Media Engineering and Media degree programmes at Metropolia University of Applied Sciences, with the purpose of showcasing 3D models of historical sites to tourists as an augmentation of reality. To address the requirements, a system consisting of a server side web application and an Android mobile application were designed and implemented.

The web application was built on Amazon EC2 cloud, using Linux, MySQL, PHP and Apache as a server environment. An admin site was created for user to upload 3D models, and a web service was built using the principle of representation state transfer (REST) to provide a public interface to outside world. The mobile application was developed on the Android platform using Android SDK and Metaio SDK. It is a standalone Android application capable of displaying AR content with no dependency to internet.

The results proved that a system of deploying 3D models through a web client and displaying them as a form of augmented reality on an internet-disabled mobile client was feasible. With the rapid growth of mobile computing power and accuracy of inertial sensors, more features can be brought to the mobile client, such as location based tracking.

Keywords	Augmented reality, 3D, Android, Mobile Application

Abbreviations

AR	Augmented Reality
AV	Augmented Virtuality
API	Application Programming Interface
VR	Virtual Reality
SDK	Software Development Kit
App	Application
3D	3-dimensional
HTTP	Hypertext Transfer Protocol
URL	Universal Resource Locators
REST	Representational State Transfer
CRUD	Create, Read, Update, Delete
GPS	Globle Pointing System
XML	Extensible Markup Language
JSON	Javascript Object Notation
LLA	Latitude/Longitude/Altitude
QR	Quick Response
COS	Coordinate System

Contents

1 Introduction 1

2 Augmented reality 2

2.1 Definition 2

2.1.1 Marker-based 2

2.1.2 Location-based 3

2.2 AR fundamentals 3

2.2.1 Display types 3

2.2.2 Workflow of AR system 4

2.2.3 Typical AR coordinate systems 5

2.3 AR in real life 7

2.4 Mobile AR development toolkit 8

3 Project architecture 11

3.1 Web application 11

3.2 Mobile application 11

3.3 Use cases 12

4 Design and implementation of mobile application 14

4.1 Development platform and toolkits 14

4.2 User experience design 15

4.3 Application class diagram 17

4.4 Implementation of the main Android component 19

4.4.1 SQLite database 20

4.4.2 Request and parse data 21

4.4.3 Load and cache online image 22

4.4.4 Download manager 24

4.4.5 Event complete receiver 24

4.4.6 Background service 25

4.5 Implementation of AR features 27

4.5.1 Content creation 27

4.5.2 Setup configuration file 28

4.5.3 AR as an Android activity 31

5 Testing 35

5.1 Controlled environment testing 35

5.2 Field testing 37

6 Result and discussion 39

7 Future development 40

8 Conclusion 41

References 42

Appendices

Appendix 1. 3D Scene content creation

Appendix 2. Web client user manual

Appendix 3. Mobile client user manual

1 Introduction

In the last few decades, people have been making great efforts on utilizing digital technology in various fields. One of the achievements lies in the augmentation of the real world scene with virtual information, which is nowadays the augmented reality (AR) technology. The concept of this technology was first introduced in 1960s by Boeing [1], and was then effectively used in the medical field, military and movie making. With the growing acceptance of multimedia and big leaps in performance in personal computing device, this technology was brought back to limelight in recent years.

Augmented reality is a multi-disciplinary technology that requires the collaboration of people from different fields, such as image processing, human computer interaction, human factor, networking, computer vision and some other fields. With the combination of the aforementioned technologies, an augmented reality system is designed to enhance the human perception of the world.

This project was initiated by Media Engineering and Media degree program at Helsinki Metropolia University of Applied Sciences as a research project. The initial intention of this project was to bring the historical ruins back to life with 3D models.

As a result of recent development in augmented reality technology, the concept of demonstrating the virtual 3D content in a form of AR is becoming increasingly popular, which is why this technology was chosen as a topic for this project. Targeted towards tourists, the client application should be portable to let them access it at anytime and anyplace. Considering the fact that foreign tourists may lack WI-FI connection or have relatively limited data roaming plan, accordingly the application should be able to function under no internet access. Eventually, the project was defined to be a standalone augmented reality application on mobile platform.

2 Augmented reality

The term "Augmented Reality" has been a trending word on the Internet lately, as well as two other terms: augmented virtuality (AV) [2] and virtual reality (VR) [3]. In fact there is no strict boundary between AR, AV and VR, as shown in the continuum in Figure 1, at one end of the continuum is the real world which is the scene the viewer sees through his naked eyes, and at the other end is completely computer generated virtual content.

The more virtual content there is, the more the scene tends to be AV. And by today's conventions, anything that fits into the middle of the continuum can be classified as AR.

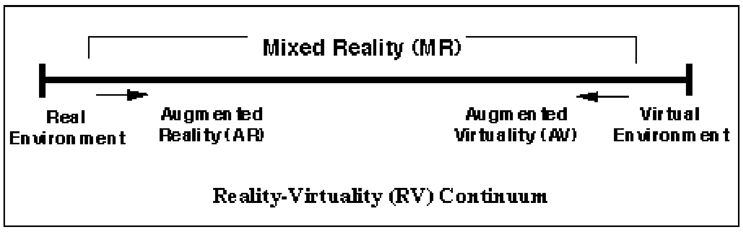


Figure 1. Simplified Reality-Virtuality (RV) Continuum [4]

2.1 Definition

Augmented reality is a live, direct or indirect, view of a physical, real-world environment the elements of which are augmented by computer-generated sensory input such as sound, video, graphics or GPS data [5]. In real life, the augmented elements can be anything, furniture, human faces, trees, buildings and so on. All of these elements can be generalized into two categories, marker and markerless.

2.1.1 Marker-based

To make the detection process easier, a marker is often introduced in a simple AR system.

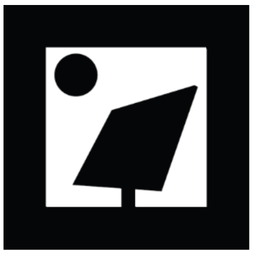


Figure 2. Marker [6]

As showed in Figure 2, a marker is a black and white square-shape. It needs to be positioned in front of the camera during the detection process. The captured video stream is analyzed and the marker is used to determine the coordinate system of the world captured by the camera. The 3D engine obtains the coordinate system and renders the virtual object onto the corresponding position in the original video.

2.1.2 Location-based

Questions arise that it might not always be feasible or aesthetic to attach a marker onto the object which is going to be overlaid with virtual information. As a result, markerless tracking was introduced. Besides camera, various types of sensors, for example accelerometer, compass and GPS sensor, have been applied in the tracking process.

2.2 AR fundamentals

2.2.1 Display types

The key part of an AR system is to overlay a virtual layer on top of the real scene. To make it possible there must be a device in between to connect the end user with the system. Three types of displays will be introduced below.

Head mounted display (HMD)

HMD is one kind of display that the user should wear on the head, on which a pair or single optical is right up against the user's eye(s) to display the output data [7]. Wherever the user turns his/her head, a real time augmentation should be presented to the user's field of view. HMD frees the user’s hands, which imposes no limitation to movement of hands. Before the release of Google Glass, most of the optics on HMD blocks the users' view and a cable is usually required to connect power supply or computing devices, thus making this type of display not portable. However, with the advent of Google Glass, as shown in Figure 3, more intriguing features in the realm of HMD for AR can be expected.

Figure 3. Google Glass [8]

Handheld display

With a series of smart phones coming to the market, AR has become a technology that is accessible to customers. With a simple application installed on the mobile phone, the phone can "interpret" what the camera "sees", rendering the virtual contents and eventually showing them to the end user on the small screen. During the rendering process, the mobile system will adopt the data from different on-board sensors [1], for example GPS, accelerometer and compass, which determines the location, distance from the ground to the device and the field of view respectively and displays the corresponding information received from the internet.

Spatial display

Instead of wearing a HMD or carrying a handheld device, the spatial display projects graphic information onto physical surfaces [1]. The biggest advantage of this kind of display is that it separates the users from the systems, which on the other hand enables groups of users to collaborate concurrently since the system is not associated to any specific user. Superior to HMD and handheld devices, the spatial display has no limitation on screen resolution, and a series of projectors can expand the view. The drawback of this display is that it does not work well under sunlight [1].

2.2.2 Workflow of AR system

The key part of AR workflow lies in the marker detection process. As shown in Figure 4, the real scene is captured by an input device (camera), the stream is sent to the system and analysed, and then the position of the augmented element is obtained. The graphic system renders the virtual object on the detected position. After the detection and rendering process are done, the virtual content is merged with the original real world scene, and at last sent to the display.

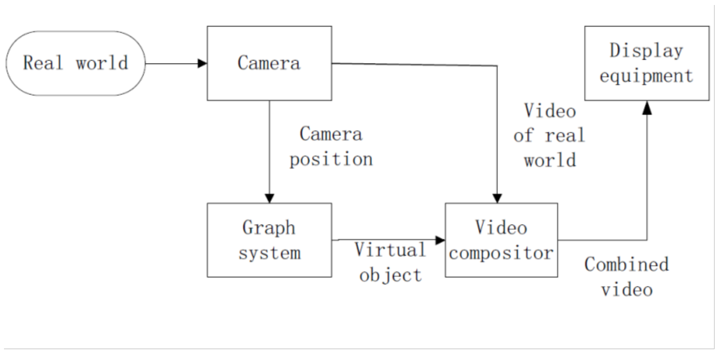


Figure 4. Workflow of AR system [9]

In the real world, the detection process can be relatively complex, especially in the markerless detection. However, under the efforts of mathematicians, series of algorithms are already there for the programmers' advantage. The most frequently used algorithms include skin a detection algorithm which is the process of finding skin-colored pixels and regions in an image or a video [10]. A gesture detection algorithm will draw a convex hull over the detected hand and count the amount of defects inside that hull [11]. If nine defects are found then it is a hand with five fingers. There are more algorithms in use, but in the scope of this thesis, they will not be discussed.

2.2.3 Typical AR coordinate systems

To make an AR system effective and interactive and to give users the feeling of being immersed in the augmented world, the motions of the user must be reflect ed onto the user's view appropriately and concurrently, which in other words is to maintain the computer graphic coordinate system aligned with the real scene coordinate system.

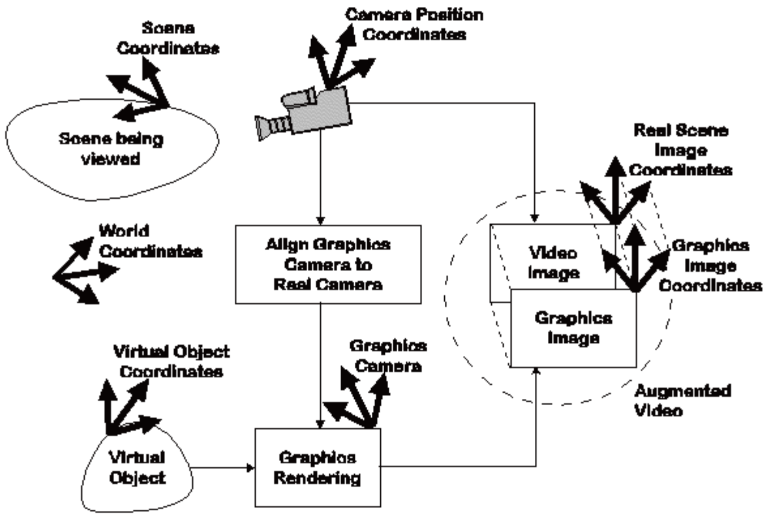

Scene Coordinates
Scene being viewed
Camera Position Coordinates
Real Scene Image Coordinates
World Coordinates
Align Graphics Camera to Real Camera
Video Image
Graphics Image Coordinates
Graphics Image
Virtual Object Coordinates
Graphics Camera
Augmented Video
Virtual Object
Graphics Rendering

Figure 5. Computer vision coordinate system [12]

As shown in Figure 5, there is no nature connection between those coordinates, thereby a series of mathematics calculations are performed to realize the alignment, as shown in Figure 6 and Equation 1.

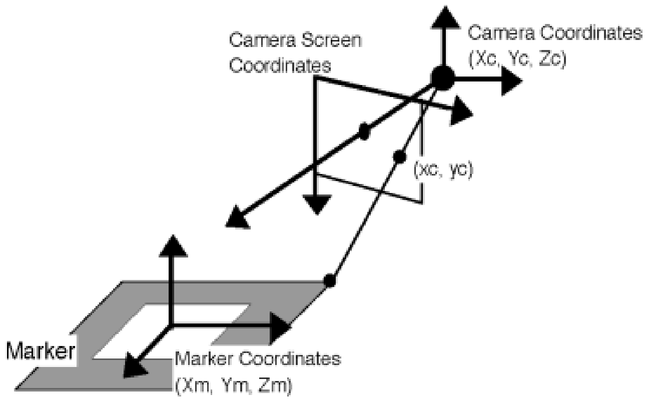

Camera Screen Coordinates
Camera Coordinates (Xc, Yc, Zc)
(xc, yc)
Marker
Marker Coordinates (Xm, Ym, Zm)

Figure 6. The relationship between marker coordinates and the camera coordinates [12]

$$\begin{bmatrix} Xc \\ Yc \\ Zc \\ 1 \end{bmatrix} = \begin{bmatrix} V11 & V12 & V13 & Wx \\ V21 & V22 & V23 & Wy \\ V31 & V32 & V33 & Wz \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xm \\ Ym \\ Zm \\ 1 \end{bmatrix} = \begin{bmatrix} V3\times3 & & & W3\times1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} Xm \\ Ym \\ Zm \\ 1 \end{bmatrix} = Tcm \begin{bmatrix} Xm \\ Ym \\ Zm \\ 1 \end{bmatrix}$$

Equation 1.

(Xc, Yc, Zc) represents the camera's coordinate system, while (Xm, Ym, Zm) represents the marker's coordinate system. V3x3 is the rotation component and W3x1 is the translation component. This calculation is performed to get the translation matrix of the marker's coordinate system with respect to the camera's coordinate system. After the two irrelevant coordinate systems have been aligned to each other, the central project model is used to project 3D points in the camera coordinate systems into the image plane, which is seen by the screen users.

2.3 AR in real life

AR has existed for more than forty years, and it has been applied in numerous fields. It has made profound improvements on how people view the world. However, prior to 1999, the expensive and software dependent technology which worked on bulky equipment had always been an unknown yet growing field to individual customers, though they were massively used in military, surgery and other industries [13]. However, with the flourishing of mobile computing devices in recent years, and the release of ARToolkit, which will be discussed later, now AR is making its way to general populations. Several applications are showed below.

KARMA

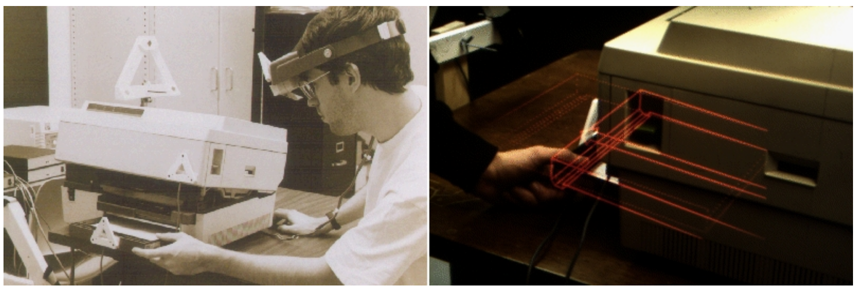


Figure 7. KARMA [14]

KARMA (Knowledge-based Augmented Reality for Maintenance Assistance), as shown in Figure 7, is an HMD AR system. KARMA was one of the functioning AR systems in the early phase of the AR history. These systems helps the user load and install the machine without reading the instructions [14], as a 3D model is drawn on top of the object to guide the user.

