SMARTPHONE BASED AUGMENTED REALITY

KHOO JUN XIANG

A project report submitted in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Mechatronics Engineering

Faculty of Engineering and Science
Universiti Tunku Abdul Rahman

September 2015

DECLARATION

I hereby declare that this project report is based on my original work except for citations and quotations which have been duly acknowledged. I also declare that it has not been previously and concurrently submitted for any other degree or award at UTAR or other institutions.

Signature	:	
Name	:	Khoo Jun Xiang
ID No.	:	1201306
Date	:	7 / 9 / 2015

APPROVAL FOR SUBMISSION

I certify that this project report entitled **"SMARTPHONE BASED AUGMENTED REALITY"** was prepared by **KHOO JUN XIANG** has met the required standard for submission in partial fulfilment of the requirements for the award of Bachelor of Engineering (Hons.) Mechatronics Engineering at Universiti Tunku Abdul Rahman.

Approved by,

Signature	:	
Supervisor	:	Dr. Chong Poh Kit
Date	:	13/4/2015
Signature	:	
Co-Supervisor	:	Mr. Chuah Yea Dat
Date	:	13/4/2015

ACKNOWLEDGEMENTS

I would like to thank everyone who had contributed to the successful completion of this project. I would like to express my gratitude to my research supervisor, Dr. Chong Poh Kit for his invaluable advice, guidance and his enormous patience throughout the development of the research. Besides, my co-supervisor, Mr. Chua Yea Dat also enlightened me with his beneficial opinion and counselling and I would like to convey my thankfulness to him.

In addition, I would also like to express my gratitude to my loving parent and friends who had helped and given me encouragement in completion of this project

SMARTPHONE BASED AUGMENTED REALITY

ABSTRACT

Lots of studies on augmented reality had been carried out and prototypes were built such as marker based augmented reality on mobile device, augmented reality touring system, head mounted based augmented reality and so forth. With the advancement of smartphone’s technologies nowadays, the augmented reality has been evolve from a marker based augmented reality mobile device to non-marker based augmented reality mobile device such as object tracking, face tracking, GPS and so forth. Furthermore, most of the smartphones or tablets have built-in CPUs, GPU and sensors which enable attractive and interactive augmented reality implementation. In this project, implementation of a smartphone based augmented reality with its ability to interact using Google Cardboard has been proposed. In addition, the proposed tracking method uses ID-encoded marker as a marker of a dedicated smart device. The Google Cardboard granted features to allow user to interact with the smart device remotely. This project focused on the development of smartphone’s augmented reality application with integration of control system. The elements involved include applications development, communications and feature extraction. Two different applications will be developed separately comprising two smartphones where one built to perform a controller-like application and the other act as a passive smart device which act upon the signal delivered by the controller. In the same token, communication played an important role as transfer of signals between smart devices is required. Lastly, feature extraction make use of the existing Metaio SDK and Google Cardboard SDK to deliver applications with the desired features.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS / ABBREVIATIONS

AR	augmented reality
CPU	central processing unit
GPU	graphical processing unit
GPS	global positioning system
GUI	graphical user interface
HMD	head mounted device
SDK	software development kit

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Background

Augmented Reality (AR) has been an emerging technologies in these recent years. AR defined as immersion of virtual environment to the real environment which it enriches the vision, audition or even taste, touch and smell (Daponte, et al., 2014). The significant timeline of AR technology begun from a cinematographer, Morton Heiling which he thought cinema should enable interaction between a human and the environment covering all the senses (Daponte, et al., 2014). The first AR Head Mounted Display was developed by Sutherland (Sutherland, 1968) while the ability to interact with virtual objects was first introduced by Myron Krueger (Golan, 2006). In addition, the first mobile AR game namely ARQuake was developed by Bruce Thomas and his team in year 2000 (Thomas, et al., 2000). Later then, many of the AR applications have been created comprising Hand Mounted Device (HMD) based application or handheld devices based application.

Nowadays, smartphones have the core features to develop an AR applications such as camera, touch screen, Inertia-Measurement Unit (IMU), internet access and so forth (Daponte, et al., 2014). The advancement of smartphone technologies which enable real-time image processing camera and powerful performance from built-in Central Processing Unit (CPU) and Graphics Processing Unit (GPU) made basic AR application development easier without external supporting device. Furthermore, smartphone based AR application can be made interactive with user due to built-in sensors such as accelerometer, gyroscope, compass (magnetometer) and Global Positioning System GPS).

Apparently, a study of potential smartphone based AR applications have been carried out comprising the field of medical, education, tourism, marketing and so forth (Adhani and Rambli, 2012). Besides, more and more smartphone based application have been developed such as wikitude to explore the surroundings, augmented to visualise 3D model is augmented reality, ARBasketball to play basketball games and so forth. These show growing interest for AR technology and application in the market.

In this paper, the process to develop an interactive smartphone based AR application will be proposed. The following chapters will be arranged as follows: chapter 2 briefs the AR Software Development Kits available, tracking module available, and some description on Google Cardboard. Next, chapter 3 explains the project methodology in developing an interactive smartphone based AR application with integrating control system. Then, followed by chapter 4 explaining ways to implement the project programmatically. On the other hand, chapter 5 justify the results of this project and discuss on issues related to the project. Last but not least, chapter 6 summarise the whole project and recommend some possible improvement.

1.2 Aims and Objectives

This project aims to create a platform to build an application for Augmented Reality Control System which will be explained in Chapter 3. The primary objectives of this project is to create an augmented reality application which it is able to interact between user and smart device. Other than that, the objective of this project also target to implement new device control system other than control method that existed in the market nowadays. Thirdly, this project also intent to make use of the AR technology and available sources to create a low cost augmented reality head mounted device.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

This focus on literature study of related works from other researcher. This helped to generate ideas on how to carry out this project with reference to researchers' study. This chapter will be comparing existing augmented reality SDK in the market, analogy of tracking methods available and a research related to Google Cardboard.

2.2 Augmented Reality SDK

Customarily, developing an AR application for a beginner is very difficult because it basically includes pattern recognition, image processing, object rendering, interaction ability and so forth. All these processes have to be done altogether then only it is able to display a virtual object on a real environment. Therefore, developing an AR applications in a custom way can be time consuming and troublesome. However, few AR companies such as Metaio, Vulforia – Qualcomm, Layar and many more have developed AR Software Development Kit (SDK) which enable AR application to be developed effectively and some SDKs do not actually need any programming skills. Below are some of the AR SDKs for reference.

2.2.1 Vulforia SDK

Vulforia is an AR SDK owned by Qualcomm. The entire Vulforia AR framework comprising of AR observer, AR application customizer and AR web server. The

Vulforia SDK’s framework overview can be understood in such a way that the developer will use application customizer to develop their preference AR application and upload to the AR web server. Next, the AR observer will download that particular resources from the AR web server using HTTP and display the AR that the developer predefined. The figure below shows the overall framework or the AR SDK.

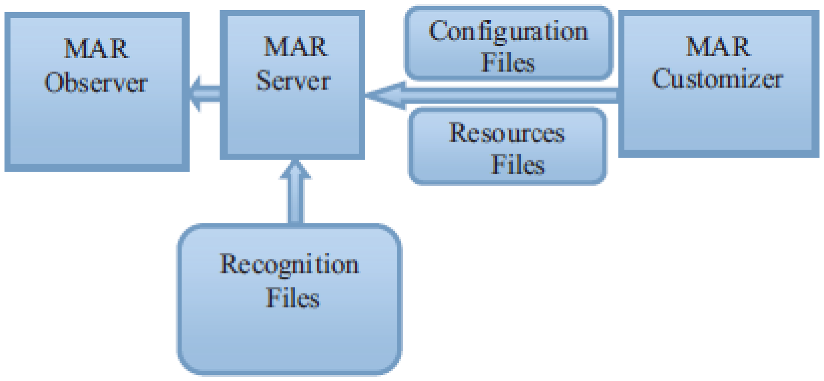


Figure 2.1: Overall AR Framework

[source: An Implementation of Generic Augmented Reality in Mobile Devices (Zhang, et al., 2014)]

The advantage of using this SDK is that the recognition technique and image tracking technology is done by the SDK, thus it made the developer is able to build a basic AR application easier and faster. Besides, it also cover text, URL, video, panorama other than 3D model. In addition, this framework is using client-server architecture, therefore the AR observer which is an application running on a smartphone able to perform different desired applications by downloading relative data file from the server. Throughout this method, the users do not need to store specific data in the smartphone and install different specific software when different AR application is needed as mentioned by the author.

However, the drawback of this method is that internet connection is needed whenever the users wanted to run the AR application. On top of that, variety of applications that available is limited, if the developers wanted to develop desired applications and it is not available in the SDK, they would still need to program the applications, design and include relevant sources and so forth.

2.2.2 Metaio SDK

Metaio SDK is a framework consist variety of component such as tracking, capturing, rendering and sensor interface. It is compatible to integrate with different platform such as IOS, Android, Unity3D and Windows. In addition, its feature includes marker or markerless tracking, face tracking, and so forth. It has some advantages like powerful 3D rendering engine, Augmented Reality Experience Language (AREL) that provide interactive AR experiences based on XML, HTML5 and Javascript, advance tracking and so forth(Amin and Govilkar, 2015).

2.2.3 D'Fusion

D'Fusion consist different component to produce different feature. For example, D'Fusion Exporter 3D is able to create 3D objects and export them and D'Fusion studio is being used to design AR projects and exportation is available too. D'Fusion SDK supported platforms consist of desktop, mobile and special Flash Plug-in.tracking features that is supported includes hand gesture, marker, markerless, face and many more (Amin and Govilkar, 2015). Besides it is also a cross-platform tools to support iPhone, Android and Web based.

2.2.4 ARMedia

The ARmedia comprises of renderer for 3D object rendering, tracker for image or markerless tracking, capture for frame capturing from the camera and interface to native android and iOS. ARmedia media framework provide variety tracking method such as natural feature trackers, gesture recognition, powerful 3D tracker and OpenCV. Besides, ARmedia is capable to support compatible smart glass and handheld devices such as android and iOS. Some features of ARmedia include dynamic lighting condition 3D tracking, ability to integrate with other AR platform, and so forth (Amin and Govilkar, 2015).

2.2.5 Wikitude

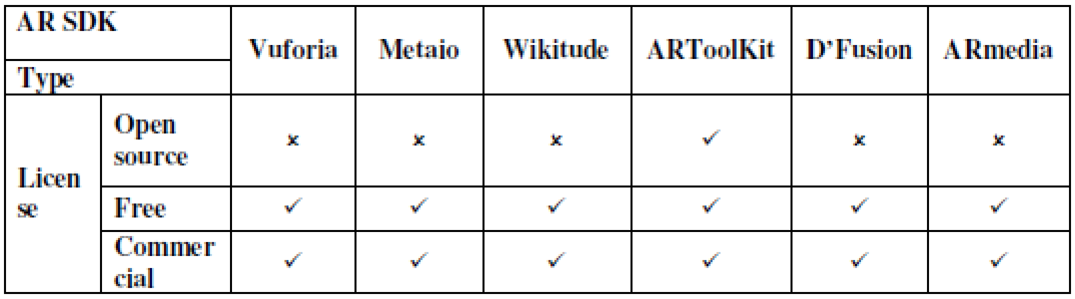
Wikitude consist of Wikitude studio which does not require programming and just simply drag and drop object on the studio screen to develop AR application. Other than Drag and Drop, it also provide HTML, Javascript and CSS for cross-platform AR application development. Besides, its tracking module includes image and GPS sensor. Wikitude also has the ability to provide hybrid tracking which it is able to combine both geolocation and image recognition for better location tracking (Amin and Govilkar, 2015). Currently, it is supporting android, iOS and Web based technology.

2.3 Comparison of the SDK

In order to make clearer comparison all the features being compared will be put in table form.

Table 2.1: License type

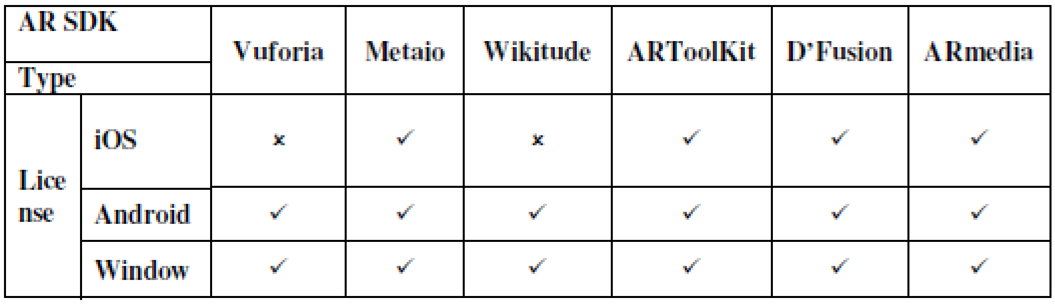
[source: Comparative Study of Augmented Reality SDK's (Amin and Govilkar, 2015)]



AR SDK / Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
License	Open source	✗	✗	✗	✓	✗	✗
	Free	✓	✓	✓	✓	✓	✓
	Commercial	✓	✓	✓	✓	✓	✓

Table 2.2: Supported platform

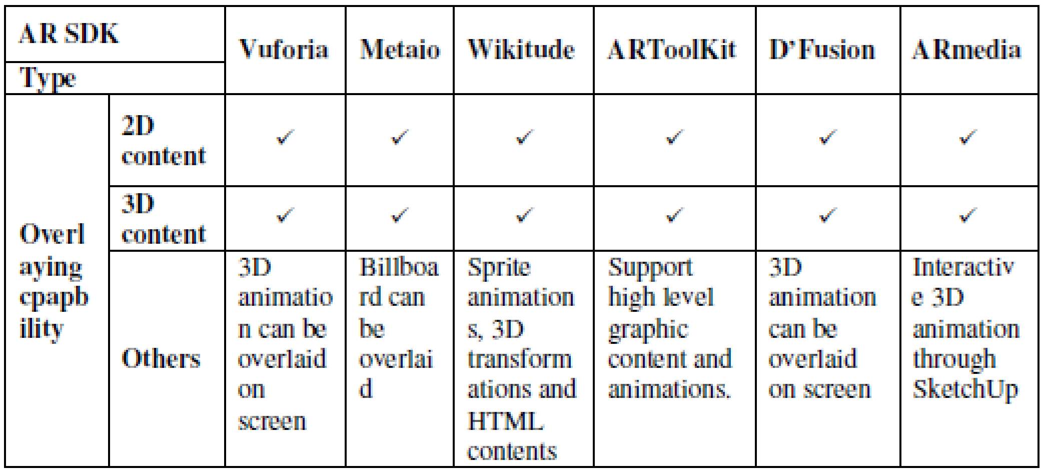
[source: Comparative Study of Augmented Reality SDK's (Amin and Govilkar, 2015)]



AR SDK / Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
License	iOS	×	✓	×	✓	✓	✓
	Android	✓	✓	✓	✓	✓	✓
	Window	✓	✓	✓	✓	✓	✓