Live sports game

AR technology is nowadays widely used in broadcasting [15]. For example, the advertisements and logos at the sides of the pitch that usually do not physically exist, but still audiences see them on TV. That is because when the video is beamed through airwaves and additional graphical information is often overlaid onto the original pictures.

Special effects

AR has been helping movie makers for years. Davy John, the character with an "octopus face" in Pirates of the Caribbean is one of the cases.

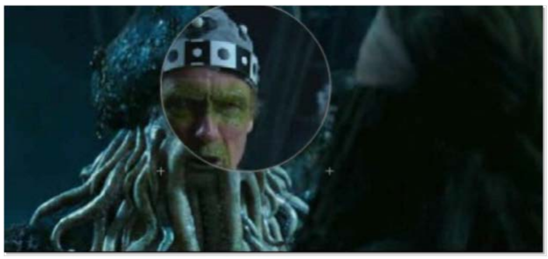


Figure 8. Pirate of Caribbean [16]

With the rectangular markers that the actor wore on his head, as shown in Figure 8, the AR software can easily detect the right position when analysing the video stream, and superimpose those areas with a virtual 3D model.

2.4 Mobile AR development toolkit

One of the key difficulties in AR development lies on the tracking of the user's field of view. In order to know from what viewpoint to draw the virtual imagery, the application needs to know where the user is looking at in the real world, which imposes great challenges to programmers because of the massive mathematic calculations.

Thanks to the talented programmers, a good number of development toolkits nowadays offer programmers easy approaches to develop simple AR applications, either marker based or non-marker based. One of the features that made these toolkits so handy and welcome is that those toolkits can auto detect the marker position and with that data programmer can overlay virtual content on top of it (based on computer vision algorithm), which alleviates the calculations from the programmer’s side.

Layar

Layar is a mobile platform for discovering digital information. They provide two approaches to view the AR contents, as can be seen in Figure 9. The first approach is through their Layar browser, to which all the data comes in as the form of “layers”. These layers can contain digital information related to both geo-location as well as object such as images and videos. Developers have to obtain a developer key, create their custom layer and then publish to the layar publishing site. The other approach is through native application, which is also referred to as layar player on their official document. However, developer also needs to publish his “layer”, as mentioned in the first approach. By sending request to the layar server, the native application can demonstrate the AR content.

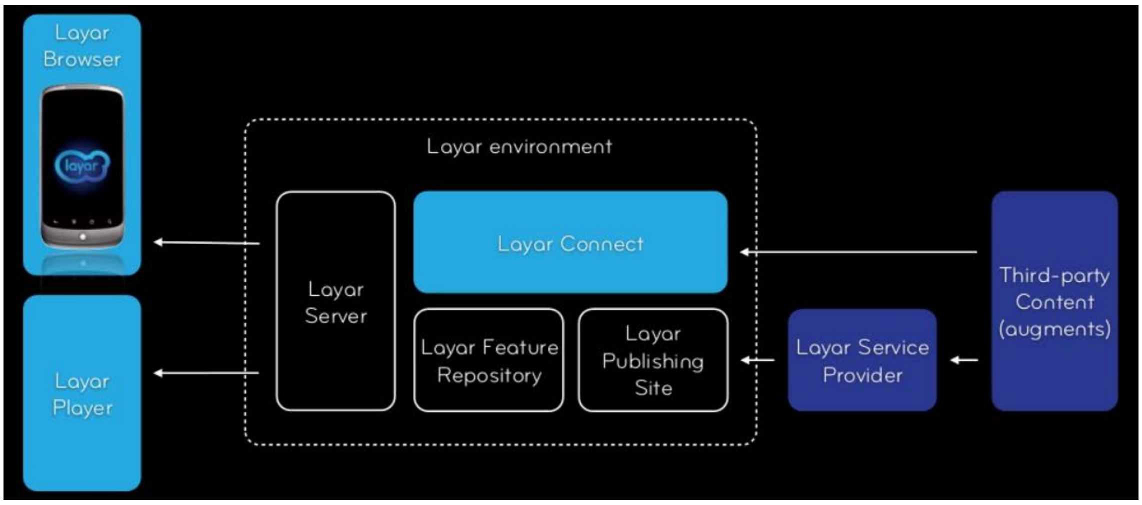


Figure 9. Layar platform.[17]

However, neither of the methods provide a “offline” mode, as the application always has to request the remote layar server, and internet connection should be guaranteed for the application to be functioning.

Metaio

Metaio SDK is a software development toolkit for developing native AR applications, which has libraries for IOS, Android and Windows. Metaio SDK includes a powerful 3D rendering engine, which wraps up functions calls to OpenGLES and implements their own mathematical algorithms of detecting and tracking the markers. Figure 10 reveals how Metaio SDK works as a middle man on different platforms.

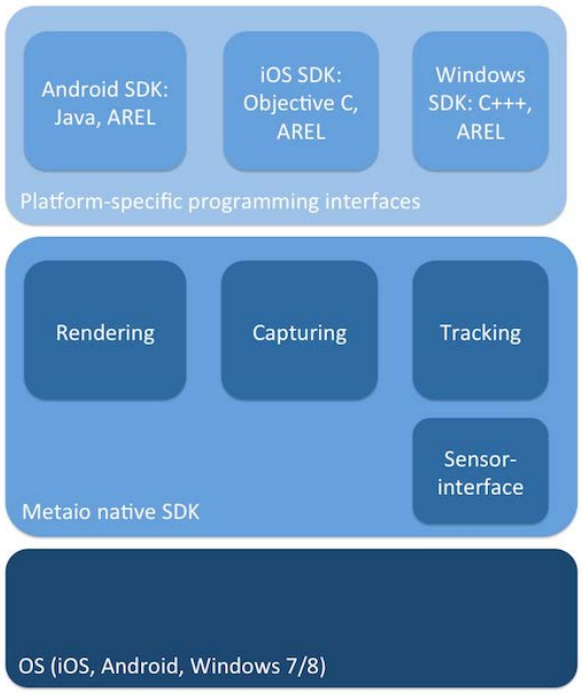


Figure 10 [18]. Metaio architecture

By the time this project was started, Metaio SDK was the only major SDK on the market that supports 3D rendering.

Wikitude

Simillar to Metaio and Layar, Wikitude provides Wikitude browser as an independent application, Wikitude SDK for developing native augmented reality application on mobile platforms, Architect for integrating the AR features to the native application by using web technologies (HTML, Javascript, and CSS)[19], and Wikitude plugin for seamless PhoneGap integration. However, at the time the project was started, Wikitude SDK did not support 3D rendering and, it was not selected by the project.

3 Project architecture

The users groups of this project are mainly students and teachers of Media degree program in Metropolia and tourists, whose purposes of using the system are different from each other. The students and teachers at Metropolia want to deploy their 3D models of ancient times in a meaningful way, while the tourists want to gain more knowledge about the place they are visiting. The system was designed to have two parts, the server side web application which was developed by Carol Rodriguez Torres and mobile application, separating the user group from each other.

There are three components involved in this communication system, and they are interacting in the way shown in Figure 11.

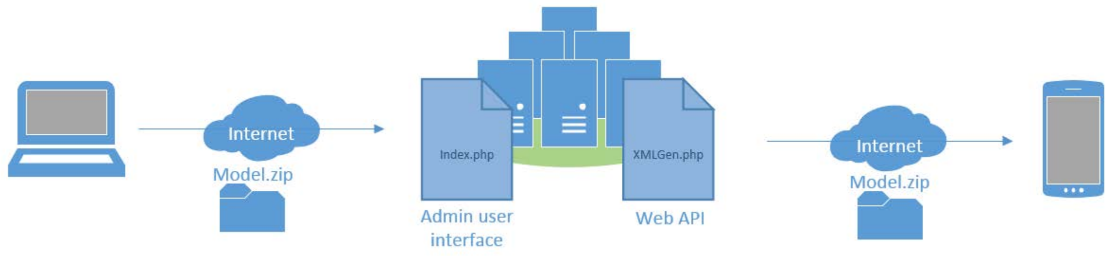


Figure 11. System communication diagram

3.1 Web application

The web application was deployed on Amazon ec2 cloud, using LAMP as a web server. It has a user interface for students and teachers to upload their 3D scenes and related information. All the files and information will be stored in the MySQL database and hard disk on the server. In order to support effective interaction between server and mobile phones, a web service is provided to the outside world.

3.2 Mobile application

By requesting the web service URL, the mobile application will receive a bundle of data which consists of all the information about the available scenes. After been properly handled, those data will be presented to tourists. Therefore, visitors can choose to download the scene(s) and launch it (them) at any time. Considering that the visitors may be in lack of internet connection as they arrived at the historical spot, the augmentation functionality was also designed to be usable under non Wi-Fi or 3G-enabled situation. In other words, the mobile application can be a standalone application, as long as the models were downloaded beforehand.

3.3 Use cases

This section discusses about the functionality of he end products from user's point of view. Figure 12 depicting all the actions user can perform at different states. As explained in earlier section, there are two end user groups for the system. The only connection between the two groups exists virtually, i.e., through the server, thus no direct communications is required or necessary. Figure 12 and Figure 13 demonstrate the two cases of use; each one is independent from the other.

Use case of web client

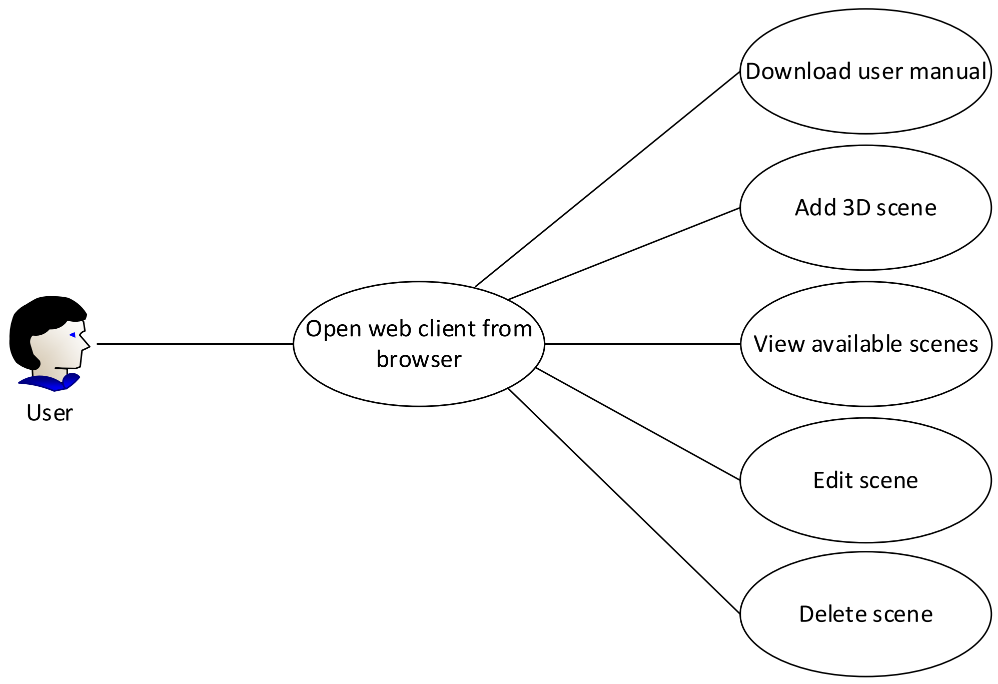


Figure 12. Use case diagram from the perspective of web client user.

As can be seen from Figure 12, the user can conduct five operations once (s)he has entered the web client interface. If the user is not familiar with the system, (s)he can refer to the user manual for guidance, which is available for downloading on the main page. For adding 3D scenes, the user has to go through several steps, input the name and description of that scene, upload the markers files separately and upload a zip file consists marker files and model related files. To update the information of one scene, for example to replace the old 3D model with a newer version, the user can click on the edit button to accomplish such action.

Use case of mobile client

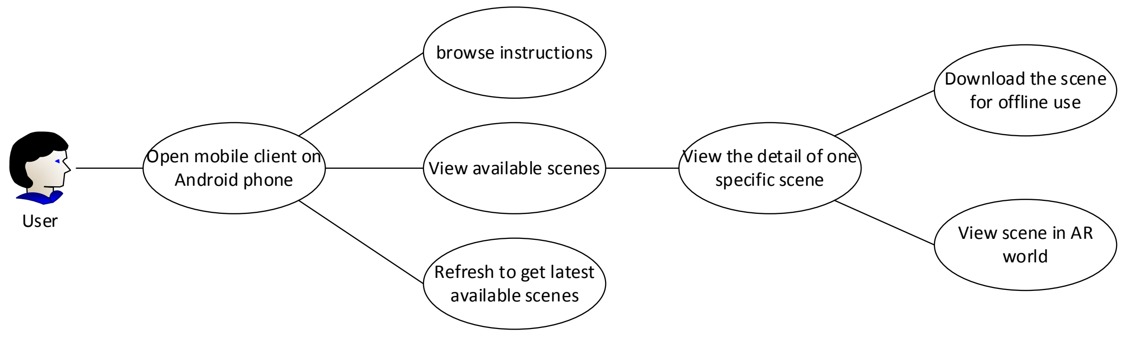

browse instructions
Download the scene for offline use
User
Open mobile client on Android phone
View available scenes
View the detail of one specific scene
View scene in AR world
Refresh to get latest available scenes

Figure 13. Use case diagram from the perspective of mobile client user.

Figure 13 displays the steps user should accomplish in order to trigger the augmented reality functionality. When the user enters the mobile application, several view of instruction and introduction to the application is offered, however, the user can skip this part at his or her desire. The user then will be presented all the available scenes reside on the server, (s)he can browse through and choose the one that interests him or her. As the user enter the detail view of the scene, (s)he can read about the description of the scene and decide whether to enable the for offline use, or if the scene has already been download, the user can start to explore the AR contents through the camera.

4 Design and implementation of mobile application

4.1 Development platform and toolkits

During the past few years, the adoption rate of smart phones has increased dramatically, meanwhile, one question is prompted to the developers, on which mobile platform should they build their application.

At the time the project was started, there were four operating systems excelled in respect to their market share. As shown in Figure 14, they were Android, IOS, Blackberry OS and Windows Phone OS, among which Android dominated the market with a share of 61%, almost three times as much as the runner-up. According the statistics from Google, 850000 Android devices are activated every day. The total of Android devices in the world would be 300 million [20]. In addition to its penetration in the market, Android was and still is the only platform that comes with an open source development toolkit, enabling developers to study the lower level implementation of Google's elegant design pattern.

Therefore, to obtain a larger audience for the application, Android was chosen as the project's build environment.

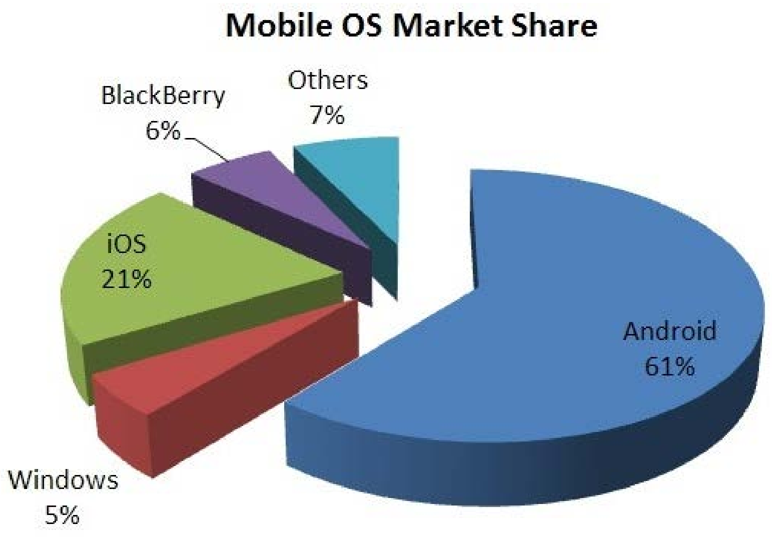


Figure 14. Mobile operating system market share at the time the project was started [21]

Android SDK

Android SDK provides a set of development tools to develop mobile application on Android operation system, which includes debugger, libraries, and a handset emulator [22]. Currently it supports development on Linux, Max OS X10.5.8 or later, windows XP or later.

Metaio SDK

As discussed in previous chapter, Metaio SDK enables the application to be programmable of implementing all the interactions features with 3D AR content and all the tracking and rendering will be done locally on the mobile device.

To use the Metaio SDK on Android device, there is a minimum requirement on the device hardware, which is listed as follows:

- Android 2.2
- ARMv7(Cortext) processor
- OpenGLES 2.0 support
- Camera
- GPS(location), Accelerometer and Magnetic sensors for GPS/Compass based tracking [23]

As mentioned before, due to the particularity of the target user, the mobile application is supposed to be used under non Wi-Fi or 3G environment in most of the cases, which is also the key and crucial point of this application. Apart from most AR web services nowadays, whose calculation and processing are done in a robust remote server maintained by the AR provider, all the AR related functionalities in this mobile application are done locally, including detect and track the marker, render the 3D scene and overlay the AR content on top of the real world in correct position. To simplify the development process of rendering the AR content, Metaio SDK was utilized.

4.2 User experience design

User experience is what a person feels about the relevance and effectiveness of the digital solution they interact with. It is also about the meaningful aspects of how people see, hear and touch it. Good user experience involves various factors, including clear user interface, smooth operation, and the ease of understanding. Many applications nowadays tend to be very task focused, instead of providing multiple scattered features within one application; each app serves one single purpose. The popularity of Any.Do [24] tells the story.

To bring a pleasant user experience to the users of 3D Relics, the design process was carried out in two folds, design in the perspective of user interface and the functional performance of the application. The user interface of 3D Relics can be classified into three categories.

The first category is the landing view, which contains one welcome view and several instruction views. This category was mainly designed to familiarize the users with what the application is about and how it should be used. Swipe gestures were implemented in the instruction views, thus the user can navigate back and forth from one view to anther freely and thereby make browsing and consuming data a more fluent experience [25].

The second category is the views to present available 3D model scenes. As each scene has a list of meta-data, displaying all the data will occupy rather large section of space. As the size of mobile screen is quite limited, the user needs to scroll up and down in order to find the exact scene he is interested in, which as a result is time consuming and leads to a bad user experience. To solve the problem and give the user a direct observation of the available scenes, the application uses two of the most informative and representative data of each scene and display the all the scenes in a list view. Once user finds their desired scene, he or she can click on the scene and new view prompts to demonstrate more detailed information of that scene.

The last category is the AR view. Because this view requires most of the interaction between user and device, it is import to keep the user not to be distracted by any elements irrelevant to the interaction. Although this view is comparatively straight forward, it overlays the AR content on top of the camera view, while no additional UI components are included.

4.3 Application class diagram

The mobile application consists of seven packages, each package performs different tasks independently or sometimes in collaboration. The classes' inner-relationship and the methods and attributes are illustrated in the following diagrams.

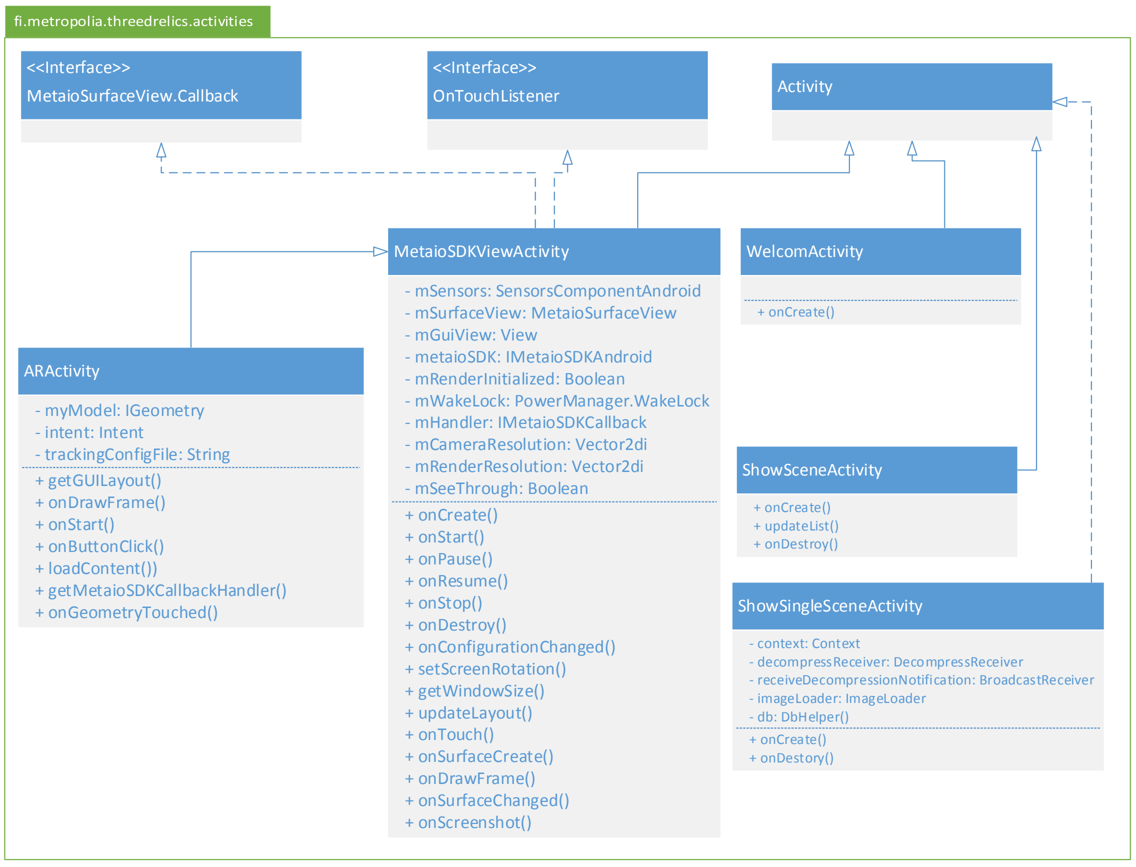


Figure 15. Class diagram part 1.

Figure 15 demonstrates the classes in *fi.metropolia.threedrelics.activities* package. This package contains all the activities in the application. There are four views designed to be shown at different stages of the application.

When user enter the application, *WelcomActivity* is created to instruct the user step-by-step on how the application should be used. After the user has gone through or skipped the instruction views, the *ShowSceneActivity* is presented to show the available scenes on server. When user clicks on one scene, the *ShowSingleSceneActivity* will provide a detailed view of the scene, while *ARActivity* implements the AR related features.

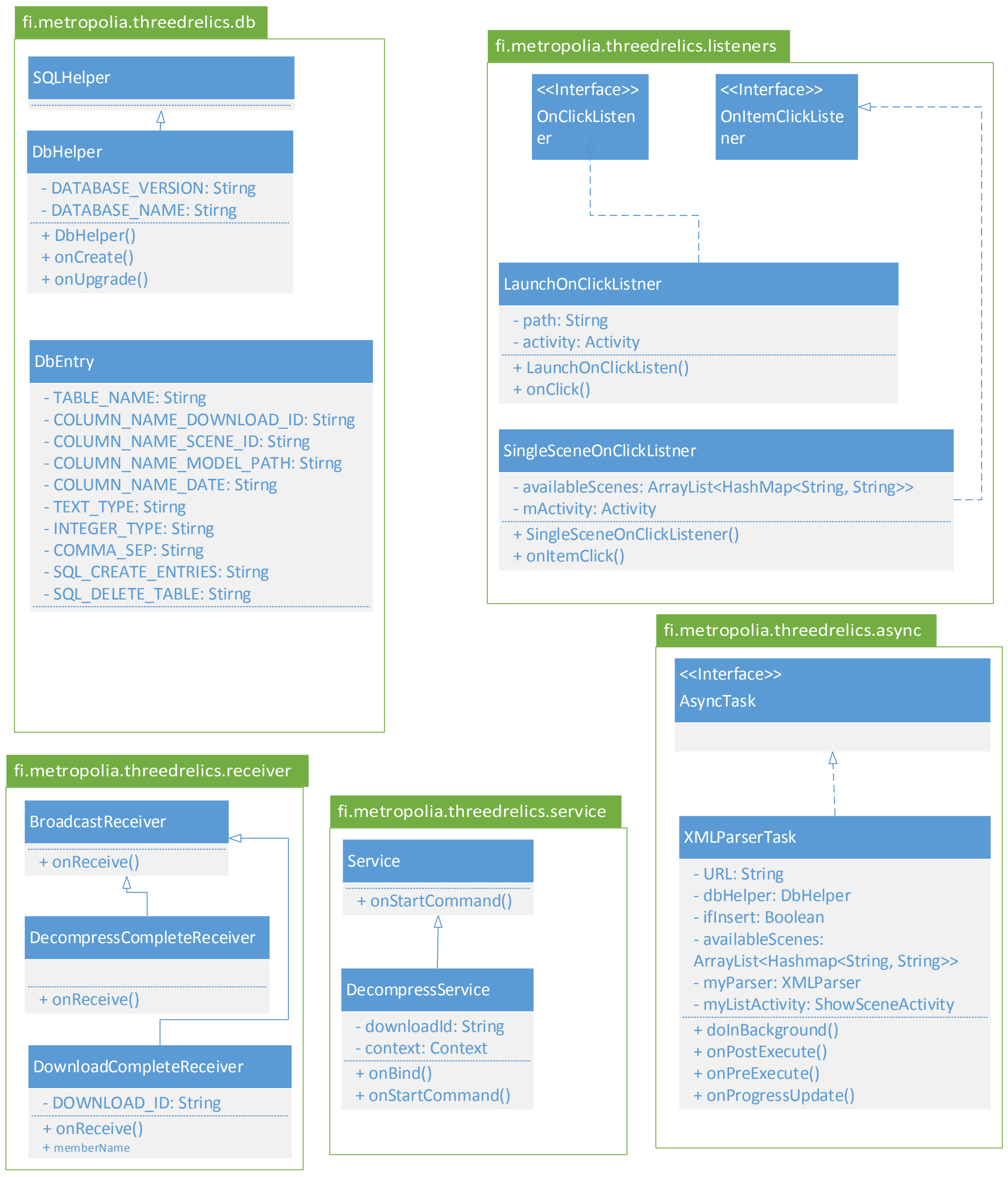


Figure 16 class diagram part 2.

Five packages are shown in Figure 16, which contain the implementation of communication with device’s local database, background running services, broadcast receivers, asynchronous tasks and on click listeners.

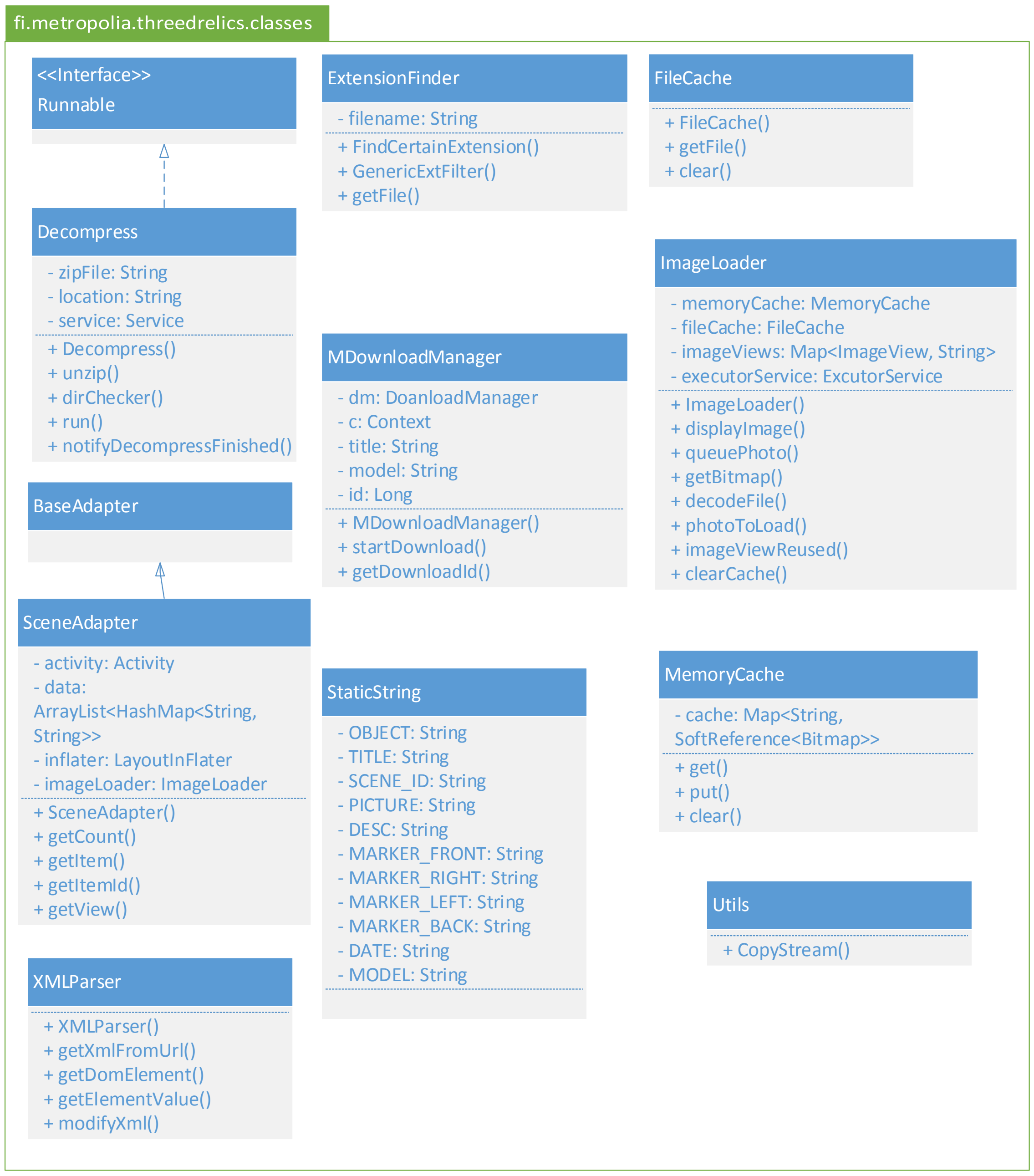


Figure 17. Class diagram part 3.