Table 2.3: Overlaying capability

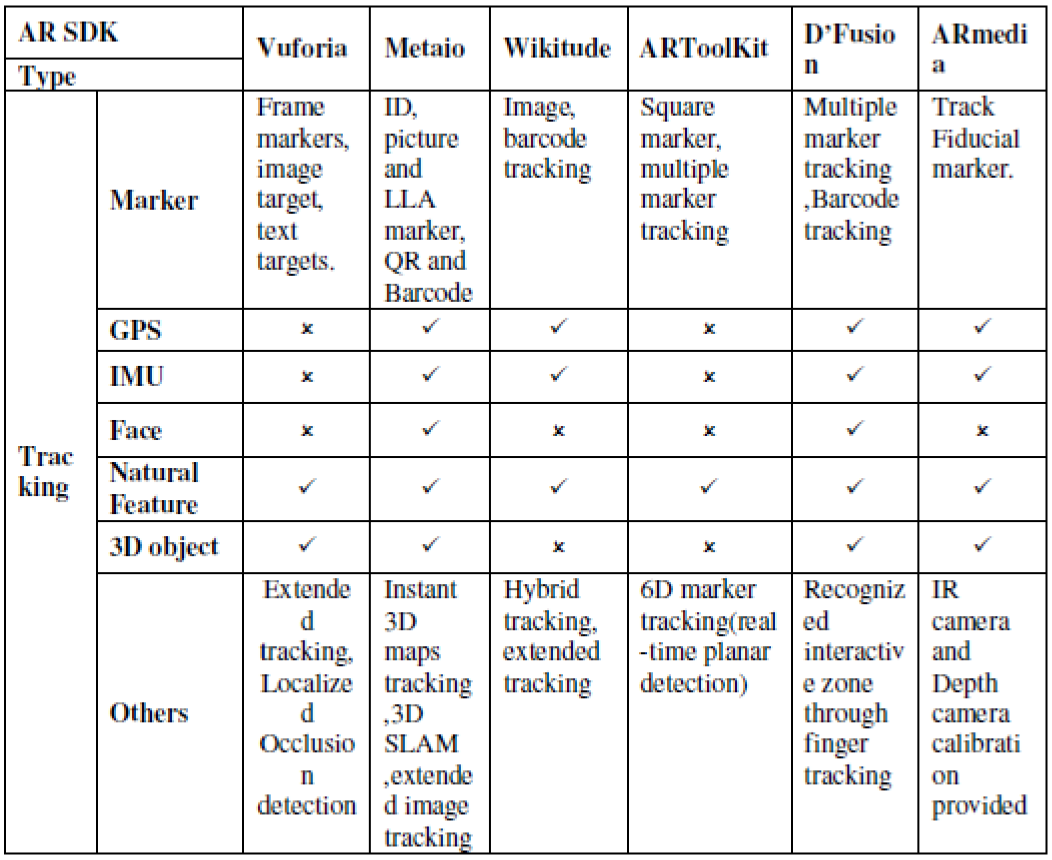
[source: Comparative Study of Augmented Reality SDK's (Amin and Govilkar, 2015)]



AR SDK / Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
Overlaying cpapbility	2D content	✓	✓	✓	✓	✓	✓
	3D content	✓	✓	✓	✓	✓	✓
	Others	3D animation can be overlaid on screen	Billboard can be overlaid	Sprite animations, 3D transformations and HTML contents	Support high level graphic content and animations.	3D animation can be overlaid on screen	Interactive 3D animation through SketchUp

Table 2.4: Tracking method.

[source: Comparative Study of Augmented Reality SDK's (Amin and Govilkar, 2015)]



AR SDK / Type		Vuforia	Metaio	Wikitude	ARToolKit	D'Fusion	ARmedia
Tracking	Marker	Frame markers, image target, text targets.	ID, picture and LLA marker, QR and Barcode	Image, barcode tracking	Square marker, multiple marker tracking	Multiple marker tracking ,Barcode tracking	Track Fiducial marker.
	GPS	×	✓	✓	×	✓	✓
	IMU	×	✓	✓	×	✓	✓
	Face	×	✓	×	×	✓	×
	Natural Feature	✓	✓	✓	✓	✓	✓
	3D object	✓	✓	×	×	✓	✓
	Others	Extended tracking, Localized Occlusion detection	Instant 3D maps tracking ,3D SLAM ,extended image tracking	Hybrid tracking, extended tracking	6D marker tracking(real-time planar detection)	Recognized interactive zone through finger tracking	IR camera and Depth camera calibration provided

2.4 Tracking Module

Tracking module plays an important role in augmented reality. Basically it determine the distinctive 3D image that is supposed to display. There are 3 types of tracking options available consisting vision based, sensor based and hybrid tracking based (Van Kravelen and Poelman, 2003). Firstly, vision based tracking module includes marker based or marker-less tracking. Commonly, marker based tracking involve picture, template marker or ID-encoded marker. On top of that, picture or template marker process in such a way that AR system track and match the pattern of marker, then it search for the corresponding virtual object and place it on the desired position. For example, in year 2013 IKEA introduce an AR app which the users are able to select and place the virtual furniture in an empty space as shown in Figure 2.2. The users will select their desired furniture and then place printed IKEA catalogue in the desired place to preview new furniture in their room before the users actually make a purchase.

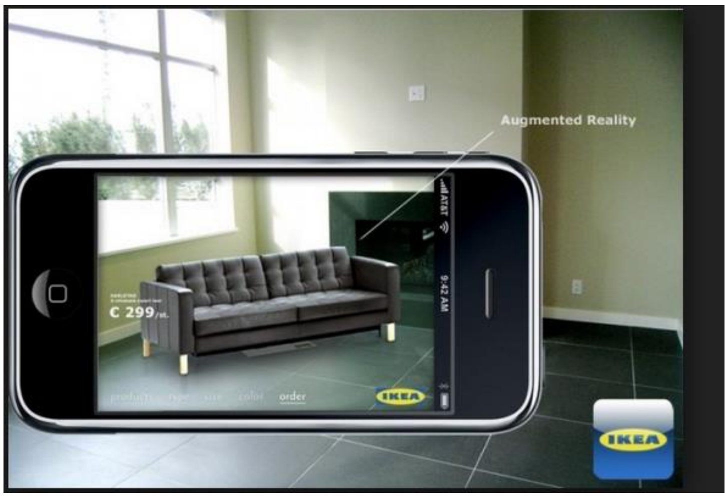


Figure 2.2: IKEA augmented reality application.

[source: IKEA to use Augmented Reality for Perfect Furniture Planning (Lee, 2013)]

In order to use the ID-encoded markers, the application require a unique decoding algorithm to determine the virtual object carried by the informative ID-encoded marker. The ID-encoded marker normally applied when a lot of markers to be matched. It is because markers have variety of intrinsic pattern, thus each unique pattern can store a unique code. However, it is difficult to apply for marker-less tracking because it applies different technique.

On the other hand, marker-less tracking module does not require any printed material to display virtual object. Instead, marker-less tracking require capture of reality objects or scenes and display subsequent virtual object when it match the respective real objects or scenes. There are 4 key points for marker-less recognition. For instance, they should be instantaneous recognition, minimum variation in different lighting condition, robust for different viewing angle and lastly necessary feature points must be provided by the object in certain range of distance between user and itself (Daponte et.al, 2014).

However, there are drawbacks for vision based tacking method. Beginning from marker based tracking system, it usually requires positioning of the markers in the real environment. Besides, the borders for some of the ID encoded markers such as ARToolkit, are relatively important where by occluded or incomplete border will disable the system from detecting the marker (Wang et al, 2010). Thirdly, the intrinsic marker inside the marker carry information in which it requires more time when there are more pattern to be matched (Wang et al, 2010).

The next tracking method is sensor based such as ultrasonic, GPS, optical and IMU (Roland et. Al, 2001). Many researches have been done on sensors based tracking. In a research, IMU sensor are being used for indoor position and motion tracking to provide navigation system (Hartmann and Trommer, 2010). Beside, position estimation using GPS triangulation method also being explained (Daponte et. Al, 2014). Moreover, separate use of sensor based or vision based tracking method have their limitations such as accuracy. Therefore, a hybrid tracking module has been proposed (Aron et. al, 2007). In this research, they proposed a method which use IMU sensors to support vision based tracking system when inconsistent results are obtained.

2.5 Google Cardboard

Google Cardboard (Figure 2.3) was introduced at the Google I/O 2014 conference. It appeared simple to build by using materials that can be easily obtained such as cardboard, magnet, 40-mm lenses, Velcro and rubber band. Design specification or template can be obtain easily through https://www.google.com/get/cardboard/get-cardboard. The Cardboard made of commonly available materials caused Cardboard to be low cost virtual reality head mounted device (HMD) as compared to Oculus Rift, Samsung’s Gear VR and many other HMDs (Yoo and Parker, 2015).

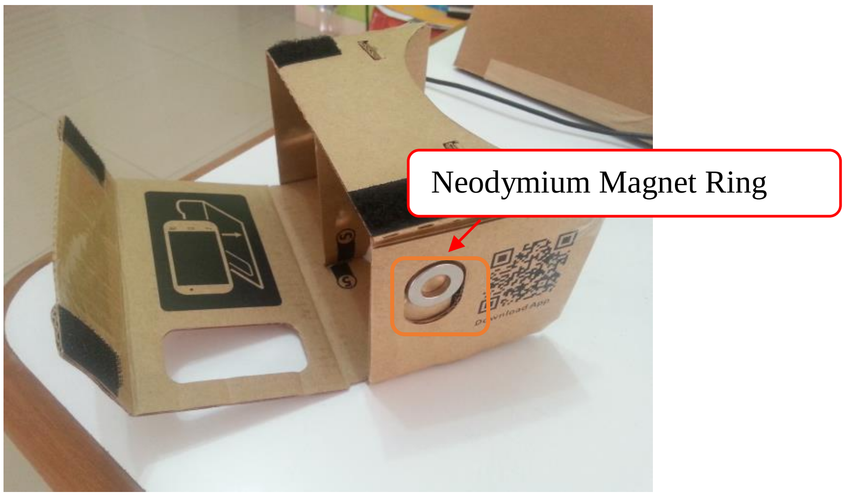


Figure 2.3: Google Cardboard

According to Yoo and Parker, they categorized 5 Cardboard interaction methods. These interaction consisting Magnetic switch, Instant Gaze, Dwelling Gaze, Tilt and External Controller are listed based on 32 Cardboard applications they had reviewed. Figure 2.4 below demonstrated the average ratings of the interaction methods preferred by the users. As observed from the statistics, tilt is the most preferable interaction method and external controller resulted in the least preferable control method. Beside, Magnetic switch is the most interaction method used with half of the apps in research utilized it. An overall rating given by the users has an average of 3.95 over 5 which indicated cardboard apps developed are above satisfaction. As a conclusion, Google Cardboard has a lot of advantage and potential for development referring to users' satisfaction feedback. Beside, its low cost nature made it inexpensive to be owned and ideal for development in many industries such as education, in a networked and collaborative environment (Yoo and Parker, 2015).

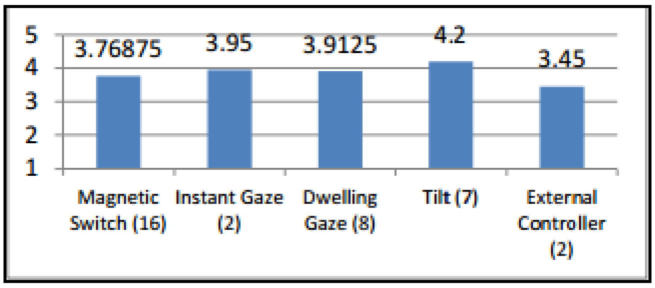


Figure 2.4: Interaction method ratings

[source: Controller-less Interaction Methods for Google Cardboard. (Yoo and Parker, 2015)]

2.5.1 Google Cardboard Magnet switch

Google Cardboard contain two magnets: a weak ceramic magnet inside the Cardboard and a neodymium ring magnet outside the Cardboard (Figure 2.3). According to an experiment ((Kjmagnetics.com, 2015), the neodymium magnet is able to demagnetize and change the magnetization of the corresponding ceramic magnet. Figure below showed the Finite Element Analysis magnetic field strength when the neodymium magnet is in place with ceramic magnet and the consequence when the neodymium magnet is pulled.

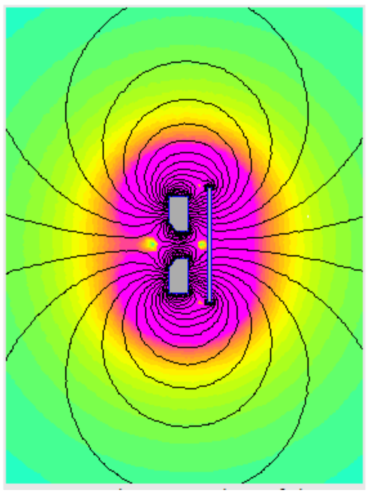


Figure 2.5: Neodymium magnet in place.

[source: Google Cardboard (Kjmagnetics.com, 2015)]

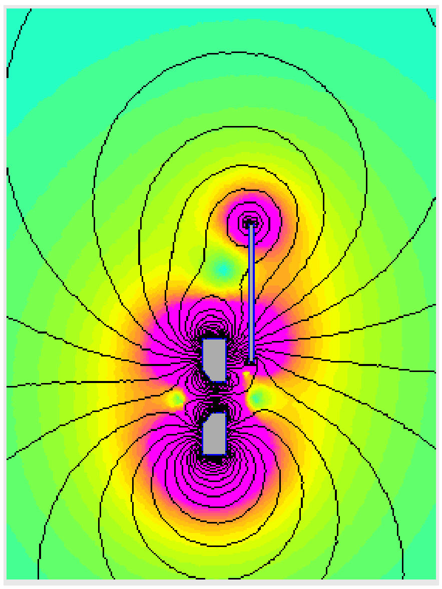


Figure 2.6: Neodymium magnet pulled down.

[source: Google Cardboard (Kjmagnetics.com, 2015)]

The simulation result showed that when the neodymium magnet is in place with ceramic magnet, strong magnetic field strength has been produced. However, when the neodymium magnet is being pulled down, visually lower magnetic field strength has been produced.

In the same token, another experiment was carried out to test on the magnetic field strength of Google Cardboard (SMART, 2015). This experiment is slightly different from the previous where this experiment run on the smartphone (new HTC One M8) containing a magnetometer. An app called "Compass" is installed in the smartphone where it able to measure magnetic field strength (unit = micro tesla, or μT) instead of Earth's magnetic field. Figure 2.7 showed the smartphone is put inside the Google Cardboard as how it used to be with the "Compass" app running concurrently.

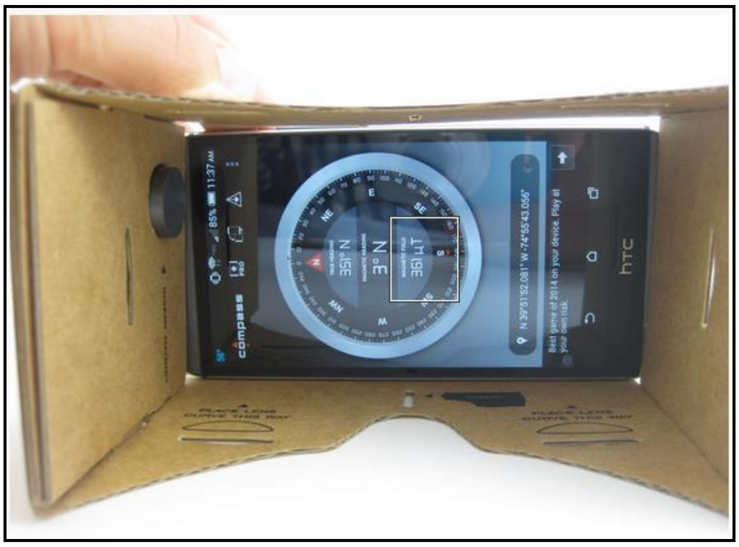


Figure 2.7: Measuring magnetic field strength with "Compass"

[source: SMART, 2015]

Then, the author measured and recorded the magnetic field strength with two scenarios. The first is the situation where neodymium is in place side by side to the ceramic magnet while the second situation is the when the neodymium magnet is pulled down and the magnetic field strength is measured. Figures below demonstrated the result of this experiment. The environment magnetic field strength is measured to be 54μT, the magnetic field strength is about 368μT when the neodymium magnet is in place and lastly the measurement is 256μT when the neodymium magnet is pulled down.