The package *fi.metropolia.threedrelics.classes is* demonstrated in Figure 17, which contains the supporter classes for the packages mentioned earlier.

4.4 Implementation of the main Android component

Just as a mobile device is designed to be used anywhere at any time, so is also 3D Relics. To enable the offline capability of AR functionality, the application has to go through a series of operations, as demonstrated in Figure 18.

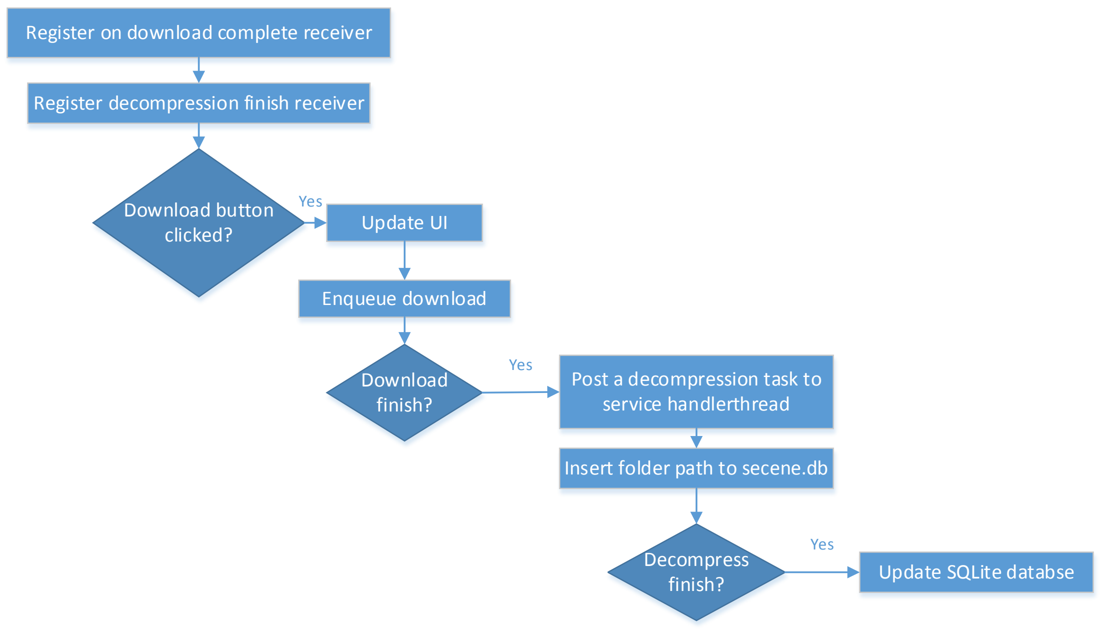

Register on download complete receiver
Register decompression finish receiver
Download button clicked?
Yes
Update UI
Enqueue download
Download finish?
Yes
Post a decompression task to service handlerthread
Insert folder path to secene.db
Decompress finish?
Yes
Update SQLite databse

Figure 18. Process of enabling offline mode.

4.4.1 SQLite database

SQLite is an open source database management system which is embedded in Android. SQLite supports standard relational SQL syntax, transactions and prepared statements [26]. In this project, SQLite database is used to keep references to the downloaded 3D model files in Android file system.

The package *fi.metropolia.threedrelics.db* contains two classes which are responsible for the communication between the application and the SQLite database. *DbEntry* class consists of the column names, create statement and delete statement as static strings. The *DbHelper*, as an inheritance of Android *SQLiteOpenHelper*, enables the application to conduct the CRUD operation to the scenes.db. Table 1 illustrates the structure of scene.db.

_id	scene_id	title	path	date	download_id

Table 1. Structure of scenes.db

4.4.2 Request and parse data

As Figure 11 demonstrates, the mobile application needs to request the web API for the scenes’ data. When the application starts, it will first send HTTP requests to server, and wait for the response in order to proceed with next task, which is parsing the data. However, if the internet connect is slow, this process will take up to several seconds and during this time user will not be able to interact with the application, which as result will bring a bad user experience. Thus an Android asynchronous task is utilized to handle this process in the background of the application. The *AsyncTask* class enables proper and easy use of the UI thread, which allows application to perform background operations and publish the results on the UI thread [27].

First, the asynchronous task sends a HTTP requests the web API and an XML formatted data is returned in response, as showed in listing 1.

```
<root>
  <object>
    <title></title>
    <scene_id></scene_id>
    <picture>http://54.247.2.103/page/descPics/watermill.jpg</picture>
    <desc>....</desc>
    <marker_front>http://54.247.2.103/page/markers/front.jpg</marker_front>
    <marker_back>http://54.247.2.103/page/markers/back.jpg</marker_back>
    <marker_left>http://54.247.2.103/page/markers/left.jpg</marker_left>
    <marker_right>http://54.247.2.103/page/markers/right.jpg</marker_right>
    <date></date>
    <model>http://54.247.2.103/page/objects/mill.zip</model>
  </object>
</root>
```

Listing 1. Sample XML data returned from the server.

Once the application receives the response, it will store the HTTP message body into a *HttpEntity* object. Then the *XMLParser* object starts to parse the data in the background. If the data is parsed successfully, the application will create a scenes.db in the android local SQLite database, and insert all scenes to table if the application is running for the first time, or only insert the newly added scene. Figure 19 demonstrates the details of the flow from start application to update the UI.

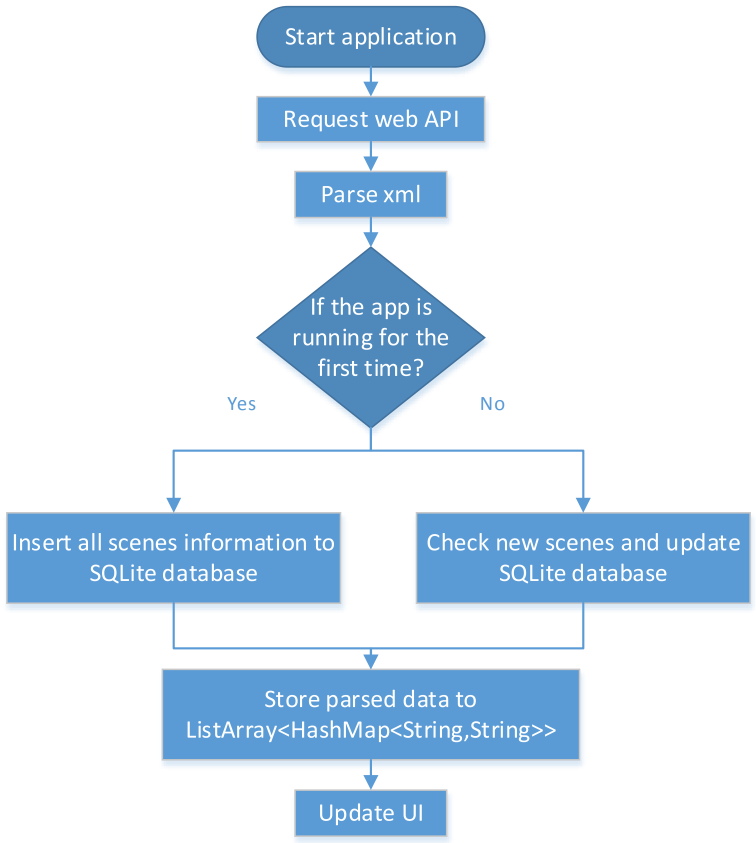

Start application
Request web API
Parse xml
If the app is running for the first time?
Yes
No
Insert all scenes information to SQLite database
Check new scenes and update SQLite database
Store parsed data to ListArray<HashMap<String,String>>
Update UI

Figure 19. XML data handling in the background.

When the whole process of updating UI is finished, scenes.db has the data shown as in Table 2, where the “scene_id” represents the unique id of each 3D scene, the “path” represents the absolute path of the folder in Android file system that contains this specific scene’s model file, texture file and markers’ files. As the user has not launch any model downloading operation, the fields “path”, “date” and “download_id” remains null.

_id	scene_id	title	path	date	download_id
1	23	mill	null	null	null
2	34	church	null	null	null

Table 2. scenes.db

4.4.3 Load and cache online image

In this application, there are several occasions that require loading online images. It is time consuming and waste of bandwidth to set up an HTTP connection and stream the file to mobile device every time the images are needed. So the application implements cache to solve this problem.

There are two types of cache in this application, the memory cache and file cache. The memory cache will store the image as a key value pair to a *Map<String, SoftReference<Bitmap>>* object, where the key is image URL and value is a reference to a *Bitmap* object. The soft reference to the bitmap will be cleared at the discretion of the garbage collector in response to memory demand [28]. Meanwhile, the image will be stored to the device's file system as a file cache.

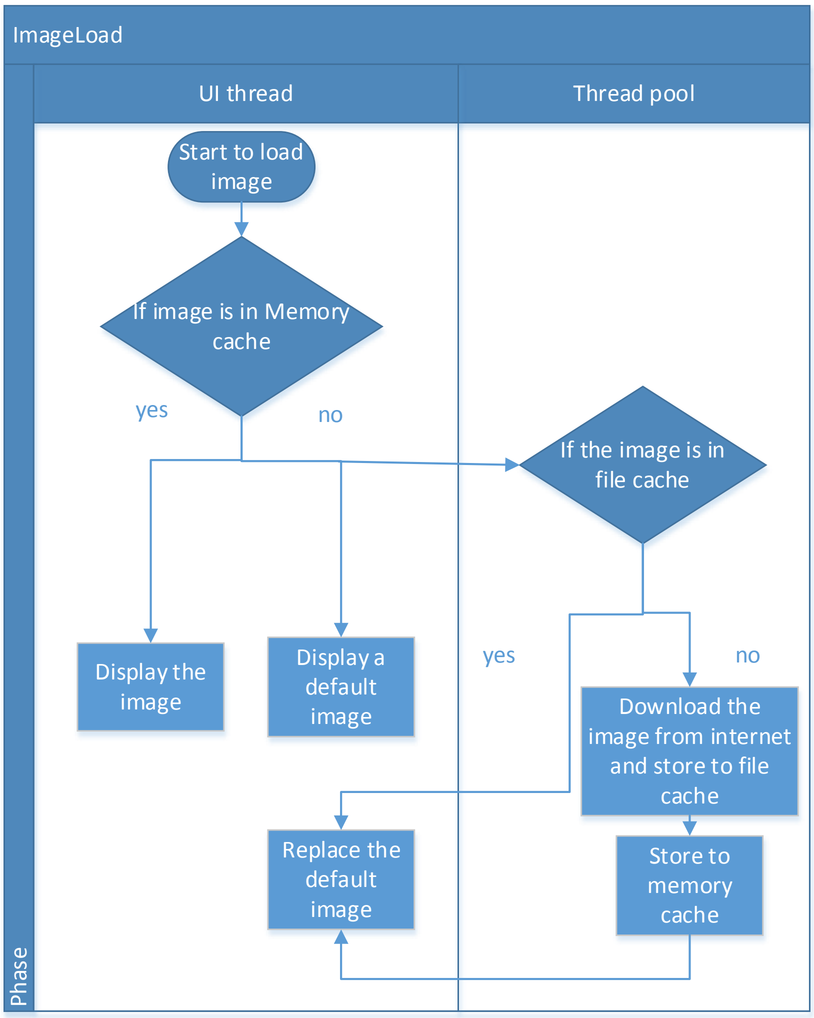

ImageLoad
UI thread
Thread pool
Start to load image
If image is in Memory cache
yes
no
If the image is in file cache
Display the image
Display a default image
yes
no
Download the image from internet and store to file cache
Replace the default image
Store to memory cache
Phase

Figure 20. Display an online image.

Figure 20 illustrates how the system will handle the request of displaying an online image. A substitute image will be displayed when the application finds no memory cache for that image, later on the substitute image will be replaced by the actual image. Usually this flow can be accomplished instantly.

4.4.4 Download manager

There are several ways to implement the download functionalities, however the Android *DownloadManager* class would be the ideal solution in this case, and it was enabled after Android API level 9. The download manager is a system service that handles long-running HTTP download [29].

As listed in Listing 1, a “model” element is returned for each scene to provice a link to the 3D model zipped file on the server. Once the user chooses to download that model, an *DownloadManager.Request* object with the model URL as a parameter will be queued to the download manager. Also a unique id for this download will be returned and then stored to the SQLite database. The reason to store the download id is to associate this download task with its scene. When the download has been finished, the *DownloadManager* will broadcast an *Intent* with download id as its extra information, by querying this download id in database, the application will acknowledge which scene has been finished downloading and proceed the next task related to this scene. This methodology enables multiple downloads concurrently, bringing a smooth user experience.

4.4.5 Event complete receiver

As the application is designed to be capable of multitasking, a mechanism is required to notify the application when the task is accomplished in each thread. Therefore, the broadcast receiver was introduced to this application. A broadcast receiver is an Android component which allows the application to register for system or application event, enabling a cross-process communication.

There are three types of broadcast receivers registered within the application. Two custom broadcast receivers are registered statically in android manifest file, with the purpose to receive the download complete event and decompress complete event respectively in the scope of the whole application. Once the *DownloadCompleteReceiver* receives the event, it will post a decompression task to *DeompressService*. Later when decompression is finished, the *DecompressCompleteReceiver* will be notified and the scene information will be updated in database accordingly, as demonstrated in Figure 11. The third broadcast receiver is registered dynamically in *ShowSingleSceneActivity*. Once user quits this activity, this receiver is unregistered simultaneously. The concept of third broadcast receiver lies in the fact that user may stay in the *ShowSingleSceneActivity* during the whole decompression process, and thus it is necessary to notify the foreground to activate the “launch” button once background running service finishes the decompression task, therefore user can immediately launch the 3D model.

4.4.6 Background service

Since the application was designed to enable multiple download tasks at the same time, a mechanism is required to handling the decompression job once the zip files are being downloaded. This mechanism should also be capable of updating the “path” in scenes.db to the current location of the unzipped files, thus a service is utilized to suit the purpose. Meanwhile, a handler is implemented to enqueue the decompress job.

However, as clarified in the Android official document, a service runs in the main thread of its hosting process, and as a result the handler generated within the service will also work in the main thread. For this reason, a worker thread is spawned within the service to avoid blocking main thread from being blocked by heavy tasks. This worker thread is an instance of *HandlerThread* class which inherits all the features from Thread class.

The service is triggered once a zip file is finished downloading, as demonstrated in Figure 11. Along with the absolute path of the downloaded zip file, the desired destination for storing the unzipped files will be sent to the decompress object as parameters. In android handler documentation, handler is defined that only Message or Runnable objects can be sent to the associated message queue. Thus, in order to send Decompress object to the message queue, the Runnable interface is implemented by the Decompress class. The decompress object is post to the message queue anonymously, because there is no way for the application to estimate the time that the system takes to decompress the zip file and thus there is no point to keep a reference to this decompress object.