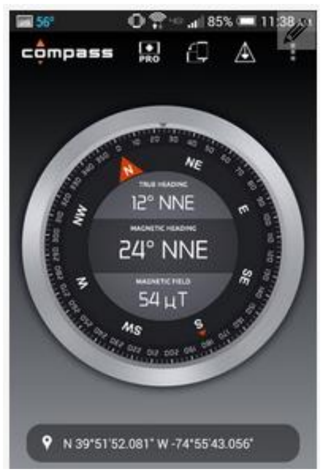


Figure 2.8: Phone by itself without Cardboard.

[source: SMART, 2015]

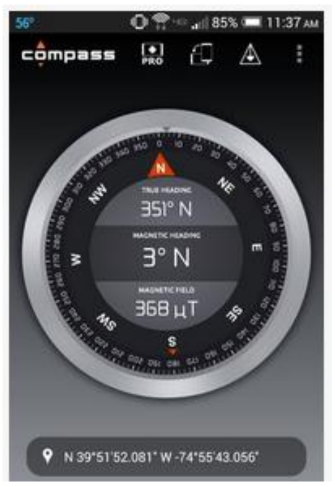


Figure 2.9: Smartphone inside Cardboard with magnet ring in place

[source: SMART, 2015]

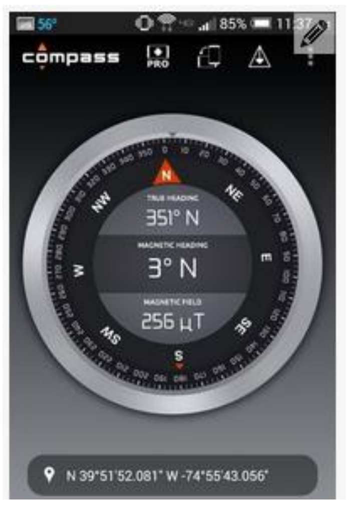


Figure 2.10: Smartphone inside Cardboard with magnet ring pulled down.

[source: SMART, 2015]

From both the experiments carried out, there are one consequence that is commonly understood whereby the neodymium magnet and ceramic magnet produce the highest magnetic field strength when they are side by side to each other. However, the magnetic field strength dropped when the neodymium is pulled down. Observation in the second experiment presented an approximate drop of 100μT in the field when neodymium magnet is pulled down. Therefore, an assumption made that the change in magnetic field strength had utilized by the app to detect the “switch” being triggered (SMART, 2015).

CHAPTER 3

METHODOLOGY

3.1 Introduction

This chapter mainly explain about proposed project management, project execution method and project chronology. Besides, brief overview will be given on system application proposed too.

3.2 Project Methodology

In order to complete this project successfully, 4 stages have been implemented. The first stages is the literature review. This is an important stage because it demonstrate the current trend of the technologies in this field. Besides, it also helps to establish projects frameworks and methodological focus.

Next, the brainstorming is needed to generate ideas on how to develop the desired system. Furthermore, this stage also identify the required software and hardware to build the system. Therefore, comparison would have to be made in order to choose the best tools needed.

Thirdly, the system prototype will be developed using Software development kit and some other programming software. At the end of this project smartphone will be used to demonstrate the results of this project.

Lastly, evaluation will be done to make sure the system is reliable and robust and improvement will be carry out to optimise the system performance. Problems and malfunction of the system should be minimised to ensure the quality of the system.

3.3 System Description

There are two applications are about to be developed in this project and it likely to mimic a situation whereby a user is using a controller to control a video player. The first application developed focus on functions comprising AR feature and controller. Meanwhile, the second application that will be developed serve as smart device to be controlled which it should have functions such as a main activity that will be running such as playing a video. In addition, this application should have simple decision making ability where it should respond to different type of signals received. Thirdly, there are three type of interactions involved. The first interaction involved user and the controller or head mounted device while the second interaction involved the controller and the smart device to be controlled. Both interaction method should be handled differently. Lastly, interaction between user and virtual object is another feature of this project. All these features and characteristics must be take into account during project execution.

3.4 System Overview

It is important to briefly look into the overall perception of the system as whole. The application flow begin with the controller functioned smartphone positioned inside the Google Cardboard with the camera turn on. The user will have to observe through the lens to the smartphone screen. When the smartphone detected an ID marker of a specified smart device, it will display the virtual 3D object. Then the virtual object is interactive whereby when the user “touch” the virtual object using magnet trigger from the Google Cardboard, the smartphone will connect to the smart device and subsequently control the smart device current activity. At the same time, the smart device must able to differentiate the signal received and make corresponding action.

The hardware that will be used in this project includes two android smartphones and a laptop. The laptop will be used to develop the AR application and modify the application code. Meanwhile, the software that needed in this project consist of Metaio SDK, Google Cardboard SDK and Android studio. Metaio SDK is needed to design the AR application, Google Cardboard is needed for magnet trigger function and Android studio is required to develop all the application needed in this project. Metaio SDK is chosen because it is free, it is easy to use for beginner in AR technology, contain powerful 3D rendering engine and most importantly it has some related source code for reference. In addition, Google Cardboard SDK is used because it is the only open source that can integrate Google Cardboard features. On the other hand, Android studio is chosen because it is free, it has a lot of open source work related to AR and android smartphone is already acquired which indicate it is cost saving too.

Secondly, creating a basic AR application will be expected to be simple as explained in the literature review. The tracking technique that will be used is ID-encoded marker. In spite of the fact that it has some limitation such as incomplete detection of marker will not produce virtual object, it is still simple to build and time efficient. Therefore, one assumption made for this project is that the application is tested out in an experimental condition whereby the ID-marker will not be block by any other object. Moreover, ID-encoded marker is ready made by the SDK, marker recognition and virtual object rendering are also done by the SDK which this would save up some time.

Since AR application design, tracking technique and virtual object rendering are early development process, thus more time will be allocated for integration of different systems to achieve head mounted augmented reality with integrated control system. However, modifying the AR application code would require some programming skills in order to make the 3D virtual object interactive and perform desired ability.

Since the fundamental application development require modification of SDKs program codes. Thus it is expected more time will be invested into understanding the program codes and modification. Besides, two applications will be constructed in this project, thus time management is important to keep this project’s progress on track. Lastly, more difficulties will be expected throughout the project development process as java programming is the main programming language. Java programming and mobile application development has never been a curriculum subject in Mechatronics Engineering, thus, challenges existed that problem solving is an important skills to overcome those challenges.

Last but not least, there some specification required in order to integrate Metaio SDK into a smartphone for example, the smartphone must have a minimum API of Android 2.3(Gingerbread), ARMv7 (Cortex) processor, OpenGLES 2.0 support, camera, GPS (Location), Accelerometer and Magnetic sensors for GPS/Compass based tracking (Metaio, 2015)

3.5 Project Management

Project management plays an important role to monitor the process of project throughout. It is needed to make sure the progress is working as what is planned earlier. Besides, it also used to identify project milestones to be achieved along the project development process.

3.5.1 Project Development Flow

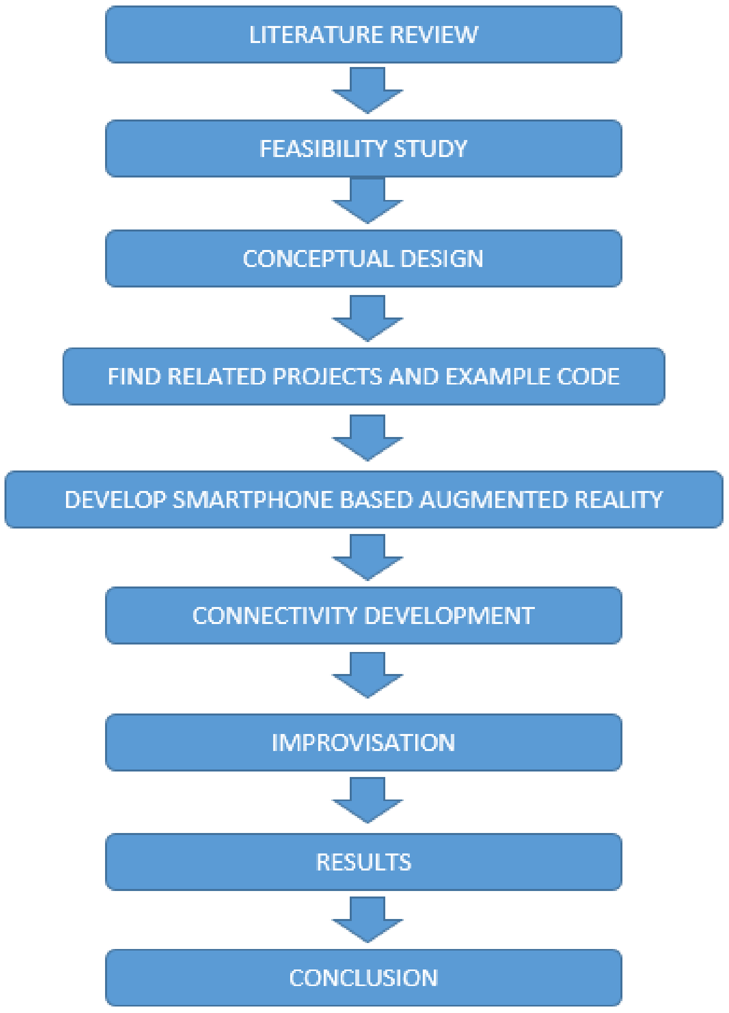


Figure 3.1: Overall system development flow.

3.5.2 Gantt Chart

The Gantt Chart below shows the project time scheduled for the coming semester of 14 weeks.

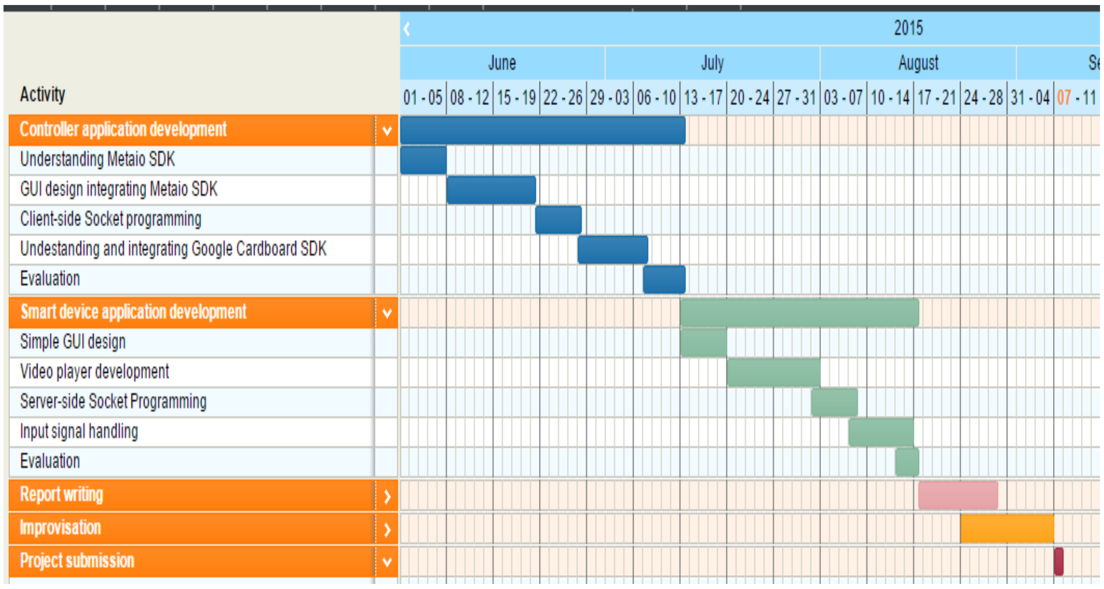


Figure 3.2: Project Gannt Chart.

CHAPTER 4

PROJECT IMPLEMENTATION

4.1 Introduction

This chapter will be explaining the process of developing the applications and algorithms used. Explanation will be divided into two sections, first section demonstrate the development of “controller” application which involves utilization of Metaio SDK, Google’s Cardboard SDK and client socket programming. On the other hand, second section put across the development of smart phone into a video player and server’s socket programming.

4.2 Development of “Controller” Application - Template

As mention above, this section unfold development of smartphone into a head mounded applicable device. It involves integration of different SDKs and communication, thus, each of the integration method will be further explained in detail. Figure 4.0 below show the program flow of the application as a whole.

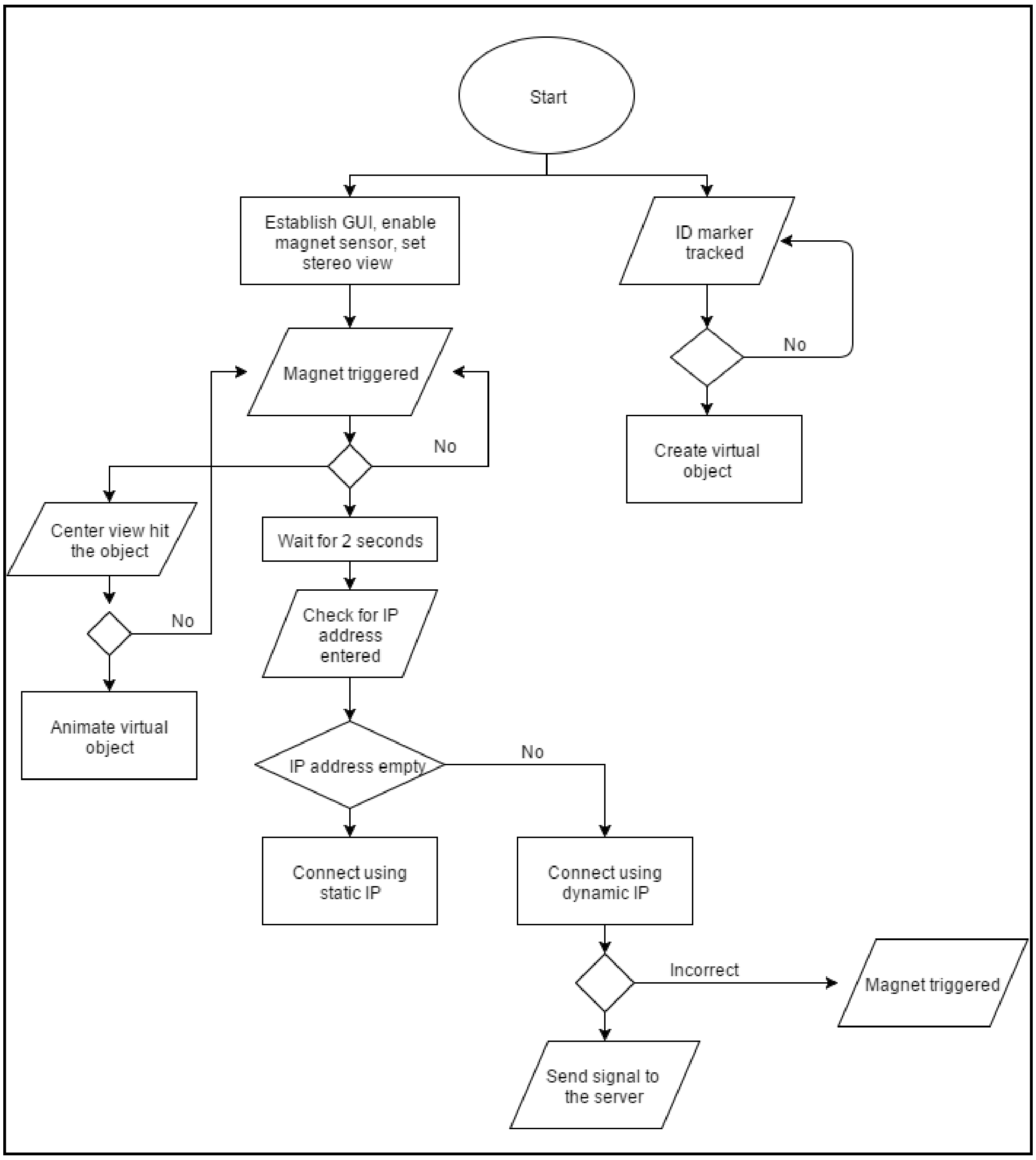
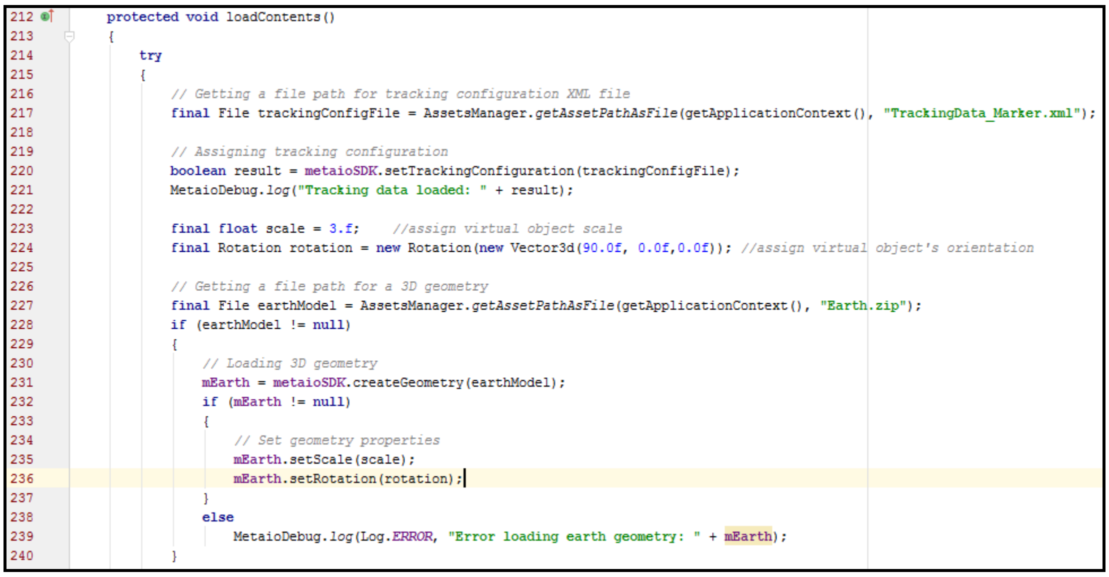


Figure 4.1: Template application program flow

4.2.1 Metaio SDK Utilization

Metaio SDK is a ready-made framework for augmented reality development with resourceful tutorials and developers forum. Few features of the SDK have been identified and modified to suit the development of this application.

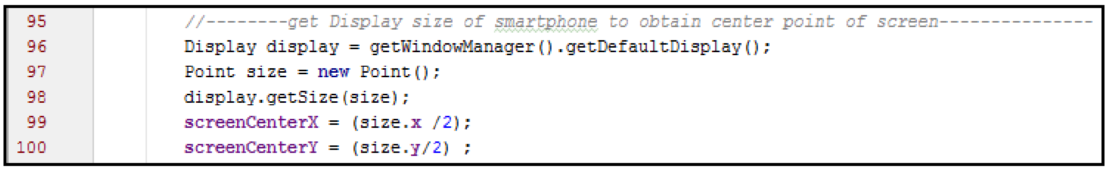
Firstly, this SDK helps to determine tracking method used in this program and the targeted ID marker is in XML file. Whenever the ID marker is captured by the operating camera, it will send signal to the program to load the virtual object. These algorithm has been shown in the program code below in Template.java.



```java
protected void loadContents()
{
    try
    {
        // Getting a file path for tracking configuration XML file
        final File trackingConfigFile = AssetsManager.getAssetPathAsFile(getApplicationContext(), "TrackingData_Marker.xml");

        // Assigning tracking configuration
        boolean result = metaioSDK.setTrackingConfiguration(trackingConfigFile);
        MetaioDebug.log("Tracking data loaded: " + result);

        final float scale = 3.f;    //assign virtual object scale
        final Rotation rotation = new Rotation(new Vector3d(90.0f, 0.0f,0.0f)); //assign virtual object's orientation

        // Getting a file path for a 3D geometry
        final File earthModel = AssetsManager.getAssetPathAsFile(getApplicationContext(), "Earth.zip");
        if (earthModel != null)
        {
            // Loading 3D geometry
            mEarth = metaioSDK.createGeometry(earthModel);
            if (mEarth != null)
            {
                // Set geometry properties
                mEarth.setScale(scale);
                mEarth.setRotation(rotation);
            }
            else
                MetaioDebug.log(Log.ERROR, "Error loading earth geometry: " + mEarth);
        }
```

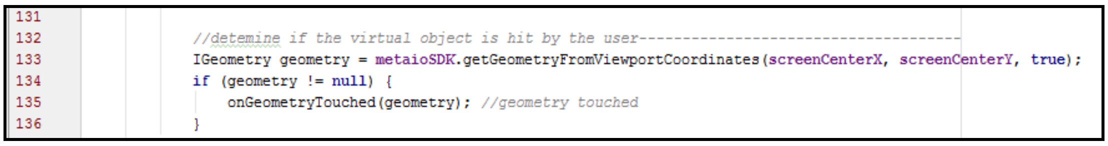
Figure 4.2 : Track and load virtual object.

The virtual object file is restricted to few format types such as .obj, .fbx, and .md2 (Metaio.com, 2015). The size and orientation of the virtual object can be define manually by the programmer. Besides, the SDK also enable interaction between user and virtual object. The SDK enable user to touch the virtual object and corresponding action will be taken. In this application, touching the virtual object is not possible as the smart phone is mounted inside Google's Cardboard. Therefore, an alternative method has been tried by setting the center point of the screen and whenever the magnet is triggered, the coordinate of the virtual object is compared to the screen's center point. If the centered point lies on within the boundary of virtual object, then the virtual object considered being touch as shown in figure below.



```
//--------get Display size of smartphone to obtain center point of screen---------------
Display display = getWindowManager().getDefaultDisplay();
Point size = new Point();
display.getSize(size);
screenCenterX = (size.x /2);
screenCenterY = (size.y/2) ;
```

Figure 4.3: Obtain center coordinate of the screen.

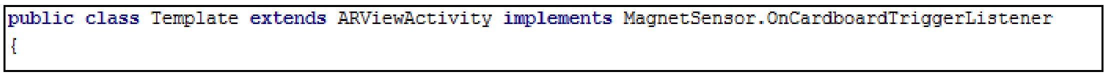


```
//detemine if the virtual object is hit by the user-----------------------------------
IGeometry geometry = metaioSDK.getGeometryFromViewportCoordinates(screenCenterX, screenCenterY, true);
if (geometry != null) {
    onGeometryTouched(geometry); //geometry touched
}
```

Figure 4.4: Check if virtual object is touched.

4.2.2 Google’s Cardboard SDK Utilization

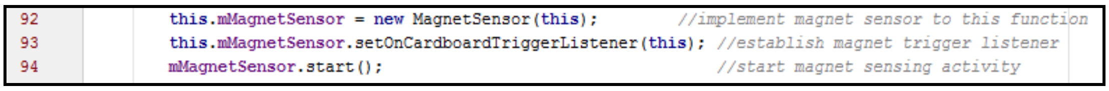
Google’s Cardboard was developed to support virtual reality development and smartphone was used for virtual environment construction. However, only magnet trigger function will be integrate to this application. In order to integrate magnet trigger feature to the application, the main program must inherit the MagnetSensor.OnCardboardTriggerListener class such as the figure below.



```
public class Template extends ARViewActivity implements MagnetSensor.OnCardboardTriggerListener
{
```

Figure 4.5: MagnetSensor inheritance.

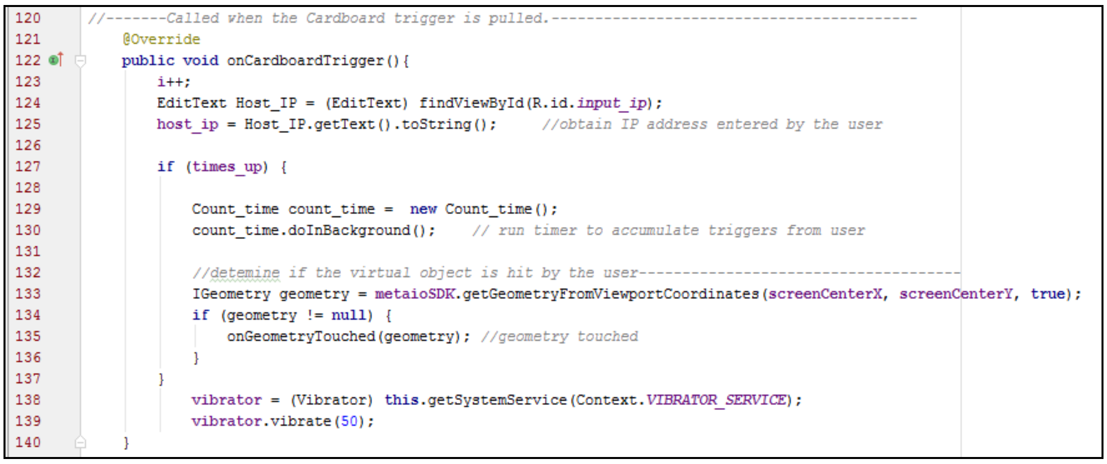
With the inheritance of MagnetSensor.OnCardboardTriggerListener class, it enable the main function to initiate magnet sensor listener to handle magnetic change when occur. Besides, magnet sensor activity has to be started to operate magnetometer embedded in the smartphone and subsequently detect magnetization of cardboard’s magnet. Initialization of magnetic sensor has been shown in Figure 4.6.



```
this.mMagnetSensor = new MagnetSensor(this);        //implement magnet sensor to this function
this.mMagnetSensor.setOnCardboardTriggerListener(this); //establish magnet trigger listener
mMagnetSensor.start();                                  //start magnet sensing activity
```

Figure 4.6: MagnetSensor initialization.

With all the function and listener initialised, feedback to the listener has been made easier by calling onCardboardTrigger function. This function has to be an override method to overrule the prior function declared in the super method because the main function inherited MagnetSensor.OnCardboardTriggerListener class. Figure 4.2.6 demonstrated the usage of onCardboardTrigger function.

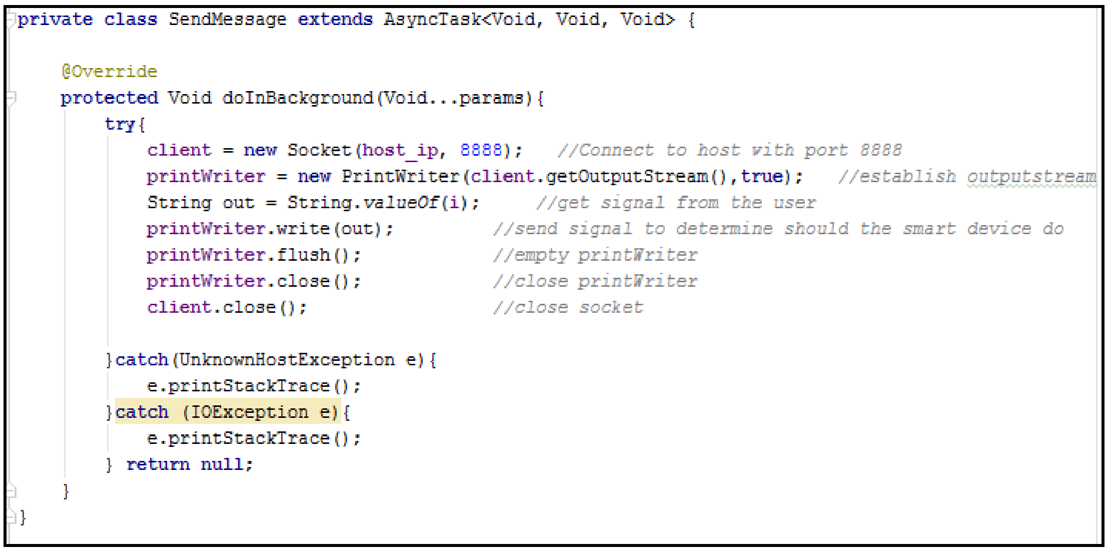


```java
//-------Called when the Cardboard trigger is pulled.-----------------------------------------
    @Override
    public void onCardboardTrigger(){
        i++;
        EditText Host_IP = (EditText) findViewById(R.id.input_ip);
        host_ip = Host_IP.getText().toString();     //obtain IP address entered by the user

        if (times_up) {

            Count_time count_time =  new Count_time();
            count_time.doInBackground();    // run timer to accumulate triggers from user

            //detemine if the virtual object is hit by the user------------------------------------
            IGeometry geometry = metaioSDK.getGeometryFromViewportCoordinates(screenCenterX, screenCenterY, true);
            if (geometry != null) {
                onGeometryTouched(geometry); //geometry touched
            }
        }
            vibrator = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
            vibrator.vibrate(50);
    }
```

Figure 4.7: onCardboardTrigger thread.

4.2.3 Client-side Socket Programming

Communication between “Controller” and smart device is established using Wi-Fi or TCP/IP protocol. Since TCP/IP protocol is being used, socket programming is important to connect and communicate between two devices over the network. Due to the reason that static IP address is being used in this application, then communication thread will automatically connect to the dedicated smart devices after magnet is being triggered. Numbers of triggers will be sent to the smart device as illustrated in the figure below.



```
private class SendMessage extends AsyncTask<Void, Void, Void> {

    @Override
    protected Void doInBackground(Void...params){
        try{
            client = new Socket(host_ip, 8888);   //Connect to host with port 8888
            printWriter = new PrintWriter(client.getOutputStream(),true);   //establish outputstream
            String out = String.valueOf(i);    //get signal from the user
            printWriter.write(out);         //send signal to determine should the smart device do
            printWriter.flush();            //empty printWriter
            printWriter.close();            //close printWriter
            client.close();                 //close socket

        }catch(UnknownHostException e){
            e.printStackTrace();
        }catch (IOException e){
            e.printStackTrace();
        } return null;
    }
}
```

Figure 4.8: Communication thread.

4.3 Development of Smart Device Application – Android Client

This subchapter will be revealing development of video player in a smartphone. Since this smart device also act as a server, socket programming on the server side will be further explained. On top of that, signal handling of the smart device is another essential section to be justified. Last but not least, device window and layout management will be explained in detail at the end of this subchapter. The whole program flow will be illustrated in the figure below.