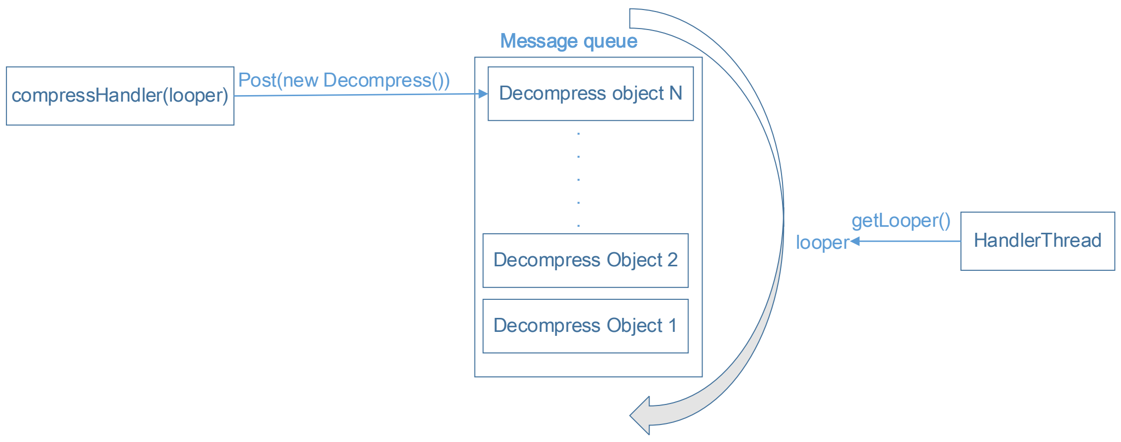


Figure 21. Work flow of decompress service.

To notify the application when the decompress process has finished, a *notifyDecompressFinished()* function is implemented in Decompress class, and it is been called after *unzip()*. In the *notifyDecompressFinished()*, an intent with the destination of unzipped files folder is sent as a broadcast. When application receives this intent, the scene's information will be updated in scenes.db, Figure 21 shows the updated columns after the download and decompression have completed, where the date is obtained from the XML data for that scene returned in HTTP response.

Start from Android API level 8, the function *getExternalFilesDir()* is available to the Context class. By calling this function, a private folder to the application itself will be returned. And the content in the folder will be removed once the application is uninstalled, in prder to release the memory on mobile phone.

path	date	download_id
/mnt/sdcard/Android/data/fi.metropolia.threedrelics/3dModelsUnZipped/23	02-02-2013	3245

Table 3. Updated contents in scenes.db

Eventually, after a round of processes and inner communications behind the scene, the folder which is named after scene's id will reside in the application's private folder, which contains all the files related to the scene, including front.jpg, back.jpg, left.jpg, back.jpg, model's .OBJ file, material library file and texture file(s).

4.5 Implementation of AR features

4.5.1 Content creation

To unleash the creativity of the content creators, Metaio SDK offers AR content supports in three areas: image, movie and 3D animation. In this project, all the AR contents were 3D models.

There are three supported model formats in Metaio SDK, which are OBJ for static objects, MD3 and FBX for animation objects [30]. To create model files for Metaio SDK, 3D computer graphics software is needed. The major 3D computer graphics software on the market now, such as Blender and 3ds Max, offer exporting choices to the above formats. Each file format has different purposes to serve, illustrated in Table 4, therefore the content creator should consider the upsides and downsides of each format and make a selection based on the content of their 3D models.

	advantages	disadvantages
MD2	1. Supports animation, but vertex animation only . 2. Very efficient for devices with weak processor. 3. File size is small.	1. Only one texture file is supported. 2. Comparatively old format. 3. No vertex color.
OBJ	1. Supports multiple meshes with different textures and materials. 2. Meshes can be assigned to different materials to their polygons.	1. Only for static meshes.
FBX	1. Contains almost everything from 3D model data to animation or even movie data.	1. Metaio SDK only supports subset of its numerous features. 2. An additional conversion step is need to strip all unwanted items from the file and turn it into a fast runtime format. 3. Larger file size.

Table 4. Advantages and disadvantages of Metaio supported file formats.

In this project, all the 3D models provided by 3D animation students are static ones, thus the OBJ file format was recommended as an exporting choice. When successfully exported, each model should have three type of files, the OBJ file, the material library files and the texture files, where the material library files contain one or more material definitions each includes the color, texture and reflection map of individual materials [31].

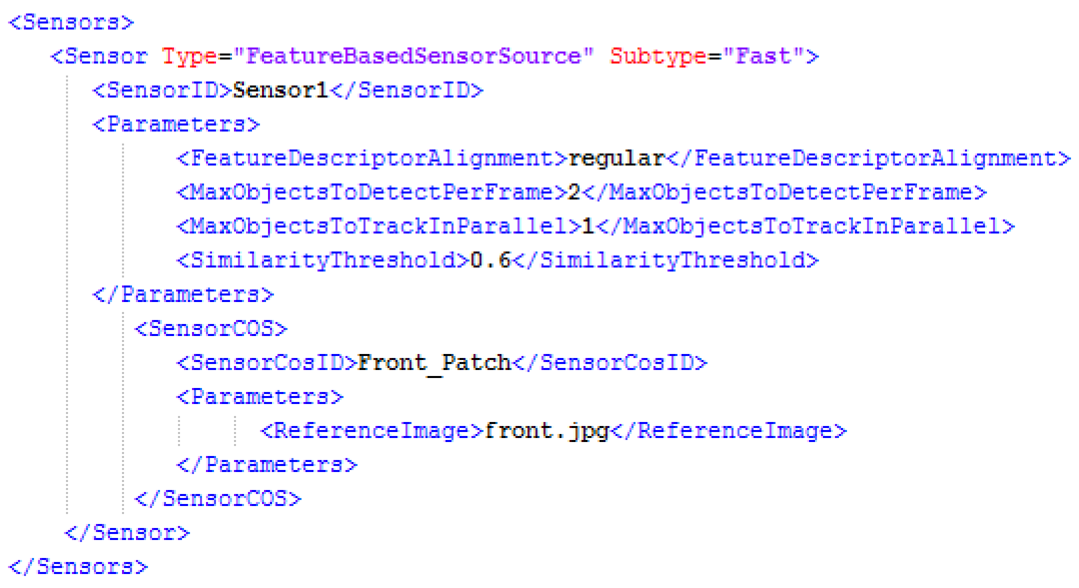
4.5.2 Setup configuration file

Currently, Metaio SDK provide several optical tracking technologies, including ID marker tracking, picture marker tracking, Latitude/Longitude/Altitude(LLA) marker tracking, markerless tracking, markerless 3D tracking and Quick Response(QR) code reader. In this project, markerless tracking is used to suit the application purpose [32].

There are two kinds of markerless tracking offered in Metaio SDK, namely fast and robust methods. The fast method is applicably under most of the cases, it runs fluently on smart phones and it is very stable on moderate textured targets [33]. This kind of sensor tracks planar objects by matching their intensities. It works at higher frame rate, and it can be sensitive to occlusions, secularities and light changes. While a robust method fits well with the highly textured target, however in order to obtain a satisfying result, more time is required for the user to hold the device in front of the tracked target. Due to the fact that the models will be rendered on the mobile phone screen, user will not be able to notice the details on model’s texture. Thus it is recommended to reduce the complexity of the model’s texture thereby to increase the rendering speed. Besides, to bring a better interact experience, the model should be render at high frame rate. Based on the reasons stated, the “fast” method was chosen for this project.

The *FeatureDescriptionAlignment* has four feature descriptor types to choose from, which are “regular”, “upright”, “gravity” and “rectified”. The “regular” feature descriptor is default for this parameter; it fits well in most of the cases. The “upright” feature descriptor assumes that during the tracking process, the camera is not rotating with the respect to the optical axis, thus not selected by the project. The “gravity” feature descriptor can only be used with the device that has inertial sensors that measures gravity. It is used to localize static objects that are either on a vertical surface or close to it. The orientation of the features will then align with gravity. Like the “gravity” descriptor, the “rectified” feature descriptor requires device with inertial sensors. It is used only for planar objects on a horizontal surface, thus is not chosen in this project. Hence, as explained above, only “regular” and “gravity” suit the purpose of this project. To examine which descriptor has the better tracking result, both were tested in the application. Both descriptors worked well with no significant difference in tracking results. Considering some devices may have unstable or no inertial sensors, the project finally chose “regular” as value for *FeatureDescriptionAlignment.*

The *SimilarityThreshold* is a threshold value used to decide whether the tracking was failed or succeed. The Metaio’s tracking quality is represented as a float value ranging from -1 to 1, where 1 stands for the best tracking quality. After series of testing under various lighting condition utilizing different marker targets were carried out. The lower the threshold value was, the faster the application detects the marker. However, as a result, many markers were not interpreted correctly. When the *SimilarityThreshold* was set to 0.6, the application detected all the marker correctly with less than one second, thus 0.6 was chosen as the threshold value for this project.



```
<Sensors>
   <Sensor Type="FeatureBasedSensorSource" Subtype="Fast">
      <SensorID>Sensor1</SensorID>
      <Parameters>
            <FeatureDescriptorAlignment>regular</FeatureDescriptorAlignment>
            <MaxObjectsToDetectPerFrame>2</MaxObjectsToDetectPerFrame>
            <MaxObjectsToTrackInParallel>1</MaxObjectsToTrackInParallel>
            <SimilarityThreshold>0.6</SimilarityThreshold>
      </Parameters>
         <SensorCOS>
            <SensorCosID>Front_Patch</SensorCosID>
            <Parameters>
                  <ReferenceImage>front.jpg</ReferenceImage>
            </Parameters>
         </SensorCOS>
    </Sensor>
</Sensors>
```

Listing 2. Configuring sensors section in markerless tracking configuration file.

For showing the front view of the 3D model, a *SensorCOS* with “Front_Patch” as its *SensorCosID* and front.jpg as its *ReferenceImage* were defined. The reference image tells the application which object to track. And based on detected image, as the red rectangular area showed in Figure 22, a virtual coordinate is drawn on the center of that area. In the later part of the configuration file, a connection will be established between coordinate system (COS) entities and *SensorCOS*.

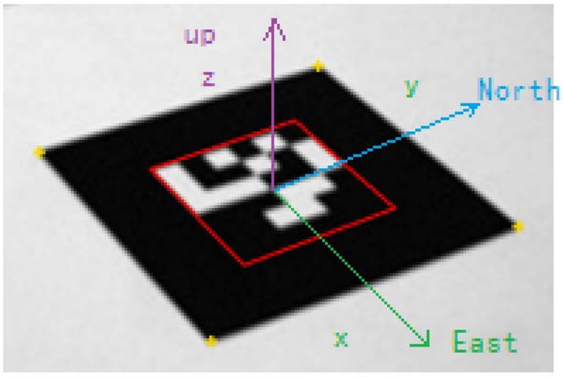


Figure 22. Metaio coordinate system and marker detecting

The *SensorCOS* represents the pose of the tracked object relative to the sensor, and the COS is the pose that will be used when augmenting contents. To correlate the *SensorCos* to *COS*, they must share the same *SensorCosID.*

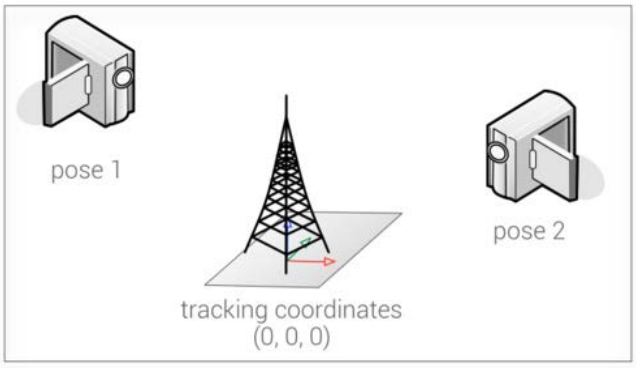
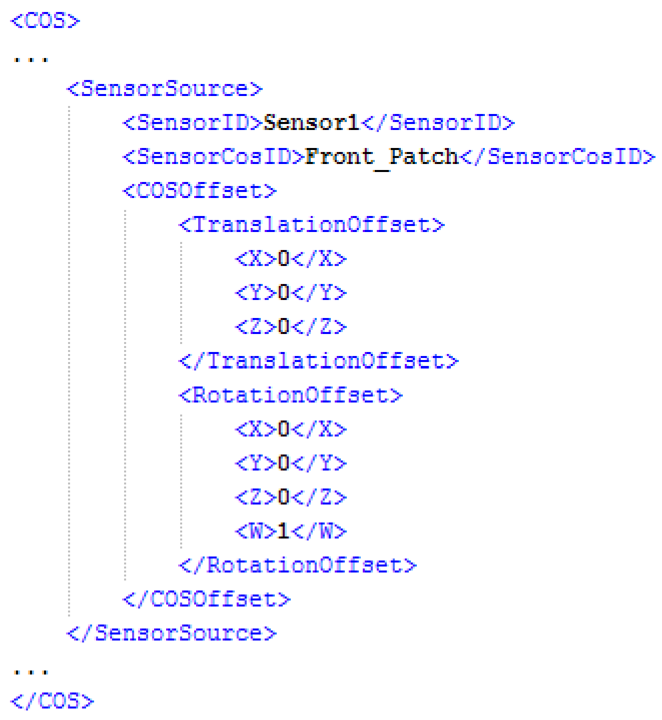


Figure 23. Poses [34]

When the pose's is been detected, as shown in Figure 23, the application will iterate through the *COS* tags until it finds the *COS* with the same *Sensor*CosID as the detected pose. Then according to the *COSOffset,* a rigid transformation is set to the coordinate. In this project, a rotation transformation is obligatory for transforming the model from front view to left view, right view and back view respectively. Those rotations were set to the augmented object programmatically in the Android code. So in the configuration file, no rotation was set.



```
<COS>
...
    <SensorSource>
        <SensorID>Sensor1</SensorID>
        <SensorCosID>Front_Patch</SensorCosID>
        <COSOffset>
            <TranslationOffset>
                <X>0</X>
                <Y>0</Y>
                <Z>0</Z>
            </TranslationOffset>
            <RotationOffset>
                <X>0</X>
                <Y>0</Y>
                <Z>0</Z>
                <W>1</W>
            </RotationOffset>
        </COSOffset>
    </SensorSource>
...
</COS>
```

Listing 3. Correlate COS to SensorCOS.

The X,Y,Z,W surrounded by <RotationOffset></RotationOffset> is a notation of quaternions, where X, Y, Z are the unit vector representation of the axis multiplied by the sine(rotation angle/2) while W represents cosine(rotation angle/2), as 0 rotation angle was set, the X, Y, Z, W are calculated as follow:

X = [1 0 0] sin(0/2) = [0 0 0]
Z = [0 1 0] sin(0/2) = [0 0 0]
Y = [0 0 1] sin(0/2) = [0 0 0]
W = cosine(0/2) = 1.

Thus the value assigned to X, Y, Z, W are 0 0 0 1 respectively.

4.5.3 AR as an Android activity

To experience augmented contents in real time, Metaio SDK proposed a methodology that is to overlay views on top of each other and only renders the augmentation in *MetaioSurfaceView.*

In this project, three views are added when the activity is been created. To capture the real world scene, device camera will be activated when the activity's *onStart()* function is been called, and a camera view will be laid at the bottom among the other views. Then a *MetaioSurfaceView*, which in essence is an inheritance of *android.opengl.GLSurfaceView*, will be initialized and overlaid onto the camera view. At last, a GUI view will be displayed on top of MetaioSurfaceView.