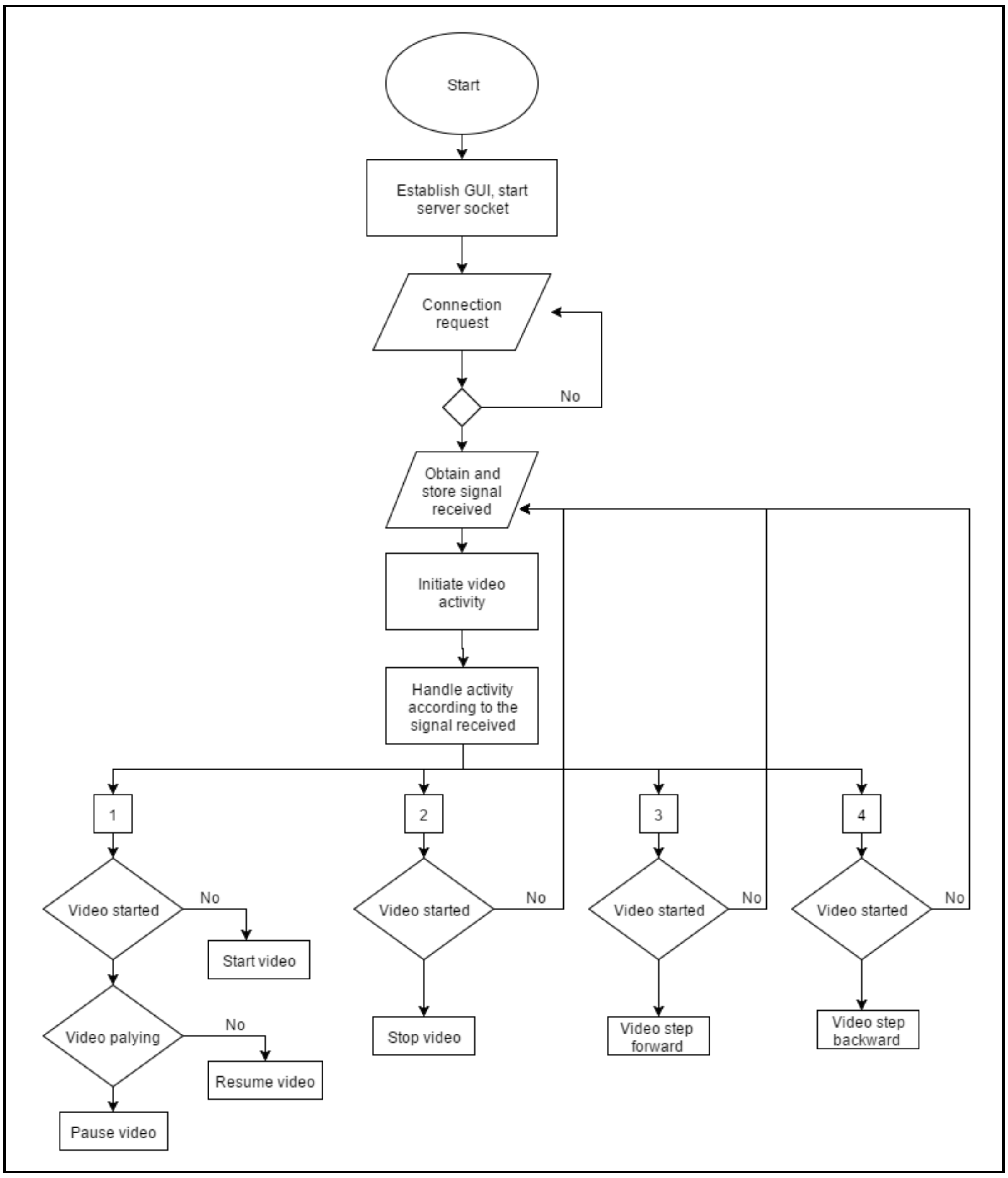
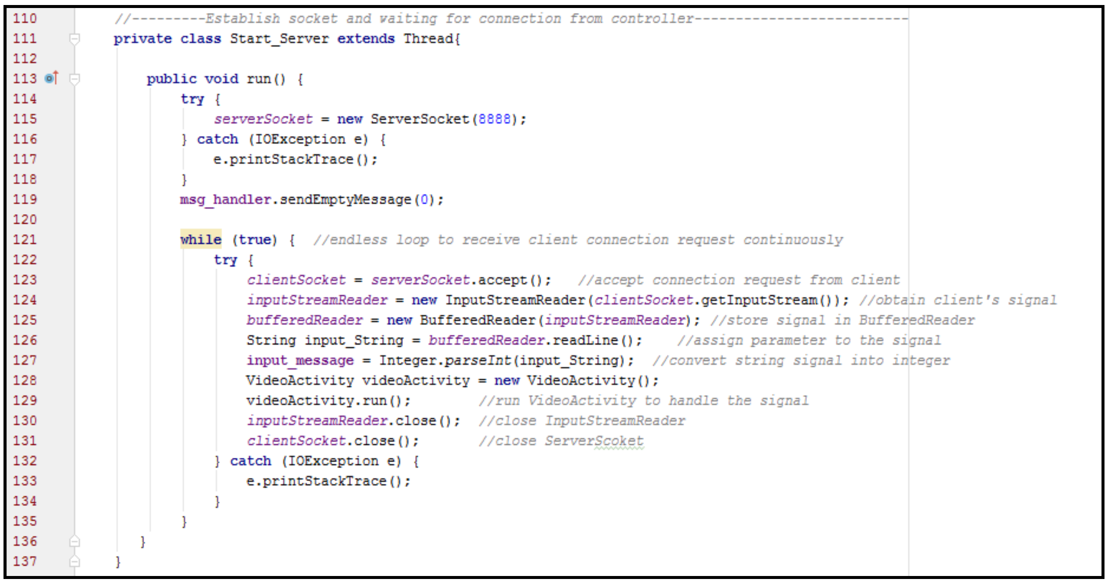


Figure 4.9: Android-client application program flow

4.3.1 Server-side Socket Programming

As the smart device need to continuously pending connection request from the client, thus, it is more likely to build the function in an thread that run endlessly in background. Once the server receive connection request from the client, it will approve the connectivity and begin communication. Then, the client's signal will be sent to the InputStreamReader and further stored in BufferedReader. As the signal received is usually in string, thus conversion to integer is required to handle the data easier. All these process are illustrated in detail in the figure below.



```java
//---------Establish socket and waiting for connection from controller-------------------------
private class Start_Server extends Thread{

    public void run() {
        try {
            serverSocket = new ServerSocket(8888);
        } catch (IOException e) {
            e.printStackTrace();
        }
        msg_handler.sendEmptyMessage(0);

        while (true) {  //endless loop to receive client connection request continuously
            try {
                clientSocket = serverSocket.accept();   //accept connection request from client
                inputStreamReader = new InputStreamReader(clientSocket.getInputStream()); //obtain client's signal
                bufferedReader = new BufferedReader(inputStreamReader); //store signal in BufferedReader
                String input_String = bufferedReader.readLine();    //assign parameter to the signal
                input_message = Integer.parseInt(input_String);  //convert string signal into integer
                VideoActivity videoActivity = new VideoActivity();
                videoActivity.run();        //run VideoActivity to handle the signal
                inputStreamReader.close();  //close InputStreamReader
                clientSocket.close();       //close ServerScoket
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Figure 4.10: Server socket programming.

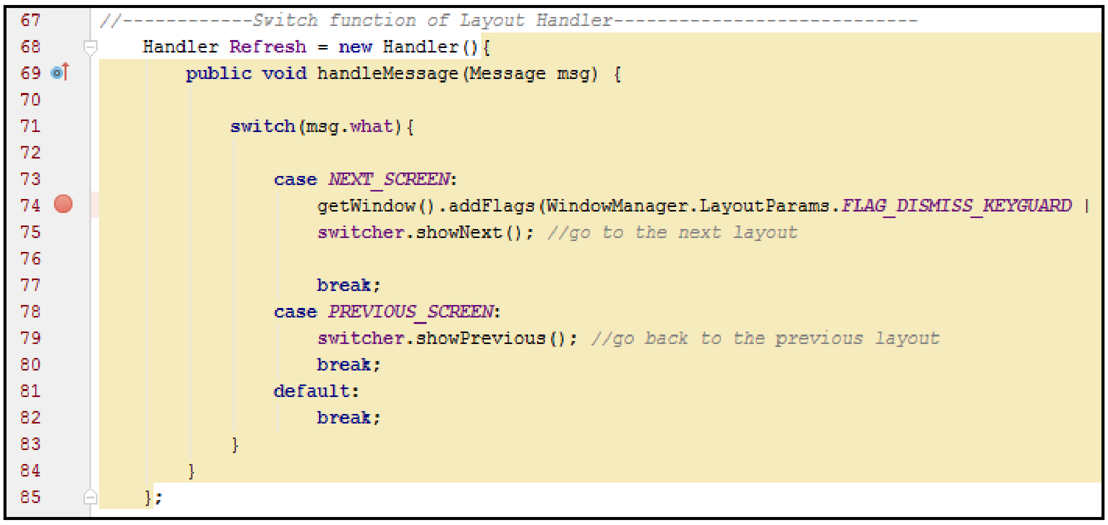
4.3.2 Video Player Development and Input Signal Handling

Since multiple signals will be sent the server, a thread consisting switch function is used to handle each signal different. As observed from the program flow earlier, 4 types of signal will be handled by VideoActivity thread. It is self-explanatory from the code on how each signal received correspond to specific video activity. Since this application is used for demo purpose, a short clip will be embedded into this application to ease the demonstration. VideoView is being used to reserve a space on the screen to play the video and the setVideoURI determine the address of the video to be played later. The video stream can be control by setting start(), pause() and stopPlayback(). When the video is being paused, it is recommended to save the frame that have been watch using getCurrentPosition() and make sure the video resume from the exact frame using seekTo() function. Beside, a parameter, video_played is being used to verify if the video has been played.

4.3.3 Device Window and Layout Management

The smartphone may either be in screen-on mode or sleep mode when server socket is pending for request in background. Smartphone has no issue to play the video in screen-on mode but sleep mode stated otherwise. Therefore, the following code "getWindow().addFlags(WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD | WindowManager.LayoutParams.FLAG_FULLSCREEN | WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED | WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);" is important for the application to bypass the safety keyguard of the smartphone and able to turn on the screen of the smartphone.

Besides, there are two layout design in this application. Therefore, a swither is used to swap the layout whenever needed. Appendix A show that two layout designs have been developed in a single xml file. The reason of using two different layout is to handle two different situation consisting one communication activity pending for connection from the client and a video activity. The order of the layout is important as it determine how the layout can be controlled. Figure below illustrate how this function is being manipulated.



```
//------------Switch function of Layout Handler----------------------------
    Handler Refresh = new Handler(){
        public void handleMessage(Message msg) {

            switch(msg.what){

                case NEXT_SCREEN:
                    getWindow().addFlags(WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD |
                    switcher.showNext(); //go to the next layout