To render the 3D model concurrently with the movement of the device, the MetaioSurfaceView implemented *MetaioSurfaceView.Callbacks* which is a Metaio encapsulation of *OpenGLSurfaceView.Renderer* interface. The onSurfaceCreated() will be called to initialize a renderer on a rendering thread for this view. Once the IMetaioSDKAndroid has updated tracking and is ready to render, the onDrawFrame() callback is called constantly in a loop to render the model in renderer thread. Figure 24 introduces the flow of presenting augmentation.

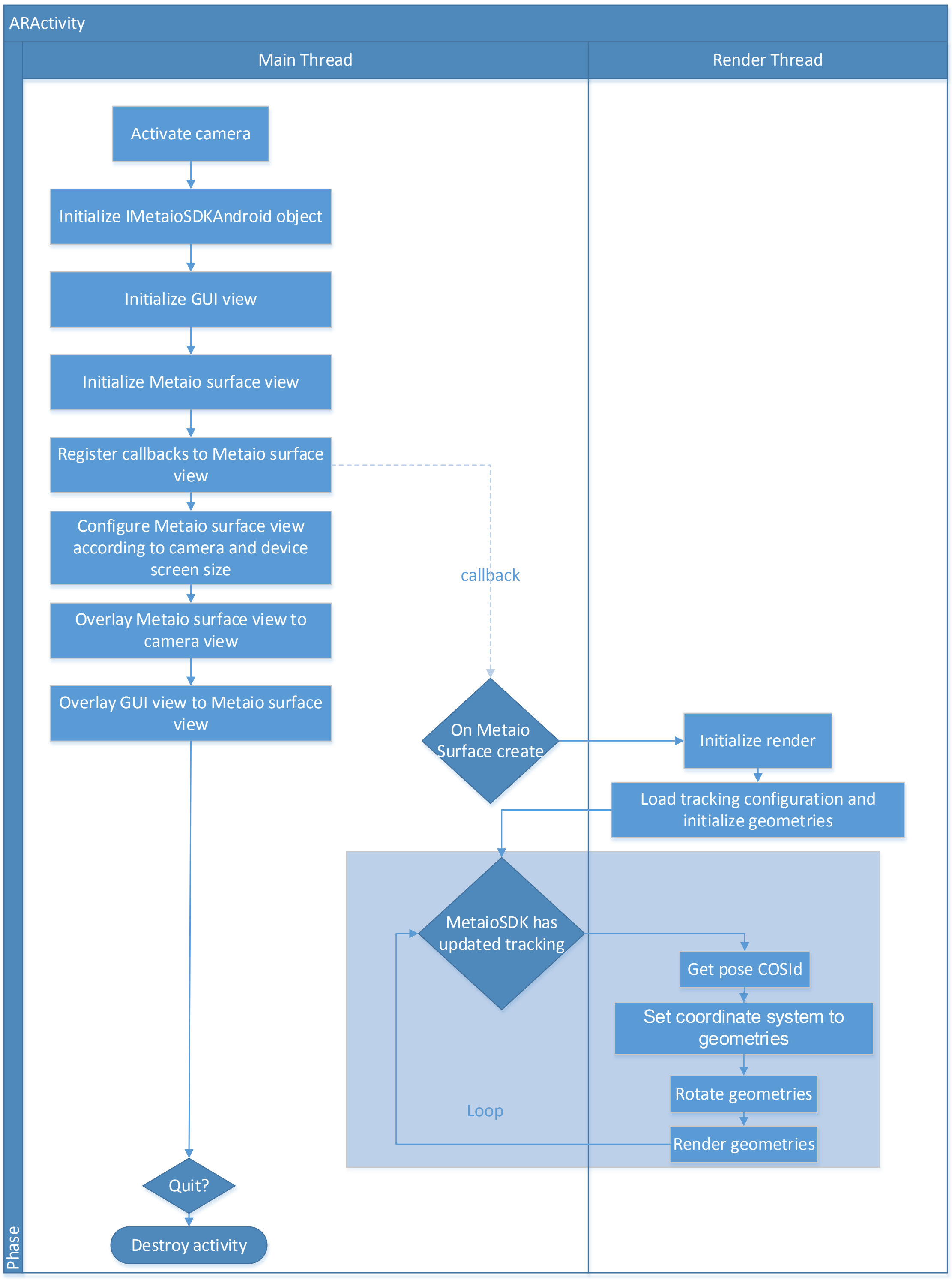
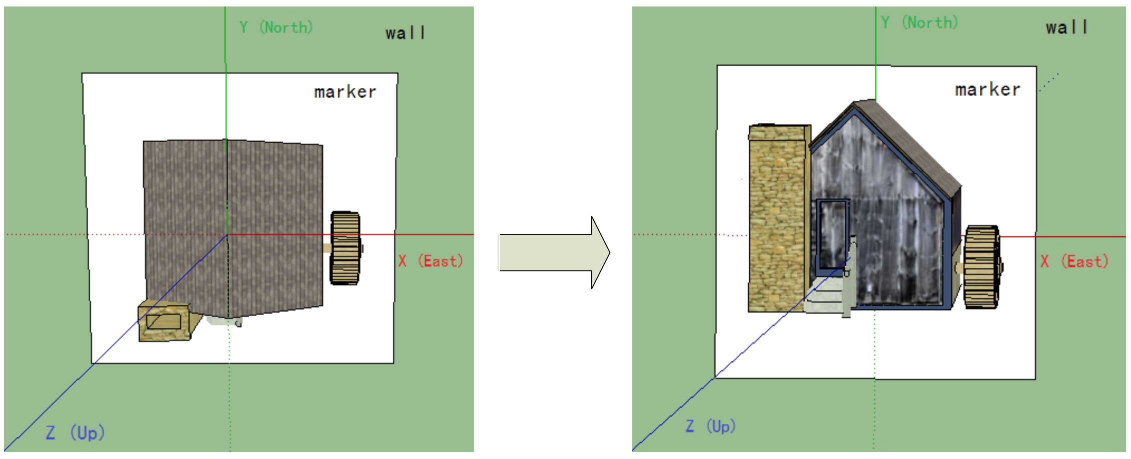

ARActivity
Main Thread
Render Thread
Phase
Activate camera
Initialize IMetaioSDKAndroid object
Initialize GUI view
Initialize Metaio surface view
Register callbacks to Metaio surface view
Configure Metaio surface view according to camera and device screen size
Overlay Metaio surface view to camera view
Overlay GUI view to Metaio surface view
callback
On Metaio Surface create
Initialize render
Load tracking configuration and initialize geometries
MetaioSDK has updated tracking
Get pose COSId
Set coordinate system to geometries
Rotate geometries
Render geometries
Loop
Quit?
Destroy activity

Figure 24. Lifecycle of ARActivity.

Considering the printed marker is supposed to be attached to a vertical surface, the tourists will see a top view of the model when pointing the device to the marker, and it is not easy to observe the back view. Therefore, a rotation was set to transform the model from top view to front view based on the virtual coordinate system of detected marker as demonstrated in Figure 25. Furthermore, to have a comprehensive observation of all side of the model, four markers were introduced to present front view, back view, left view and right view respectively.



Top view Front view

Figure 25. Transform from top view to front view.

As soon as a marker is detected, the marker's correlated coordinate system is set to the model, a certain rotation is added to the model programmatically after model has been loaded.

5 Testing

The testing of the mobile application consists of preliminary tests and final tests. The preliminary tests were carried out in different phases of the development process, either with the teachers, students or developers. That way, many subtle bugs were eliminated before the final tests.

5.1 Controlled environment testing

This test was conducted inside a classroom following the testing plan in Table 5. Four students were involved in this testing, among whom two had no IT background, thus creating some diversity in the tester group.

The device used in testing sessions was Samsung Nexus S with Android version 4.1.2. There was three 3D scenes obtained beforehand from the project's 3D model support Mr. Ale Torkkel, all of which were already exported to .obj file format correctly as instructed in Appendix 1. For each scene, four markers representing front, back, right, left were attached to the wall to be read by the device.

As Table 5 indicates, the results of the testing were almost positive, except in Test No. 5 one tester found out that when he pressed the back button immediately after launch button was clicked, the application stopped suddenly. The cause of the bug was identified later to be a null pointer exception, which after a series of debugging and discussions with Metaio developers, turned out to be a Metaio design flaw, because the OpenGL renderer is not unregistered properly by IMetaioAndroidSDK object after the OpenGL drawing context is destroyed. The Metaio developers are expecting to fix this bug in their next release.

Test No.	Test Description	Steps	Result
1	Install mobile client.	1. Open browser on the device and type the URL of the application. (http://users.metropolia.fi/~huiwend/3DRelics/threedrelics.apk) 2. Carry out installation	Successful
2	Launch the mobile client.	1. Launch the application.	Successful
3	Download 3D Scenes.	1. Click on one of the available scenes and download the scene. 2. Quit the current scene view and choose another available scene to download. 3. Wait for a while until you see the launch button is activated in both scenes.	Successful
4	Update the existing 3D scene.	1. Wait for the staff to upload a new version of each scene through web client. 2. Return to the available scenes view. 3. Click on the scene you have downloaded in previous test. 4. Check if the update button is activate for this scene. Click update button to download the newer version.	Successful
5	Launch the scene and interact with it under no internet connection.	1. Return to the available scenes view. 2. Disable the internet connection. 3. Click on one scene you have downloaded and launch the scene. 4. Find the related four markers of the scene, move the camera around the marker to view the scene in different angels. 5. Quit the application	Partly successful

Table 5. Testing plan and results.

During the test, some feedback was received from the testers concerning the user experience, such as to add a short step-by-step instruction on how to use the application at the beginning and add notifications when the model is been downloading and decompressing. Because during this process both “download” and “launch” buttons were inactive and the users were not sure what to do next. These features were later implemented in the final version of the application.

5.2 Field testing

This test took place at Vanhankaupungintie 9, Helsinki, where the oldest church of Helsinki used to reside. The church was built in 1550, when Helsinki was moved to Vironniemi in the 1640s. The church was then left to the local hospital, but it soon fell into decay and was abandoned. Eventually, all that left was the stone base [34].

In this test, “3D Relics” was installed into two devices, which were Samsung Galaxy S3 and Samsung Galaxy Tab 2. The church scene was then downloaded for later use.

As Figure 26 demonstrates, a marker was attached to a pole that was planted into the ground. Both devices detected the markers and rendered the scene successfully under non-internet environment. Figure 27 shows the screenshot of the testing results.

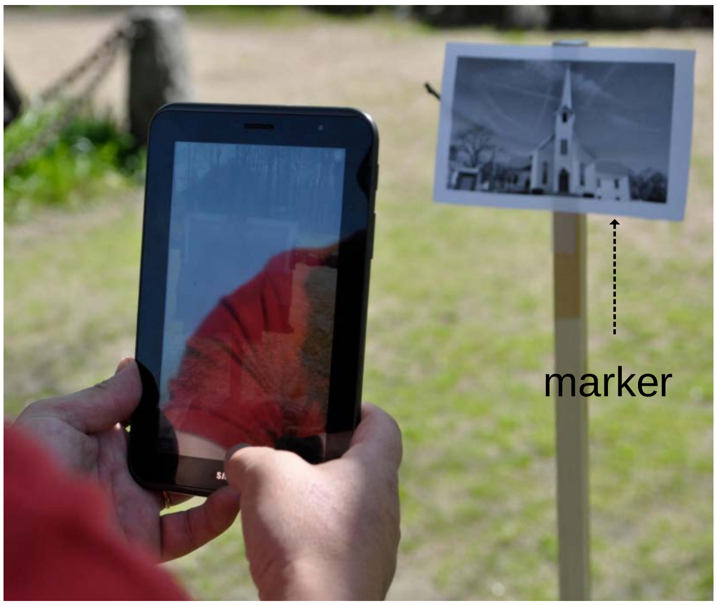


Figure 26. Detecting the marker. Photo taken by Huiwen Duan.

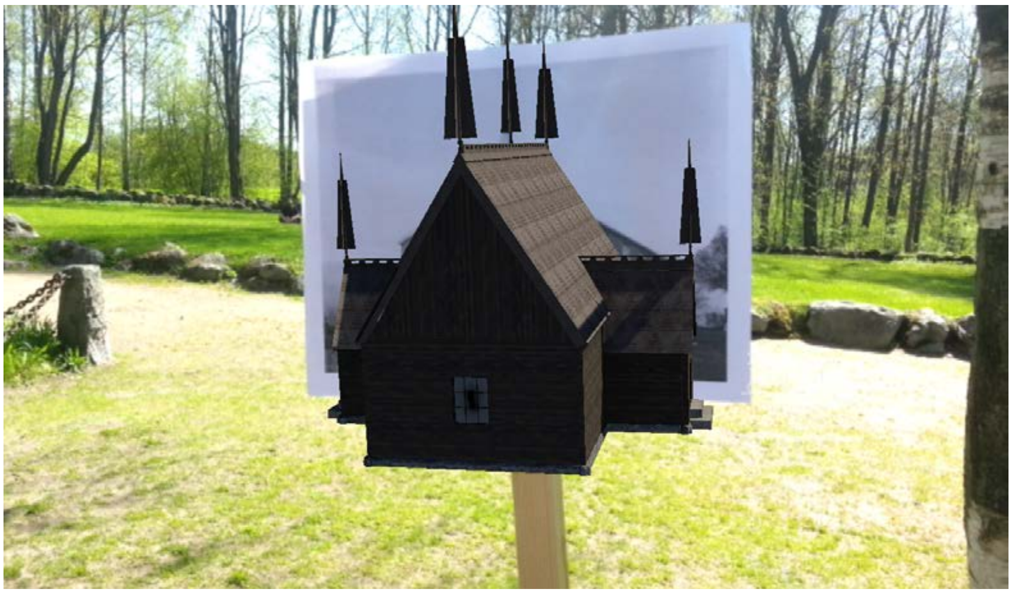


Figure 27. Screenshot from Samsung Galaxy S3.

However, it is worth mentioning that there were some factors affecting the tracking and rendering results. For example because of the wind it was difficult to keep the marker as flat as it should be. Also, when the sunlight goes through the marker, the application took longer time to detect the marker due to less image contrast on the marker.

To enhance the tracking results, it is recommended to attach the marker to a cardboard and use colour image as marker.

6 Results and discussion

At the completion of this project, an integrated system has been established; two user groups were able to perform the desired functionalities through web application and mobile application. By developing 3D Relics, it was proved that the 3D models can be presented as an augmentation of the real world. The tracking and rendering results were satisfying. Apart from most of the AR application on the market, 3D Relics fulfilled the prime aim of this research project that is to display the AR contents under non-internet enabled situation.

The project followed the rapid application development methodology. The project started with the development of preliminary data models and business process models using structured techniques. In the next stage, requirements were verified using prototyping, eventually to refine the data and process models. These stages were repeated iteratively. By apply this methods, the mobile application structure has been improved several times when the developer found better solutions of optimizing the performance, and also the structure on the web client has been modified to provide the desired data for mobile client.

The whole development process has consolidated the developer's Android programming skills and brought the developer to a new level in understanding Android design pattern. The developer also come to know that develop an application for mobile platform is very different from other platforms, as the limited computing power and the screen size has put many constrains on the application. Thus to effectively pass information, the application should be taken care of at the design phase. Furthermore, it was an opportunity to explore a number of other fields, including computer graphics rendering flow, image processing, 3D modeling and mathematics. Most importantly, the developer has gained a more thorough comprehension in software development.