                    break;
                case PREVIOUS_SCREEN:
                    switcher.showPrevious(); //go back to the previous layout
                    break;
                default:
                    break;
            }
        }
    };
```

Figure 4.11: Switch utilization.

CHAPTER 5

RESULTS AND DISCUSSION

5.1 Introduction

In this chapter, the results and operations of the application will be demonstrated. This chapter will also justify the results obtained, problems encountered along the prototype development, discussion on the issues that has yet to be solved and possible solutions can be made to the system.

5.2 Results Presentation

In this project, two smartphones have been used. The first smartphone is LG nexus 4 (Figure 5.1), while the second smarphone is Samsung S3 (Figure 5.2). Besides, a google cardborad(Figure 5.3) is being used to integrate with smartphone to produce head mounted device.

Figure 5.1: LG Nexus 4.

Figure 5.2: Samsung S3.

As discuss earlier, two applications have been developed in this project, the first application known as “Template” will be installed in LG Nexus 4 and it will be known as controller in this chapter. Subsequently, the application “android-client” will be installed in Samsung S3, where it will be targeted smart device to be controlled. In this case, Samsung S3 is used to mimic the function of a video player playing video in a smart TV. Before running the applications, both of the smartphones must be connected to the same network or wifi. Next, since the “android-client” application served as a server pending for connection request, thus it should be activated before hand so that the smart device will continuously waiting for connections form the controller. The smart devices can be left in either screen-on mode or sleep mode because the video will able to be played remotely regardless of the mode it is positioned.

Next, the “Template”application in the Nexus 4 can be started subsequently and it will establish a GUI as shown in Figure 5.4. The user does not required to enter the server IP address manually because static IP address will be used in this application. However, the user will only needed to enter the IP address manually if the static IP assigned is failed to connect both devices. Thus, a dynamic IP of the desired smart device is used as alternative, Figure 5.5 shows an example IP address obtained from Samsung S3. At the same time, the red circles on the phone screen act as a centered eye view of the user as shown in Figure 5.4. The centered eye view help user to engage with the virtual object displayed on the screen. This feature helps to improve input handling from the magnet triggers. As mention in the earlier chapter, an ID marker is used for smart device recognition. When an ID marker is identified by the application, a virtual object will appear on the screen, visually above the ID marker as shown in Figure 5.5. Next, the user will able to pull down the magnet trigger on the left of the cardboard to connect with the desired smart device. However, there is a additional features to improve the input handling as mention earlier. When the magnet trigger is being pulled and released, the application will compare the centered eye view coordinate to the virtual object. If the centered eye view coordinates match with the virtual object, the controller will be able to connect with the smart device. On the other hand, if the user is not looking as the virtual object, then triggering the magnet will not result in any difference to the application. Once the magnetic object is being triggered with virtual object set in view, connection from controller to the smart device will be established with signal sent to the smart device. The smart device repoonse dependant on the signal it receives. The number of magnet triggerd in a period of 2 seconds determine what the smart device should do. A pull and release of the neodymium magnet ring is considered a trigger. If the movie is not playing, and the user pulls the trigger once, it will play a short video clip in the smart device and vice versa. Next, 2 triggers stop the video clip immediately and step forward and step backward can be done by trigger the magnet 3 times and 4 times respectively. Figure 5.6 showed that a video is being played on the smart device.

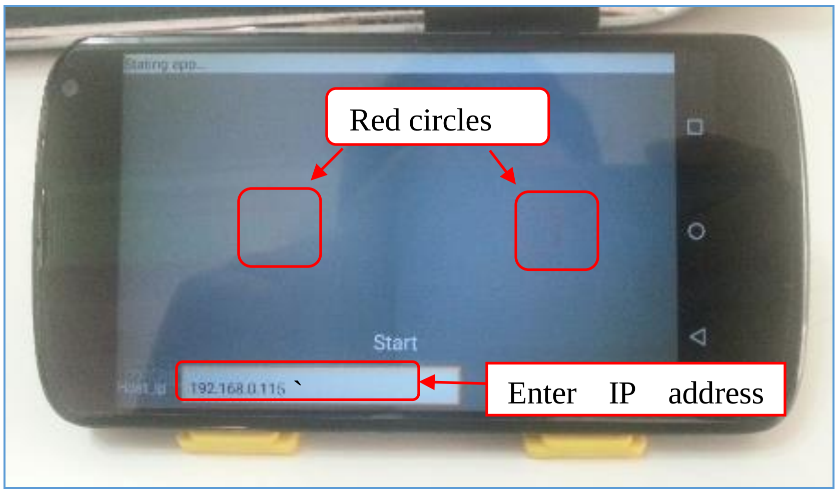
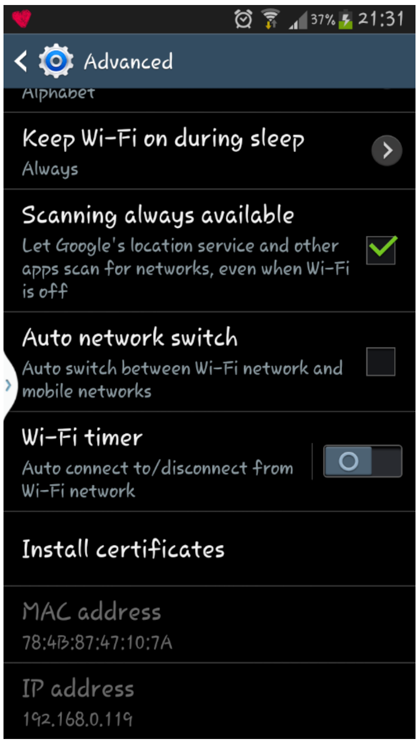


Figure 5.3: GUI of “Template” application.


21:31

Advanced

Alphabet

Keep Wi-Fi on during sleep
Always

Scanning always available
Let Google's location service and other apps scan for networks, even when Wi-Fi is off

Auto network switch
Auto switch between Wi-Fi network and mobile networks

Wi-Fi timer
Auto connect to/disconnect from Wi-Fi network

Install certificates

MAC address
78:4B:87:47:10:7A

IP address
192.168.0.119

Figure 5.4: IP address of Samsung S3.

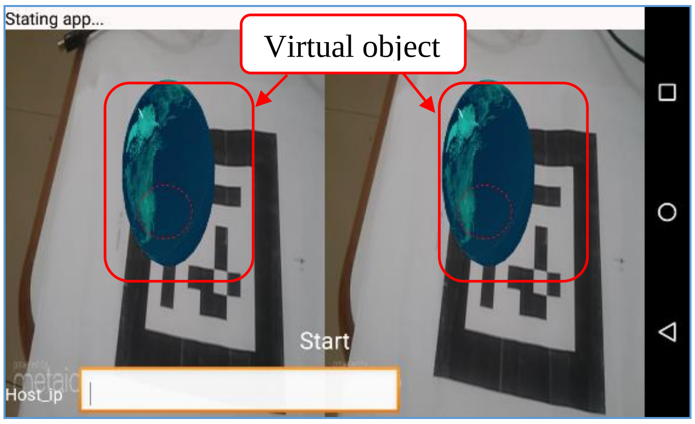


Figure 5.5: Virtual object created.

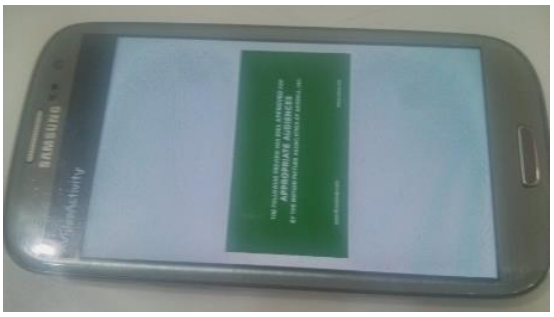


Figure 5.6: Playing video on the smart device.

5.3 Application performance statistics

It is important to know the performance of the application on memory usage, CPU performance, and GPU performance is also being tested since virtual object creation required graphical processing. However, this analysis focus on template application only as this application is the main application to be focused in this project.

This application analysis begin with memory usage by the applications. Memory monitoring is good in determining the allocated memory to the application, available and used memory by the application and a quick test on determining the app crash might be related to running out of memory. Figure 5.7 below showed the memory monitor reports in a period of time.

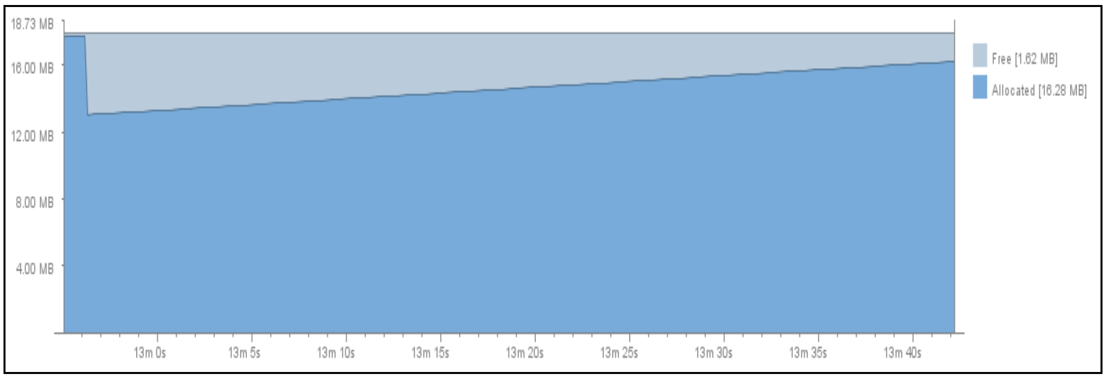


Figure 5.7 : Memory usage over time.

As observed from the graph, the dark blue indicates the amount of memory that the app is currently using, while the light blue indicate the available and unallocated memory. The total allocated memory for application is about 18MB. Whenever garbage collection occur, the memory usage dropped to 13MB resulting roughly 5MB garbage memory collected. Depending on the available memory of the smartphone, in the case of smartphone used in this project, nexus 4 has a total usable memory of 1700MB. Thus, this 18MB is relatively small at about 1% of the total memory. Thus, the application will not slow down the smartphone severely when the application is in used.

Secondly, Figure 5.8 illustrate the CPU performance over time by the application. The pink colour indicate the CPU used by the user which the red colour represent the kernel's CPU usage. The graph showed the highest user's CPU usage is at approximately 30% while the kernel used about 5-10%. The CPU usage application is less than half as observed from the graph. This result indicate that the application is not burdening the smartphone, to run in full speed.

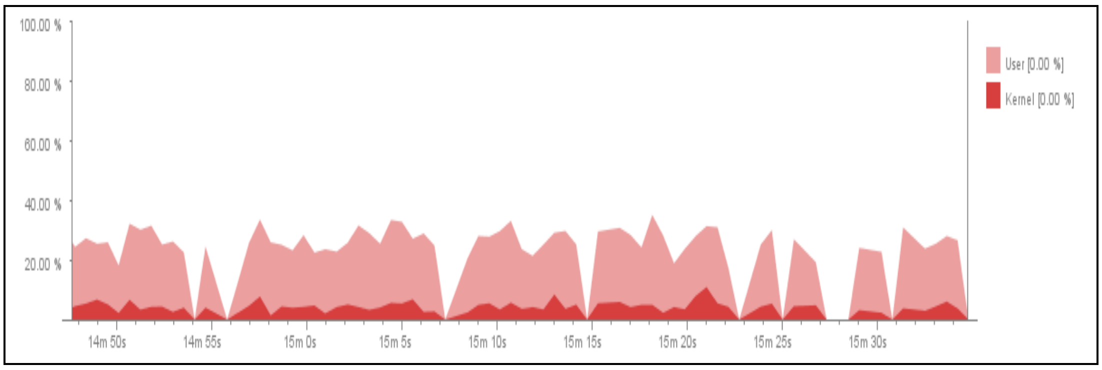


Figure 5.8 : CPU usage over time.

Thirdly, since virtual reality is being implemented in the smartphone and openGLES 2.0 is being utilised. Therefore, GPU monitoring can be useful to inspect the application performance. Figure 5.9 below illustrated the GPU performance over a period of time.

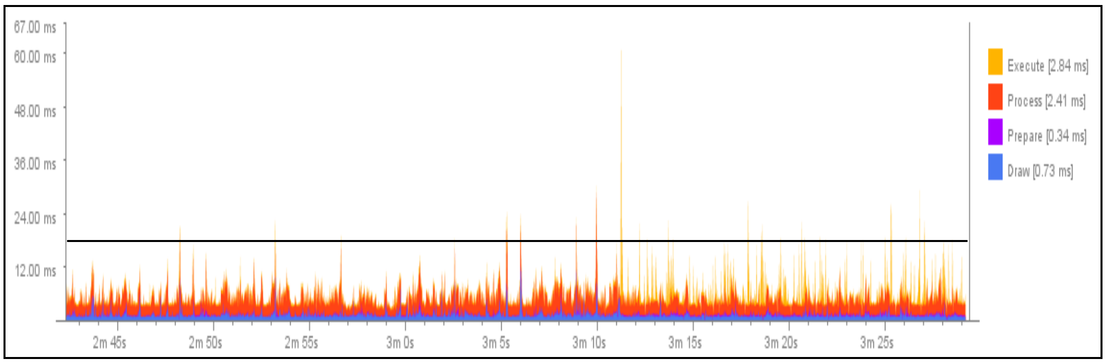


Figure 5.9 : GPU usage over time.

The graph demonstrate the time per frame in millisecond against period of time. According to the information(), the GPU monitoring gives graphical statistics on how the application user interface window perform based on the 16ms per frame target. On top of that, it also able to identify rendering pipeline stand out throughout the processing time. Lastly, spikes in frame rendering can be easily determine too. The graph can be divided into two parts, the first part indicate the application is in a condition where ID-encoded marker is not detected, resulting no virtual object to be created. Next, as the time passed through 3m10s, the application is in a situation where the camera is observing the ID-encoded marker, and a virtual earth is being displayed on the screen. The orange colour indicates the GPU is in execution process. There is an important information in this graph whereby the frame rendering of a user interface window should not be more than 16ms per frame benchmark. It the application does, that animation of the application may be distorted. It is because whenever a frame exceeded the benchmark, the application is missing a frame resulting stuttering images. Although there are some spikes that is over the bench mark, the animation result is still visually acceptable. However, this issue shall be resolved to enhance user experience.

5.4 Significant Changes

Both the SDK are built to support different applications. Metaio SDK primarily supports the augmented reality applications development while Google Cardboard SDK support virtual reality development. Both of the SDK require OpenGLES 2.0 for the smart device to develop virtual environment or virtual object but the SDKs are utilizing it differently. Consequently, GUI development resulted in a major issue because the program clashes when both libraries were being integrated at the same time. However, the program works fine when the SDK GUI develop separately. Therefore, Metaio SDK was being used for GUI development solely while Google Cardboard SDK was being used for its magnet trigger input handling.

Magnetic input handling was crucial in this project because it enabled interaction between user and the smart device. The benefit of magnetic trigger is that it has straight forward function and easy to handle by user with no additional knowledge required. Due to the reason that magnetic input is monotonous, thus combinations of magnet being pulled are developed and they are used by the application to perform different tasks. Different numbers of pulls from the user determine the task desired for the application, however, this may limit the number of tasks that can be performed by the application because it is inconvenient to the user when more pulls are required.

Centered view of the camera has also being utilized to improve the system. The coordinate of centered camera view is being used to engage the virtual object which is targeted by the user. If there are a few different virtual objects appear at the same time, only the virtual object coordinates that are correspond to the centered eye view will be interacted. This enables user to choose which device they wanted to control. Besides, this may help to avoid confusion to the system too.

In addition, socket programming has been integrated into Metaio SDK. This feature is very important to establish connections between the controller and the smart device to be controlled. TCP/IP protocol is being used in this project. In the same token, static IP is the primary address to be used in this application. This help to reduce inconvenience to the user that user would not have to manually search for the smart device’s IP address and input the respective IP address to the controller. However, dynamic IP address also being used as an alternative in this application. This help to act as a backup plan if the static IP address is not able to establish connectivity between the smartphones. There are advantage using static IP address since the server’s IP address must always be known and it is constant. Besides, it is more reliable as compared to dynamic IP address. Commonly, static IP address result is security risk as it is easier to track for data. On the other hand, dynamic IP assign new IP address each time the user connect to network. Thus, frequent checking and manual input is required for every time the user wanted to connect to the smart device and it is rather redundant. In contrast, dynamic IP address results lesser security risk. Moreover, dynamic IP addressing is more cost effective than static IP as there is automatic network configuration and lesser human intervention.

5.5 Problems Encountered and Solutions

5.5.1 Utilizing the SDKs

Both the Metaio SDK and Google's Cardboard SDK are well developed and tutorials are given. However, it takes time to be familiar with the SDK and understand the SDK so that changes can be made correctly and effectively. Trials and errors and debugging tools are very useful to identify the working flow of the codes and program. Besides, internet resources such as existing tutorials on the SDK's website, forums, and blogs were helpful in utilizing the

5.5.2 Time Management

Time given to complete this project was limited to 14 weeks. However, it is a challenge because there are ongoing academic subjects need to be carried out concurrently with this project. Therefore, multitasking is very important because this project must be completed without affecting academic performance. In addition, time management is also key factor to ensure this project hits the checkpoint throughout the process and able to be completed on time.

5.5.3 Metaio SDK

Using the Metaio SDK requires more effort. The reasons is that tutorials on the stereo rendering functions is not well explain, thus making use of the SDK is not easy. Moreover, stereo GUI designing is a huge problem because no tutorials have been given and to understand the functioning codes require more time and higher level of expertise. However, the SDKs code can be tracked and identified to roughly know how the program function and modify the program accordingly.

5.5.4 Integrating Metaio SDK with Google Cardboard SDK

Integrating both SDKs together to make an application is a difficult task. Metaio SDK is mainly developed for augmented reality application while Google Cardboard SDK often used for virtual reality application. However, integrating both SDK is possible depending of the functions needed and both SDK should be integrated carefully so that they does not crash the program. Due to the reason that both SDK were built for different applications, thus, they have different designated functions such as GUI development. Both the virtual reality design requires OpenGLES 2.0 from the device, however both SDK handles it differently. Thus, it is not possible to unite both SDK to create one GUI. Thus, GUI development only can be done using one of the SDK.

5.5.5 Useful Debugging Tools

Two useful debugging tools have been used in this project development. The first debugging tools is breakpoint debugger. Setting breakpoint at certain programs or functions helps to identify and understand the working flow of the code. Besides, it also enable programmer to check on parameters and their values further help the programmer to verify the modification made and values passing are correct. Since this project is mainly modify and amend existing program code, breakpoint debugger is helpful up to certain point.

Next, the alternative debugging tools is logcat debugging. This is rather a faster debugging method where by the programmer put logging ability in each of the code function and subsequently allow the whole program to run freely. After that, logging message of each function code will display in the logcat monitor and the running code will be displayed sequentially. Whenever there is an error appeared, the logcat monitor will subsequently display in the monitor and programmer will able to identify which function went wrong and trace back the source of error. Using this method, error identification is rather faster than breakpoint debugger. Then, breakpoint debugger can be utilized in such a way to trace back the error source.

CHAPTER 6

CONCLUSION AND RECOMMENDATIONS

6.1 Conclusion

At the of this project development, an augmented reality head mounted display has been created using Google Cardboard and Metaio SDK as the fundamental program. Furthermore, two android applications consisting "Template" to perform the augmented reality controller function and "Android-Client" as the application run by the smart device being controlled have been developed and demonstrated in Chapter 4 and 5. In the same token, a simple control system has been implemented successfully using the proposed method and the applications worked reasonably fine. There are a lot of room of potentials for the applications to be further developed. Since Google Cardboard is low cost by nature and smartphone is considered a necessity to almost every household, this Google Cardboard based application has actually resolved one of the entry barrier to reach out the consumers in the market. Although this applications is still in developing process, it has its potential grow bigger with more control ability features.

6.2 Unresolved Issue and Product Limitation

Although this application has been developed successfully, there are still some issues that have yet to be solved. Besides, there are a lot of rooms for improvisation in this system.

Firstly, an alternative recognition system shall be used instead of ID marker. The reason is that, ID marker require clear view to the camera. If the camera cannot detect the full pattern of the ID marker, or border of the marker is corrupted, the virtual object will not able to be created on the screen. Furthermore, ID marker also occupy some spaces which it may affect the aesthetic appealing of the environment. Therefore, alternative recognition system such as marker-less tracking system, inertial measurement units (IMU) sensors, and so forth can be implemented to improve recognition system.

Next, GUI development to enrich user experiences has yet to be developed. For example, the smartphone should able to notify the user how many counts the user has triggered the magnet ring, so that the smartphone or controller able to send accurate signal to smart device. However, there are some complexity in stereo GUI development of Metaio SDK, this feature has yet to be developed.

Thirdly, the resolution of the smartphone screen is another issue to be encountered. LG Nexus 4 being used in this project has a display resolution of 768 x 1280 pixels (approximately 318 ppi pixel density). Thus, when the user view through the through Google Cardboard, the image perceived has slightly larger pixels resulting the image to be slightly blurred. On top of that, this consequence lead to uncomforted eye when the user use this application for a period of time.

6.3 Recommendation and Future Improvement

Firstly, this system also can be made to control more smart devices. For example, it may be designed to serve as a smart home system in augmented reality environment. As the augmented reality technology is growing, automated home system control is another emerging technology, both of this idea can be merged to produce a smart home and augmented reality control system.

Moreover, using wifi connectivity can be power consuming. Although wifi coverage is the large up to couples of meters, its power consumption is relatively high. Thus, smart devices that runs on battery may dries off easily. This is because the application will have to keep the smart device running so that it able to accept connections in background all the time even when it is in sleep mode. Thus in a long run, lesser power consumption wireless technologies such as Bluetooth or ZigBee will able to put smart device in a longer waiting period.

This application is still in developing stage, many of potential features can be added into this application. As discuss earlier, it maybe develop as a smart home augmented reality control system. Besides, it can be develop into a universal controller whereby it can control all the smart devices that have been registered into the system and perform actual control features. For instance, if it is going to control a smart TV, it can be made to control the TV channels, volume, switch among TV's applications and many more.

The input handling from the user can be further improvised too. The magnetic input from the cardboard magnet to the smartphone is inconsistent, making the input system to be unstable. Therefore, a hand gesture input method can be integrated to improve this system. A hand gesture input handling can be more interesting and user-friendly as compared to magnetic input. One of the reason is that, the magnetic input is monotonous and it requires complimentary support to perform diversify input method. For example, numbers of magnetic triggered can be used to perform different tasks. However, this can be solved using a hand gesture input system. An additional 3D camera such as Microsoft's Kinect and Intel Realsense can be integrated into the controller to handle hand gesture recognition easily. Hand gesture is more user friendly is such a way that tapping, waving and dedicated finger posture enable user to interact with virtual object accurately and effectively as compared to magnetic input.

REFERENCES

Adhani, N.I., Rambli, D.R.A., 2012. A survey of mobile augmented reality applications. 1st International Conference on Future Trends in Computing and Communication Technologies [e-journal], Available at: http://www.academia.edu/3061694/A_Survey_of_Mobile_Augmented_Reality_Applications [15 February 2015].

Amin, D., Govilkar, D., 2015, COMPARATIVE STUDY OF AUGMENTED REALITY
SDK'S. AIRCC [online], 5(1), pp. 11-25. Available at: http://airccse.org/journal/ijcsa/papers/5115ijcsa02.pdf [10 April 2015]

Aron, M., Simon, G., Berger, M.O., 2007. Use of inertial sensors to support video tracking. ACM [online], 18 (1), pp. 57– 68. Available at: http://dl.acm.org/citation.cfm?id=1229034 [6 April 2015].

Asad, M., 2009. iPhone in iPhone Augmented Reality App [online] Available at: http://www.redmondpie.com/iphone-in-iphone-augmented-reality-app-9140263/ [Accessed 10 April 2013]

Daponte, P., De Vito, L., Picariello, L., Riccio, M., 2014. State of the art and future developments of the Augmented Reality for measurement applications. Science Direct [online], 57, pp. 53-70. Available at: http://www.sciencedirect.com/science/article/pii/S0263224114003054 [27 March 2015]

Dev.metaio.com, 2015. Setting up the Development Environment | metaio Developer Portal. [online] Available at: http://dev.metaio.com/sdk/getting-started/android/setting-up-the-development-environment/index.html [Accessed 7 Sep. 2015].

Golan, L., 2006. Computer vision for artists and designers: pedagogic tools and techniques for novice programmers. Springer [e-journal], pp. 462-482. Available at: http://www.yorku.ca/caitlin/futurecinemas/resources/coursepack/readings/computervision.pdf [2 march 2015].

Hartmann, B., Link, N., Trommer, G.F., 2010. Indoor 3D Position Estimation Using Low-Cost Inertial Sensors and Marker-Based Video-Tracking. IEEE [online], pp. 319-326. Available at: http://ieeexplore.ieee.org/search/searchresult.jsp?newsearch=true&queryText=Indoor+3D+Position+Estimation+Using+Low-Cost+Inertial+Sensors+and+Marker-Based+Video-Tracking [3 April 2015].

Kjmagnetics.com, 2015. *K&J Magnetics Blog*. [online] Available at: https://www.kjmagnetics.com/blog.asp?p=google-cardboard [Accessed 6 Sep. 2015].

Lang, P., Kusej, A., Pinz, A., Brasseur, G., 2002. Inertial tracking for mobile augmented reality. IEEE [online], vol. 2, pp. 1583–1587. Available at: http://ieeexplore.ieee.org.libezp.utar.edu.my/xpl/articleDetails.jsp?tp=&arnumber=1007196&queryText%3D.+Inertial+tracking+for+mobile+augmented+reality [15 February 2015]

Lee, S., 2013. IKEA to use Augmented Reality for Perfect Furniture Planning [online] Available at: http://news.filehippo.com/2013/08/ikea-to-use-ar-for-perfect-furniture-planning/ [Accessed 10 April 2013]

Metaio.com, 2015. metaio | Product Support. [online] Available at: https://www.metaio.com/product_support.html [Accessed 5 Sep. 2015].

Rolland, J.P., Davis, L., Baillot, Y., 2001. A survey of tracking technology for virtual environments. [e-journal], pp. 67–112. Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.89.8135&rep=rep1&type=pdf [3 April 2015].

SMART, S. 2015. Dodocase VR Tutorial (Instructables Build Night). [online] Instructables.com. Available at: http://www.instructables.com/id/Dodocase-VR-Tutorial-Instructables-Build-Night/?ALLSTEPS [Accessed 6 Sep. 2015].

Stephan, G., Alexander, G., Lukas B., 2010. Server-side object recognition and client-side object tracking for mobile augmented reality. IEEE [online], pp. 1-8. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5543248&queryText%3DServer-side+object+recognition+and+client-side+object+tracking+for+mobile+augmented+reality [6 April 2015]

Sutherland, I.E., 1968. A head mounted three dimensional display. American Federation of Information Processing Societies (AFIPS) [online], pp. 757–764. Available at: http://dl.acm.org/citation.cfm?id=1476686 [2 March 2015].

Thomas, B.H., Close, B., Donoghue, J., Squires, J., De Bondi, P., Morris, M., Piekarski, W., 2000. ARQuake: an outdoor/indoor augmented reality first person application. IEEE [online], pp. 139-146. Available at: http://ieeexplore.ieee.org.libezp.utar.edu.my/xpl/articleDetails.jsp?tp=&arnumber=888480&queryText%3DARQuake%3A+an+outdoor%2Findoor+augmented+reality+first+person+application [2 March 2015].

Van Krevelen, D.W.F., Poelman, R., 2003. A survey of augmented reality technologies, applications and limitations. Scopus [online], 9(2), pp. 1-20. Available at: http://www.scopus.com/record/display.url?eid=2-s2.0-80052120343&origin=inward&txGid=7B000A91971A042FCBDE658CBBA64F46.kqQeWtawXauCyC8ghhRGJg%3a2 [9 February 2015]

Wang, J.-t, Shyi, C.-N., Hou, T.-W., Fong, C.P., 2010. Design and Implementation of Augmented Reality System Collaborating with QR Code. IEEE [online], pp.414-418. Available at: http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=5685477&url=http%3A%2F%2Fieeexplore.ieee.org%2Fxpls%2Fabs_all.jsp%3Farnumber%3D5685477 [3 April 2015]

Yoo, S., Parker, C., 2015. Controller-less Interaction Methods for Google Cardboard. ACM [online], pp. 127 Available at: http://delivery.acm.org.libezp.utar.edu.my/10.1145/2800000/2794359/p127-yoo.pdf?ip=58.27.19.239&id=2794359&acc=OPEN&key=69AF3716A20387ED%2ECB81A03442DECDAF%2E4D4702B0C3E38B35%2E6D218144511F3437&CFID=543445029&CFTOKEN=62197189&__acm__=1441520837_2566d821fd35c241ffd7d625db942034 [Accessed 6 Sep. 2015]

Zhang, Q., Chu, W., Ji, C., Ke, C., Li, Y., 2014. An Implementation of Generic Augmented Reality in
Mobile Devices. IEEE [online], pp. 555-558. Available at: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=7065112&queryText%3DAn+Implementation+of+Generic+Augmented+Reality+in+Mobile+Devices [21 March 2015]

APPENDICES

APPENDIX A: Computer Programme Listing

Template.java