7 Future development

The potential of AR on mobile platforms remains vastly untapped, and more new features can be brought to 3D Relics.

As mentioned in an earlier chapter, there are multiple tracking technologies for AR applications to adopt, one of which is the location based tracking. By utilizing the device’s embedded sensors, the system can overlay the virtual content on a specific latitude, longitude and altitude. This tracking technology was implemented at the beginning phase of this research project, but as the current sensor readings on Android phones are very unstable and lack of accuracy, this approach was abandoned. Nonetheless, with the rapid growth of mobile hardware performance, this method can be reactive in the near future.

The AR related functionalities can be explored more, such as gesture interaction with the virtual content. When user click on certain part of the virtual model, information related to that part could be prompted. In fact, this functionality was already programmed in the application. However, due to lack of detailed information to the corresponded model, this feature was disabled currently.

Another improvement can be adding the “sharing” functionality to the application. As social network is thriving nowadays, the tourists may want to share the information with their friends when they “check in” the historical spot. To implement this feature, the application can easily connect to the user’s SNS account, such as Facebook and Twitter, by utilizing the public API provided by the SNS. On the other hand, this feature could also be turned out a way to advertise these historical sites.

8 Conclusion

The way people obtain information has changed dramatically in recent years. Instead of passively receiving the information, they expect to interact with the information source. To meet these new behavioural transformations, technologies from different fields were brought together in a form of augmented reality applications to address these needs. Augmented Reality is a technology that enables the interaction between the user and the machine in real time and offers the user a new vision of the surrounding world. Though AR has been utilized in diverse fields for decades, it was never close to individual customers until last few years. With the rapid growth of smartphone users, AR will become more accessible to a broader audience.

In this project, AR technology was used to present the tourists with a more in-depth tour of the historical sites. A system consisting of a web client and a mobile client was designed to fulfill this goal. One specific requested feature for the mobile client was that the tourists should be able to view the AR content without internet access. In other words, the mobile application should be capable of all the tracking and rendering of the AR contents locally. After months of efforts, the whole system was established, and it worked as desired.

In this thesis, a number of AR use cases were examined, the structure of the system has been described and a whole flow of developing a standalone AR mobile application have been presented. However, what AR can do in the future cannot be predicted。With the efforts of scientists all over the world, in the not-so-distant future a more splendid world with the help of AR technology can be expected.

References

1. Johnson L., Levine A., Smith R. The 2010 Horizon Report [online]. Texas, United States: The New Media Consortium; January 2010.
URL: http://wp.nmc.org/horizon2010/chapters/simple-augmented-reality/. Accessed 1 May 2013.

2. A new world of gaming: creating mixed reality [online]. United States: Orbotix; June 2011.
URL: http://www.gosphero.com/tag/augmented-virtuality/. Accessed 6 May 2013

3. Botella C, Quero S. Virtual reality and psychotherapy [online]. Amsterdam, Netherland; 2004.
URL: http://www.cybertherapy.info/cybertherapy/3_Botella.pdf. Accessed 3 May 2013.

4. Milgram P. Augmented reality: a class of display on the reality-virtuality continuum [serial online]. 1994;2351:12.
URL:http://etclab.mie.utoronto.ca/people/paul_dir/SPIE94/SPIE94.full.html. Accessed 3 May 2013.

5. Cassella D. What is augmented reality [online]. Designtechnica Corporation; November 2009.
URL: http://www.digitaltrends.com/mobile/what-is-augmented-reality-iphone-apps-games-flash-yelp-android-ar-software-and-more/. Accessed 2 May 2013.

6. Smart Grid augmented reality [computer programme]. Version 2. US: General Electric Company; 2012.

7. A helmet-mounted virtual environment display system [online].Ohio, United States: Wright-Patterson Air Force Base; December 1989.
URL: http://www.dtic.mil/dtic/tr/fulltext/u2/a203055.pdf. Accessed 2 May 2013.

8. Baron. Google Glass [online]. PixelDrunk; February, 2013.
URL: http://www.pixeldrunks.com/2013/02/27/google-glass/. Accessed 16 May 2013

9. Denis K, Bernhard R, Petter R, Alexander B, Reinhard B, Dieter S, Eigil S. Integrated Medical workflow for augmented reality applications [online].
URL:http://www.icg.tugraz.at/Members/denis/publication/kalkofen__integrated_workflow.pdf. Accessed 6 May 2013.

10. Maxim B. Image processing: skin detection [online]. CodeProject; July 2009
URL:http://www.codeproject.com/Articles/38176/Image-Processing-Skin-Detection-Some-Filters-and-E. Accessed 6 May 2013.

11. Zhou R, Junsong Yuan, Zhengyou Z. Robust hand gesture recognition based on finger earth mover's distance with a commodity depth camera [online].
URL:http://eeeweba.ntu.edu.sg/computervision/people/home/renzhou/Ren_Yuan_Zhang_MM11short.pdf. Accessed 6 May 2013.

12. Coordinate system [online]. Seattle, United States: University of Washington.

URL:http://www.hitl.washington.edu/artoolkit/documentation/cs.htm. Accessed 6 May 2013

13. Current augmented reality applications [online]. PBworks; May 2009. URL:http://augreality.pbworks.com/w/page/9469034/Current%20Applications%20of%20AR. Accessed 6 May 2013.

14. Macintyre B. Knowledge-based augmented reality for maintenance assistance [online]. New York, United States: Columbia Univeristy; May 2007. URL: http://monet.cs.columbia.edu/projects/karma/karma.html . Accessed 6 May 2013.

15. Kim S. An architecture of augmented broadcasting service for next generation smart TV [online]. Seoul, South Korea: IEEE International Symposium; June 2012. URL:http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=6264289&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D6264289. Accessed 1 May 2012.

16. Hines M. AR in movie making [online]. Trend Hunter; March 2011. URL:http://www.trendhunter.com/trends/the-witness. Accessed 1 May 2012.

17. Layar code example for developers [online]. SlideShare; June 2011. URL: http://www.slideshare.net/layarmobile/layar-code-examples-for-developers. Accessed 1 May 2013.

18. Metaio developer portal overview [online]. Metaio; December 2012. URL: http://dev.metaio.com/overview/. Accessed 1 May 2013.

19. Start with the augmented reality Android SDK [online]. Wikitude; November 2011. URL: http://www.wikitude.com/developer/get-started/android/. Accessed 1 May 2012.

20. Google: 850,000 Android devices activated every day [online]. Mashable; February 2012. URL:http://mashable.com/2012/02/27/android-daily-activations/. Accessed 1 May 2013.

21. Mobile-os-market-share-1q-2012 [online]. Solostream; April 2012. URL:http://androidheadlines.com/2012/12/malware-on-android-should-i-be-worried.html/mobile-os-market-share-1q-2012-2. Accessed 1 May 2013.

22. Android software development [online]. SlideShare; March 2013. URL: http://www.slideshare.net/quiver78tub/android-application-development-steps-to-the-building-blocks-17666828. Accessed 1 May 2013.

23. Setting up the Development Environment [online]. Metaio; May 2012. URL:http://dev.metaio.com/sdk/getting-started/android/setting-up-the-development-environment/. Accessed 1 May 2013.

24. Swipe views [online]. Anroid Developers; March 2012. URL: http://developer.android.com/design/patterns/swipe-views.html. Accessed 1 May 2013.

25. Android SQLite database and ContentProvider [online]. Vigella; January 2013.

URL: http://www.vogella.com/articles/AndroidSQLite/article.html. Accessed 1 May 2013.

26. AsyncTask [online]. Android Developers; August 2012.
URL: http://developer.android.com/reference/android/os/AsyncTask.html. Accessed 1 May 2013.

27. Class SoftReference<T> [online]. Oracle; 2011.
URL: http://docs.oracle.com/javase/6/docs/api/java/lang/ref/SoftReference.html. Accessed 1 May 2013.

28. DownloadManager [online]. Android Devloper; April 2012.
URL:http://developer.android.com/reference/android/app/DownloadManager.html. Accessed 8 May 2013.

29. Support model formats [online]. Metaio; May 2011.
URL:https://dev.metaio.com/content-creation/3d-animation/format/supported-model-formats/. Accessed 8 May 2013.

30. Alias/WaveFront material(.mtl) file format [online]. Wavefront; October 1995.
URL: http://www.fileformat.info/format/material/. Accessed 8 May 2013.

31. ID marker [online]. Metaio; July 2012.
URL:http://dev.metaio.com/sdk/tracking-configuration/optical-tracking-technologies/id-marker/. Accessed 8 May 2013.

32. Markerless tracking configuration [online]. Metaio; July 2012.
URL:http://dev.metaio.com/sdk/tracking-configuration/optical-tracking-technologies/markerless/. Accessed 8 May 2013.

33. Coordinate system [online]. Metaio; July 2012.
URL: http://dev.metaio.com/sdk/getting-started/coordinate-system/. Accessed 8 May 2013.

34. Helsingin ensimmäinen kirkko [online]. Art and design city Helsinki Oy; November 2005.
URL:http://www.arabianranta.fi/kulttuuri/helsingin_ensimmainen_kirkko/?show=1. Accessed 16 May 2013.

3D Scene content creation

Export the 3D file

The 3D models have to be exported within the modelling program one by one. The following example will be using 3ds max 2013.

1. Create an empty destination folder for the exported files.

2. Select the 3D model file and open it with 3ds max 2013. When the file is opened, click the upper left icon as indicate in Figure 1.

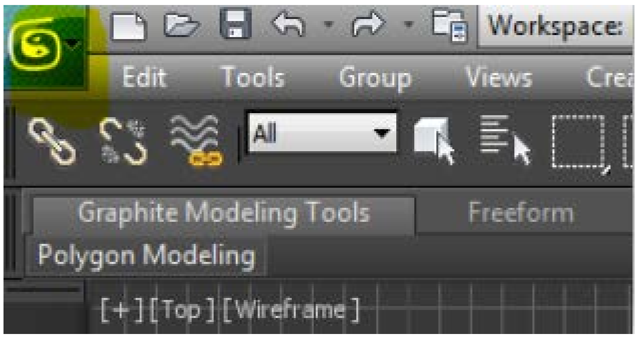
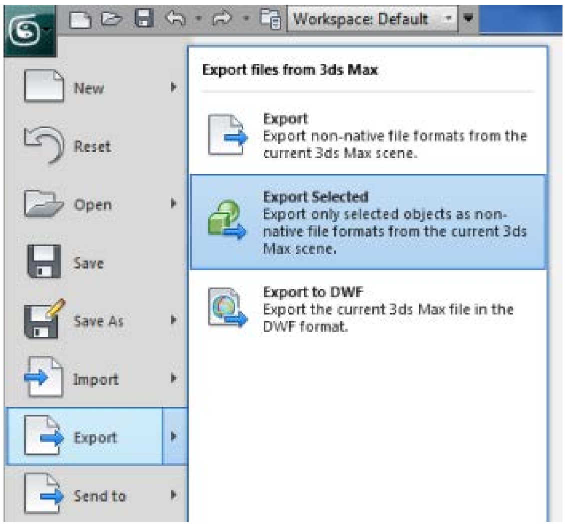


Figure 1. Screen shot of 3ds max control panel

3. Mouse over “Export” in the drop down menu, and select “Export Selected” in its submenu as Figure 2.



Export files from 3ds Max

Export
Export non-native file formats from the current 3ds Max scene.

Export Selected
Export only selected objects as non-native file formats from the current 3ds Max scene.

Export to DWF
Export the current 3ds Max file in the DWF format.

Figure 2. Export 3d file.

4. Give a name for the file to be exported, and save as type *.OBJ as indicated in Figure 3, and click “save”.

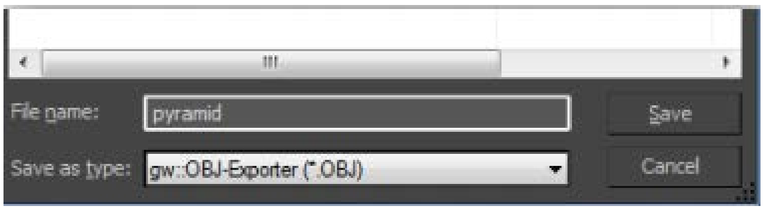
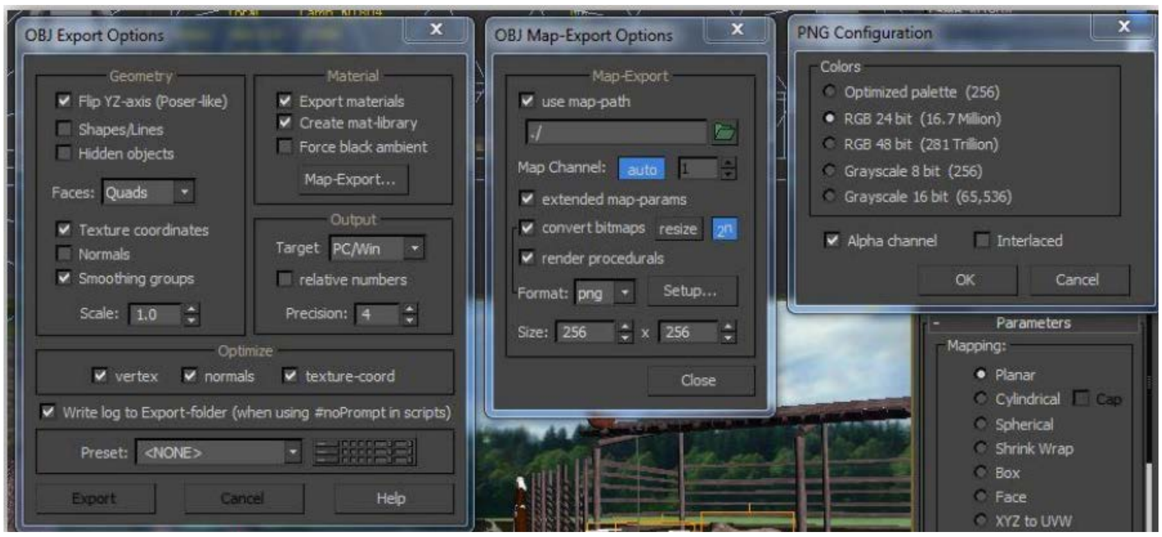

File name: pyramid
Save as type: gw::OBJ-Exporter (*.OBJ)
Save
Cancel

Figure 3. Export file type.

5. “OBJ Export Option” window (most left one in Figure 4) will appear after save is clicked, configure the settings as follow, and click “Map-Export...”. Set up the export options as the middle window in Figure 4 and click on “Setup...” in this window. Choose RGB 24bit for “PNG Configuration” window. After selecting the right settings, click “Export” in the “OBJ Export Option” window.


OBJ Export Options

Geometry
☑ Flip YZ-axis (Poser-like)
☐ Shapes/Lines
☐ Hidden objects
Faces: Quads
☑ Texture coordinates
☐ Normals
☑ Smoothing groups
Scale: 1.0

Material
☑ Export materials
☑ Create mat-library
☐ Force black ambient
Map-Export...

Output
Target: PC/Win
☐ relative numbers
Precision: 4

Optimize
☑ vertex ☑ normals ☑ texture-coord

☑ Write log to Export-folder (when using #noPrompt in scripts)
Preset: <NONE>
Export Cancel Help

OBJ Map-Export Options

Map-Export
☑ use map-path
./
Map Channel: auto 1
☑ extended map-params
☑ convert bitmaps resize 2ⁿ
☑ render procedurals
Format: png Setup...
Size: 256 x 256
Close

PNG Configuration

Colors
○ Optimized palette (256)
● RGB 24 bit (16.7 Million)
○ RGB 48 bit (281 Trillion)
○ Grayscale 8 bit (256)
○ Grayscale 16 bit (65,536)
☑ Alpha channel ☐ Interlaced
OK Cancel

Figure 4. Export configuration.

Prepare files for the scene and zip

After the 3D file has been exported successfully, some other files need to be added to the destination folder, at the last the folder should contain:
- front.jpg
- back.jpg
- right.jpg
- left.jpg

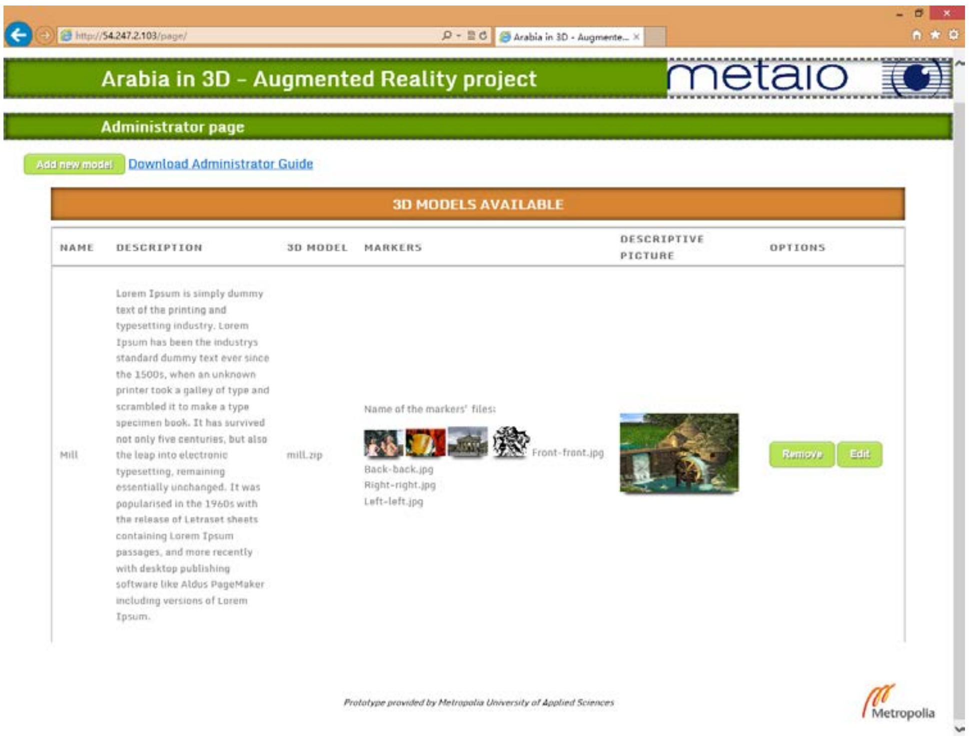
- material library file(.mtl)
- model file(.obj)
- texture file(.png)

After all these files are ready, compress the folder as a zip file. This zip file should be upload to server through web client.

Web client user manual

The web client of 3D Relics is used for deploying the scenes to server.

1. Open the web browser and enter 54.247.2.103/page/ to the address bar and the web client will open as Figure 1. You can either edit or delete the model by click on “edit” and “remove” button.



Arabia in 3D – Augmented Reality project

Administrator page

Download Administrator Guide

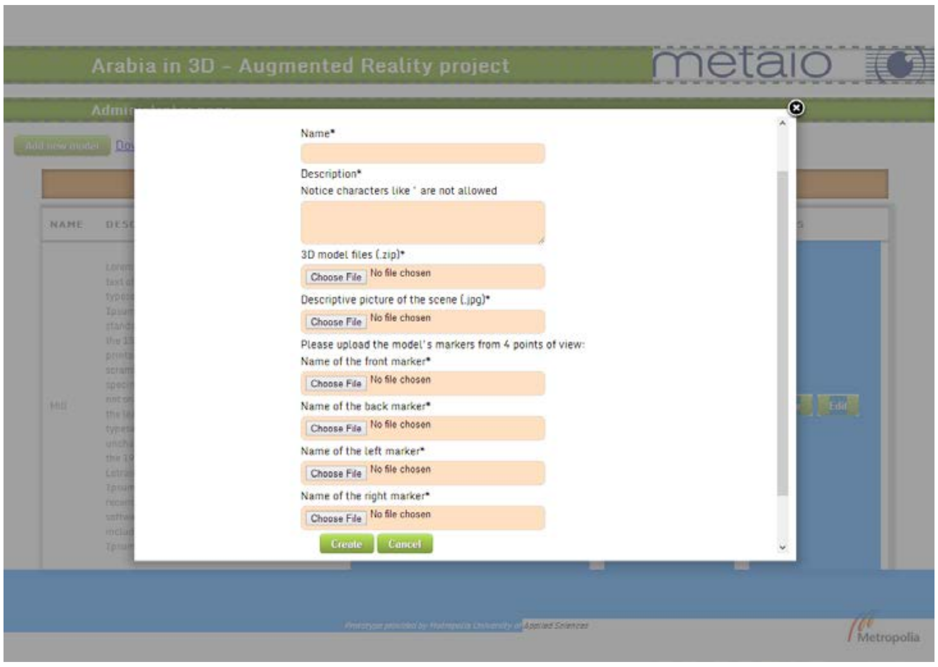
3D MODELS AVAILABLE

NAME	DESCRIPTION	3D MODEL	MARKERS	DESCRIPTIVE PICTURE	OPTIONS
Mill	Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industrys standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.	mill.zip	Name of the markers' files: Front-front.jpg Back-back.jpg Right-right.jpg Left-left.jpg		Remove Edit

Prototype provided by Metropolia University of Applied Sciences

Figure 1. Web client user interface.

2. Click on “Add new model” button the upper left corner, and a box in Figure 2 will prompt up. Type the corresponding information of this model and upload the zip file, as shown in Figure 3, containing the marker files and 3D model files



Name*

Description*
Notice characters like ' are not allowed

3D model files (.zip)*
Choose File No file chosen

Descriptive picture of the scene (.jpg)*
Choose File No file chosen

Please upload the model's markers from 4 points of view:

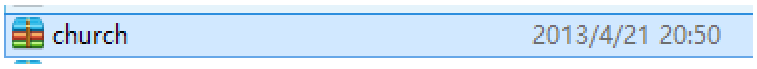
Name of the front marker*
Choose File No file chosen

Name of the back marker*
Choose File No file chosen

Name of the left marker*
Choose File No file chosen

Name of the right marker*
Choose File No file chosen

Figure 2. Upload 3D model



church 2013/4/21 20:50

Figure 3. Zipped scene file

Mobile client user manual

The 3D Relics mobile application is used to present the 3D model of historical sites as a form of augmented reality. The following instructions describe how to use the application step by step.

1. Open the application, swipe left to read the instructions on how to use the application or press “go to app” to start application right away.

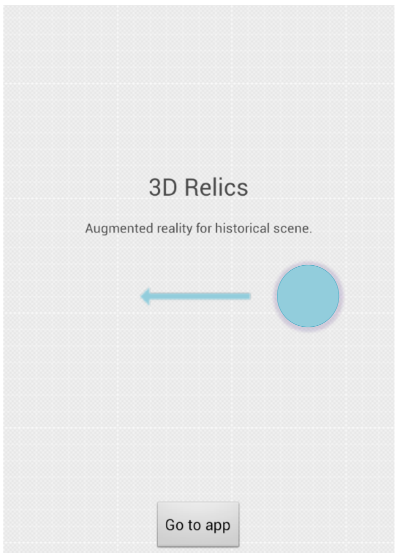


Figure 1. Welcome view.

2. Choose one of the available scenes you wish to view in the list, as shown is Figure 2.

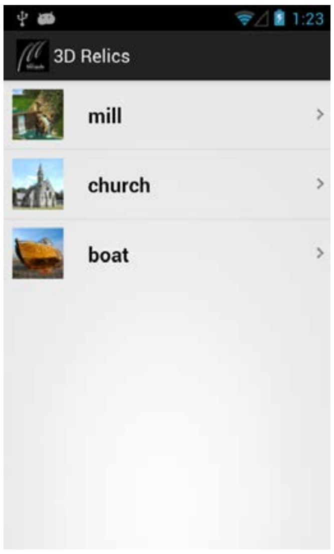
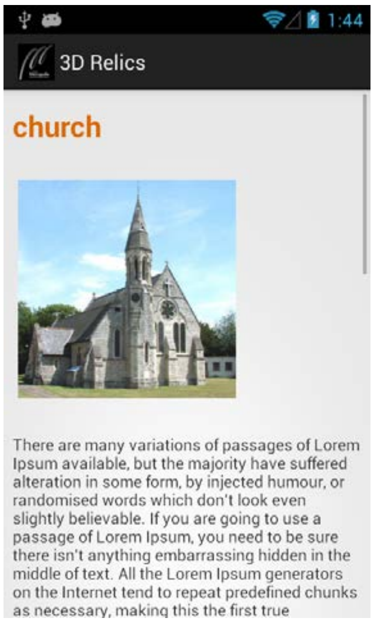


Figure 2. Available scenes from server.

3. When enter the detail view of the scene, as shown in Figure 3, you will see the description of the scene and four markers (represent front, back, left and right side of the scene) related to it.



church

There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true

Figure 3. Detailed view of the historical site.

4. Click on “Download” button to download the scene to your device, then a notification will prompt saying the application starts to download the model, as shown in Figure 4.

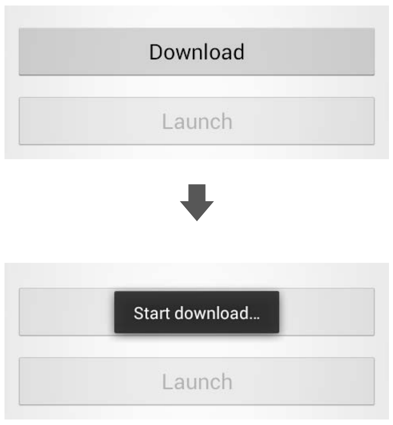


Figure 4. Download the scene.

5. During the downloading phase, you are free to download other scenes. When the download has been finished, a message will be displayed as in Figure 5.

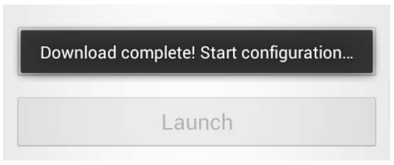


Figure 5. Download complete.

6. Before you can launch the scene, the application will first proceed a series of configuration to enable the augmented reality functionalities of downloaded scene. When the configuration has been completed, a message as Figure 6 will be prompted. Meanwhile, the “Launch” button will be activated.

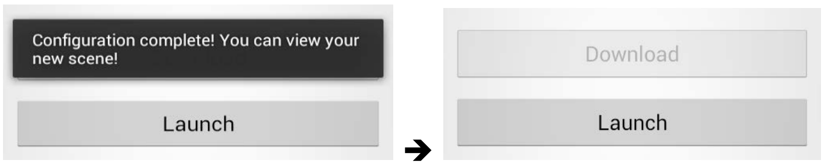


Figure 6. Configuration complete.

7. Click on the launch button and point your device to the marker, as shown in Figure 7. Once you have seen the model overlaid on the marker, you can move your device and view the augmented scene in different angles. The result you will be able to see is shown in Figure 8.

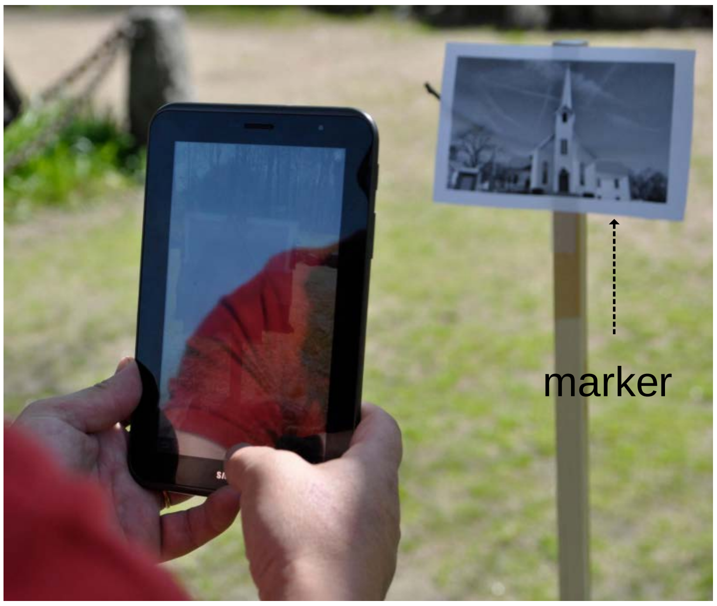


Figure 7. Interact with augmented reality scene.

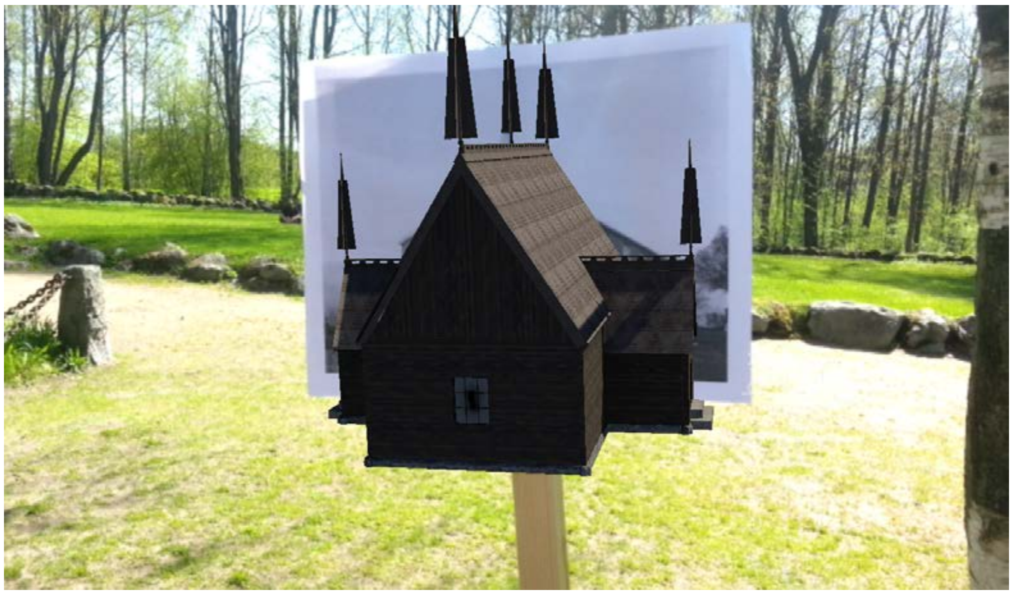


Figure 8. Rendering result from user’s point of view.

8. Once there is an update of the scene on the server, the “Update” button will be shown on the scene’s detailed view, you can update the model by following the same pressure as described above.