```
// Copyright 2007-2014 Metaio GmbH. All rights reserved.
package com.metaio.Template;

import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.io.BufferedReader;
import java.net.UnknownHostException;
import java.lang.String;

import android.content.Context;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.CountDownTimer;
import android.os.Vibrator;
import android.util.Log;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;
import android.os.Handler;
import android.os.Message;
import android.view.SurfaceHolder;
import android.view.SurfaceView;
import android.view.Display;
import android.graphics.Point;

import com.google.vrtoolkit.cardboard.sensors.MagnetSensor;
import com.metaio.sdk.ARViewActivity;
import com.metaio.sdk.CameraView;
import com.metaio.sdk.MetaioDebug;
import com.metaio.sdk.jni.EVISUAL_SEARCH_STATE;
import com.metaio.sdk.jni.IGeometry;
import com.metaio.sdk.jni.IGeometryVector;
import com.metaio.sdk.jni.IMetaioSDKCallback;
import com.metaio.sdk.jni.IVisualSearchCallback;
import com.metaio.sdk.jni.ImageStruct;
import com.metaio.sdk.jni.MetaioSDK;
import com.metaio.sdk.jni.Rotation;
import com.metaio.sdk.jni.TrackingValues;
import com.metaio.sdk.jni.TrackingValuesVector;
import com.metaio.sdk.jni.Vector3d;
import com.metaio.sdk.jni.VisualSearchResponseVector;
import com.metaio.tools.io.AssetsManager;

public class Template extends ARViewActivity implements
MagnetSensor.OnCardboardTriggerListener
{

   private IGeometry mEarth;
   private IGeometry mEarthOcclusion;
   private IGeometry mEarthIndicators;
   private boolean mEarthOpened;
   private MetaioSDKCallbackHandler mSDKCallback;
   private VisualSearchCallbackHandler mVisualSearchCallback;

   private static SurfaceView surfaceView;
   private TextView dsp_message;

   private String host_ip;
   private Socket client;
   private PrintWriter printWriter;
   private Vibrator vibrator;

   private MagnetSensor mMagnetSensor;
```

Template.java

```
   private int screenCenterX;
   private int screenCenterY;
   private int i;
   private int j;
   private boolean times_up = true;
   private boolean video_played;

   Handler msg_handler = new Handler(){
      @Override
      public void handleMessage(Message msg) {
         dsp_message = (TextView)findViewById(R.id.dsp_msg1);
         //dsp_message.setText(msg);
      }
   };

   @Override
   public void onCreate(Bundle savedInstanceState)
   {
      super.onCreate(savedInstanceState);
      mEarthOpened = false;
      metaioSDK.setStereoRendering(true);

      mSDKCallback = new MetaioSDKCallbackHandler();
      mVisualSearchCallback = new VisualSearchCallbackHandler();

      surfaceView = (SurfaceView)findViewById(R.id.surfaceView);

      this.mMagnetSensor = new MagnetSensor(this);      //implement magnet
sensor to this function
      this.mMagnetSensor.setOnCardboardTriggerListener(this);    //establish
magnet trigger listener
      mMagnetSensor.start();                              //start magnet sensing
activity

      //--------get Display size of smartphone to obtain center point of
screen---------------
      Display display = getWindowManager().getDefaultDisplay();
      Point size = new Point();
      display.getSize(size);
      screenCenterX = (size.x /2);
      screenCenterY = (size.y/2) ;
      i = 0;
      video_played=false;
      if (metaioSDK != null)
      {
         metaioSDK.registerVisualSearchCallback(mVisualSearchCallback);
      }
   }

//--------additional button for debugging purpose---------------------------
-------------------
   public void start_pressed(View v)throws IOException{
      TextView InputView = (TextView)findViewById(R.id.input_view);
      InputView.setVisibility(View.VISIBLE);
      EditText Host_ip = (EditText) findViewById(R.id.input_ip);
      Host_ip.setVisibility(View.VISIBLE);
      vibrator = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
      vibrator.vibrate(50);
      };


//-------Called when the Cardboard trigger is pulled.-----------------------
------------------
   @Override
   public void onCardboardTrigger(){
      i++;
```

Template.java

```
      EditText Host_IP = (EditText) findViewById(R.id.input_ip);
      host_ip = Host_IP.getText().toString();        //obtain IP address 
entered by the user

      switch (i) {
         case 1:{
            IGeometry geometry = 
metaioSDK.getGeometryFromViewportCoordinates(screenCenterX, screenCenterY, 
true);
         if (!video_played && times_up && (geometry != null)) {

            Count_time count_time = new Count_time();
            count_time.doInBackground();    // run timer to accumulate 
triggers from user

            //determine if the virtual object is hit by the user------------
-------------------------
            //IGeometry geometry = 
metaioSDK.getGeometryFromViewportCoordinates(screenCenterX, screenCenterY, 
true);
            if (geometry != null) {
               onGeometryTouched(geometry); //geometry touched
            }
         }
            else if (video_played){
            if (times_up) {

               Count_time count_time = new Count_time();
               count_time.doInBackground();    // run timer to accumulate 
triggers from user

               //determine if the virtual object is hit by the user---------
----------------------------
               //IGeometry geometry = 
metaioSDK.getGeometryFromViewportCoordinates(screenCenterX, screenCenterY, 
true);
               if (geometry != null) {
                  onGeometryTouched(geometry); //geometry touched
               }
            }
         }
            else{
            i=0;
         }
            break;}
         default:{
            if (video_played){
            if (times_up) {

               Count_time count_time = new Count_time();
               count_time.doInBackground();    // run timer to accumulate 
triggers from user

               //determine if the virtual object is hit by the user---------
----------------------------
               IGeometry geometry = 
metaioSDK.getGeometryFromViewportCoordinates(screenCenterX, screenCenterY, 
true);
               if (geometry != null) {
                  onGeometryTouched(geometry); //geometry touched

                  }
            }
         }
            else{
               i=0;
```

Template.java

```
            }
            break;
         }
      }
      vibrator = (Vibrator) this.getSystemService(Context.VIBRATOR_SERVICE);
      vibrator.vibrate(50);
   }

//---------Start timer before send signal to outputstream-------------------
---------------------
   private class Count_time extends AsyncTask<Void, Void, Void> {

      @Override
      protected Void doInBackground(Void... params) {
         try {
            times_up = false;
            final Handler handler = new Handler();
            handler.postDelayed(new Runnable() {
               @Override
               public void run() {
                  j=i;
                  i=0;
                  SendMessage sendMessageTask = new SendMessage();
                  sendMessageTask.execute(); //execute connection thread
                  times_up = true;
               }
            },2000);   //wait for two second before execute connection
thread

         } catch (Exception e) {
            e.printStackTrace();
            times_up = true;  //two second has over write true to parameter
         } return null;
      }
   }

//----Establish Connection to smart device and send signal------------------
-----------------------
   private class SendMessage extends AsyncTask<Void, Void, Void> {

   @Override
   protected Void doInBackground(Void...params) {

      if (host_ip.isEmpty()) {
         host_ip = "172.28.44.3";
      }

      switch (j) {
         case 1: {
            video_played = true;
            break;
         }
         case 2: {
            video_played = false;
            break;
         }
         default: {
            break;
         }
      }
            try {
               client = new Socket(host_ip, 8888);   //Connect to host with
port 8888
               printWriter = new PrintWriter(client.getOutputStream(), true);
//establish outputstream
               String out = String.valueOf(j);        //get signal from the
```

```
user
               printWriter.write(out);           //send signal to determine
should the smart device do
               printWriter.flush();            //empty printWriter
               printWriter.close();            //close printWriter
               client.close();                    //close socket
               j = 0;
            } catch (UnknownHostException e) {
               e.printStackTrace();
            } catch (IOException e) {
               e.printStackTrace();
            }return null;
      }
   }


//------when application is closed------------------------------------------
------------------------
   @Override
   protected void onDestroy()
   {
      super.onDestroy();
      mSDKCallback.delete();
      mSDKCallback = null;
      mVisualSearchCallback.delete();
      mVisualSearchCallback = null;
      mMagnetSensor.stop();
   }

//----- Attaching layout to the activity------------------------------------
------------------------
   @Override
   protected int getGUILayout()
   {

      return R.layout.template;
   }

//------------------load virtual Object-------------------------------------
-----------------------------------------------------
   @Override
   protected void loadContents()
   {
      try
      {
         // Getting a file path for tracking configuration XML file
         final File trackingConfigFile =
AssetsManager.getAssetPathAsFile(getApplicationContext(),
"TrackingData_Marker.xml");

         // Assigning tracking configuration
         boolean result =
metaioSDK.setTrackingConfiguration(trackingConfigFile);
         MetaioDebug.log("Tracking data loaded: " + result);

         final float scale = 3.f;   //assign virtual object scale
         final Rotation rotation = new Rotation(new Vector3d(90.0f,
0.0f,0.0f));    //assign virtual object's orientation
//(float)Math.PI/2, 90.0f, 90.0f));

         // Getting a file path for a 3D geometry
         final File earthModel =
AssetsManager.getAssetPathAsFile(getApplicationContext(), "Earth.zip");
         if (earthModel != null)
         {
            // Loading 3D geometry
```

Template.java

```
            mEarth = metaioSDK.createGeometry(earthModel);
            if (mEarth != null)
            {
               // Set geometry properties
               mEarth.setScale(scale);
               mEarth.setRotation(rotation);
            }
            else
               MetaioDebug.log(Log.ERROR, "Error loading earth geometry: " +
mEarth);
         }

         //------get 3D model from path---------
         final File earthOcclusionModel =
AssetsManager.getAssetPathAsFile(getApplicationContext(),
"Earth_Occlusion.zip");
         if (earthOcclusionModel != null)
         {
            mEarthOcclusion = metaioSDK.createGeometry(earthOcclusionModel);
//create virtual object
            if (mEarthOcclusion != null)
            {
               mEarthOcclusion.setScale(scale);
               mEarthOcclusion.setRotation(rotation);
               mEarthOcclusion.setOcclusionMode(true);
            }
         }
         else
            MetaioDebug.log(Log.ERROR, "Error loading earth occlusion
geometry: " + mEarthOcclusion);

         //---------get 3D model from path(needed when virtual object is
touched)----------------
         final File earthIndicatorsModel =
AssetsManager.getAssetPathAsFile(getApplicationContext(),
"EarthIndicators.zip");
         if (earthIndicatorsModel != null)
         {
            mEarthIndicators =
metaioSDK.createGeometry(earthIndicatorsModel);
            if (mEarthIndicators != null)
            {
               mEarthIndicators.setScale(scale);
               mEarthIndicators.setRotation(rotation);
            }
            else
               MetaioDebug.log(Log.ERROR, "Error loading earth indicator
geometry: " + mEarthIndicators);
         }
      }
      catch (Exception e)
      {
         MetaioDebug.log(Log.ERROR, "Failed to load content: " + e);
      }
   }

//-------------Animate Virtual object when it is being touch----------------
--------------------
   @Override
   protected void onGeometryTouched(IGeometry geometry)
   {
      //hide IP address input of the GUI
      TextView InputView = (TextView)findViewById(R.id.input_view);
      InputView.setVisibility(View.INVISIBLE);
      EditText Host_ip = (EditText) findViewById(R.id.input_ip);
      Host_ip.setVisibility(View.INVISIBLE);
```

Template.java

```
      MetaioDebug.log("Template.onGeometryTouched: " + geometry);

      if (geometry != mEarthOcclusion) //check if the virtual object touch 
is the earth
      {
         if (!mEarthOpened)  //open the virtual object if it is not open
         {
            mEarth.startAnimation("Open", false);
            mEarthIndicators.startAnimation("Grow", false);
            mEarthOpened = true;
         }
         else  //close the virtual object
         {
            mEarth.startAnimation("Close", false);
            mEarthIndicators.startAnimation("Shrink", false);
            mEarthOpened = false;
         }
      }
   }

   @Override
   protected IMetaioSDKCallback getMetaioSDKCallbackHandler()
   {
      return mSDKCallback;
   }

   final class MetaioSDKCallbackHandler extends IMetaioSDKCallback
   {

      @Override
      public void onSDKReady()
      {
         MetaioDebug.log("The SDK is ready");
      }

      @Override
      public void onAnimationEnd(IGeometry geometry, String animationName)
      {
         MetaioDebug.log("animation ended" + animationName);
      }

      @Override
      public void onMovieEnd(IGeometry geometry, File filePath)
      {
         MetaioDebug.log("movie ended" + filePath.getPath());
      }

      @Override
      public void onNewCameraFrame(ImageStruct cameraFrame)
      {
         MetaioDebug.log("a new camera frame image is delivered" + 
cameraFrame.getTimestamp());
      }

      @Override
      public void onCameraImageSaved(File filePath)
      {
         MetaioDebug.log("a new camera frame image is saved to" + 
filePath.getPath());
      }

      @Override
      public void onScreenshotImage(ImageStruct image)
      {
         MetaioDebug.log("screenshot image is received" + 
image.getTimestamp());
```

Template.java

```
      }

      @Override
      public void onScreenshotSaved(File filePath)
      {
         MetaioDebug.log("screenshot image is saved to" +
filePath.getPath());
      }

      @Override
      public void onTrackingEvent(TrackingValuesVector trackingValues)
      {
         for (int i=0; i<trackingValues.size(); i++)
         {
            final TrackingValues v = trackingValues.get(i);
            MetaioDebug.log("Tracking state for COS " +
v.getCoordinateSystemID()+" is "+v.getState());

         }

      }

      @Override
      public void onInstantTrackingEvent(boolean success, File filePath)
      {
         if (success)
         {
            MetaioDebug.log("Instant 3D tracking is successful");
         }
      }
   }

   final class VisualSearchCallbackHandler extends IVisualSearchCallback
   {
      @Override
      public void onVisualSearchResult(VisualSearchResponseVector response,
int errorCode)
      {
         if (errorCode == 0)
         {
            MetaioDebug.log("Visual search is successful");
         }
      }

      @Override
      public void onVisualSearchStatusChanged(EVISUAL_SEARCH_STATE state)
      {
         MetaioDebug.log("The current visual search state is: " + state);
      }
   }
}
```

MainActivity.java

```
package com.fyp_client.jxiang.android_client;

import android.app.ProgressDialog;
import android.net.Uri;
import android.os.Handler;
import android.os.Message;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;

import android.os.PowerManager;
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.DataInputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

import java.io.PrintWriter;
import java.lang.String;
import java.net.UnknownHostException;

import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.VideoView;
import android.widget.ViewSwitcher;
import android.media.MediaPlayer;
import android.view.WindowManager;
import android.widget.MediaController;

public class MainActivity extends AppCompatActivity
{

    EditText textOut;
    private Integer input_message;
    private ImageView mImageView;
    private static ServerSocket serverSocket;
    private static Socket clientSocket;
    private static InputStreamReader inputStreamReader;
    private static BufferedReader bufferedReader;

    public VideoView myVideoView;
    private int position;
    private ProgressDialog progressDialog;
    private MediaController mediaController;
    public boolean video_played;
    private ViewSwitcher switcher;
    private static final int NEXT_SCREEN = 1;
    private static final int PREVIOUS_SCREEN = 2;

//----------------App status handler---------------------------------------
    Handler msg_handler = new Handler(){

        @Override
        public void handleMessage(Message msg) {
           TextView status_MSG = (TextView)findViewById(R.id.status_msg);
            status_MSG.setText("listening to client...");
        }
    };
```

MainActivity.java

```
//------------Switch function of Layout Handler----------------------------
    Handler Refresh = new Handler(){
        public void handleMessage(Message msg) {

            switch(msg.what){

                case NEXT_SCREEN:

getWindow().addFlags(WindowManager.LayoutParams.FLAG_DISMISS_KEYGUARD |
WindowManager.LayoutParams.FLAG_FULLSCREEN |
WindowManager.LayoutParams.FLAG_SHOW_WHEN_LOCKED |
WindowManager.LayoutParams.FLAG_TURN_SCREEN_ON);
                    switcher.showNext(); //go to the next layout

                    break;
                case PREVIOUS_SCREEN:
                    switcher.showPrevious(); //go back to the previous
layout
                    break;
                default:
                    break;
            }
        }
    };

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        video_played = false;
        textOut = (EditText)findViewById(R.id.textout);
        Button buttonSend = (Button)findViewById(R.id.send);
        buttonSend.setOnClickListener(buttonSendOnClickListener);
        switcher = (ViewSwitcher) findViewById(R.id.profileSwitcher);
        mediaController = new MediaController(this);
    }

    //-----------------Button Listener to start connection------------------
------------------------
    Button.OnClickListener buttonSendOnClickListener = new
Button.OnClickListener() {

        public void onClick(View arg0) {
            Thread start_Server = new Thread(new Start_Server());
            start_Server.start();
            mImageView = (ImageView)findViewById(R.id.ID_image);
            mImageView.setImageResource(R.drawable.arcs_id_marker);
        }
    };

    //---------Establish socket and waiting for connection from controller--
------------------------
    private class Start_Server extends Thread{

        public void run() {
            try {
                serverSocket = new ServerSocket(8888);
            } catch (IOException e) {
                e.printStackTrace();
            }
            msg_handler.sendEmptyMessage(0);

            while (true) {  //endless loop to receive client connection
request continuously
                try {
                    clientSocket = serverSocket.accept();    //accept
```

MainActivity.java

```
 connection request from client
                    inputStreamReader = new
InputStreamReader(clientSocket.getInputStream()); //obtain client's signal
                    bufferedReader = new BufferedReader(inputStreamReader);
//store signal in BufferedReader
                    String input_String = bufferedReader.readLine();
//assign parameter to the signal
                    input_message = Integer.parseInt(input_String);
//convert string signal into integer
                    VideoActivity videoActivity = new VideoActivity();
                    videoActivity.run();        //run VideoActivity to
handle the signal
                    inputStreamReader.close();  //close InputStreamReader
                    clientSocket.close();       //close ServerScoket
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
    }

    //-------------Video activity thread---------------------------------
    private class VideoActivity implements Runnable{
        @Override
        public void run() {
            try{
                switch (input_message) {

                    case 1:
                    if (!video_played) {
                        video_played = true;
                        myVideoView = (VideoView)
findViewById(R.id.video_view);
                        try {
                            myVideoView.setMediaController(mediaController);

myVideoView.setVideoURI(Uri.parse("android.resource://" + getPackageName() +
"/" + R.raw.deadpool_trailer));

                        } catch (Exception e) {
                            Log.e("Error", e.getMessage());
                            e.printStackTrace();
                        }
                        myVideoView.start(); // start video
                        Refresh.sendEmptyMessage(NEXT_SCREEN); //send signal
to handler to change GUI
                        break;
                    } else {
                        if (myVideoView.isPlaying()) {     //check if the
video is currently playing
                            position = myVideoView.getCurrentPosition();
//save current frame of video
                            myVideoView.pause(); //pause video
                        } else {
                            myVideoView.seekTo(position);  //go to the frame
saved when video is paused
                            myVideoView.start();    //resume the video
                        }
                        break;
                    }

                    case 2:{
                        if(video_played) {
                            myVideoView.stopPlayback(); //stop the video
                            video_played = false;   //reset parameter
                            Refresh.sendEmptyMessage(PREVIOUS_SCREEN);
```

MainActivity.java

```
//send signal to handler to change GUI
                            break;
                        }
                        else
                        {break;}
                    }

                    case 3:{    //video step forward
                        if(video_played) {
                            position = myVideoView.getCurrentPosition();
//get currrent video frame
                            myVideoView.seekTo(position + 8000);    //step
forward the video by 8 seconds
                            myVideoView.start();
                            break;
                        }
                        else
                        {break;}
                    }

                    case 4:{ //video step backward
                        if (video_played) {
                            position = myVideoView.getCurrentPosition();
//get currrent video frame
                            myVideoView.seekTo(position - 8000);     //step
backward the video by 8 seconds
                            myVideoView.start();
                            break;
                        }
                        else
                        {break;}

                    }
                    default:    //exit right away for undefined signals
received
                        break;
                }
            }catch (Exception e){
                e.printStackTrace();
            }

            myVideoView.setOnCompletionListener(new
MediaPlayer.OnCompletionListener(){
                @Override
                public void onCompletion(MediaPlayer mediaPlayer){
                    myVideoView.stopPlayback(); //stop the video
                    video_played = false;   //reset parameter
                    Refresh.sendEmptyMessage(PREVIOUS_SCREEN);
                }
            });
        }
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        // Inflate the menu; this adds items to the action bar if it is
present.
        getMenuInflater().inflate(R.menu.menu_main, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        // Handle action bar item clicks here. The action bar will
        // automatically handle clicks on the Home/Up button, so long
        // as you specify a parent activity in AndroidManifest.xml.
```

MainActivity.java

```java
        int id = item.getItemId();

        //noinspection SimplifiableIfStatement
        if (id == R.id.action_settings) {
            return true;
        }

        return super.onOptionsItemSelected(item);
    }

}
```