ANDROID AUGMENTED REALITY APPLICATION: SDSU UNIVERSITY CAMPUS GUIDE

A Thesis
Presented to the
Faculty of
San Diego State University

In Partial Fulfillment
of the Requirements for the Degree
Master of Science
in
Computer Science

by
Kavya Bangalore Nagaraja
Spring 2015
SAN DIEGO STATE UNIVERSITY

The Undersigned Faculty Committee Approves the
Thesis of Kavya Bangalore Nagaraja:

Android Augmented Reality Application: SDSU University Campus Guide

______________________________
Carl Eckberg, Chair
Department of Computer Science

______________________________
William Root
Department of Computer Science

______________________________
Robert Grone
Department of Mathematics and Statistics

March 21, 2015
Approval Date
DEDICATION

There are number of people without whom this thesis might not have been written, and to whom I am greatly indebted.

This dissertation is dedicated to my parents, Vanaja and Nagaraj, who have always loved me unconditionally and whose good examples have taught me to work hard for the things that I aspire to achieve.

This work is also greatly dedicated to my husband, Hariprasad, who has been a constant source of support and encouragement during the challenges of graduate school and life. I am truly thankful for having you in my life.

I also dedicate this work to my mother-in-law and my brother for their support, encouragement and comfort, who are forever appreciated. It's to the two of you I credit giving me a competitive spirit.
This project aims at developing an android augmented reality application that would have the capability to show university campus related information such as library, faculty and courses offered from a particular department. All this information is available by getting sensor data from your android device camera and overlaying images in real-time.

Augmented Reality (AR) is a generic term for an interactive 3D environment that blends with our physical reality, usually through a webcam, or in this case, an android device camera. AR by definition is a live, direct or indirect, view of a physical, real world environment whose elements are augmented by computer-generated sensory input such as sound, video graphics or GPS data.

The “SDSU University Campus Guide” mobile application is built on by taking pictures and videos of a particular building within a university campus and creating a sensible presentation (by stitching all pictures). Where a user focuses his/her android device camera on to a particular image of a live building, the information related to that particular department will be displayed, after “recognizing” that building from the archived pictures.

This application helps University students to get information about events, faculty, department or particular department related courses by just one click on this app.

This AR app uses Vuforia as a software platform and JAVA as a programming language which provides superior vision based image recognition and offers the widest set of features and capabilities to improve the University campus tour guide for the students to get to know their University better and easier. The application has been prototyped of a subset of campus buildings.
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABSTRACT ....................................................................................................................... v</td>
</tr>
<tr>
<td>LIST OF TABLES ............................................................................................................... viii</td>
</tr>
<tr>
<td>LIST OF FIGURES ........................................................................................................... ix</td>
</tr>
<tr>
<td>ACKNOWLEDGEMENTS ....................................................................................................... xi</td>
</tr>
<tr>
<td>CHAPTER</td>
</tr>
<tr>
<td>1 INTRODUCTION ............................................................................................................ 1</td>
</tr>
<tr>
<td>1.1 Overview ..................................................................................................................... 1</td>
</tr>
<tr>
<td>1.2 Motivation ................................................................................................................... 1</td>
</tr>
<tr>
<td>1.3 About Application ...................................................................................................... 3</td>
</tr>
<tr>
<td>1.4 Document Organization ............................................................................................. 3</td>
</tr>
<tr>
<td>2 SOFTWARE PLATFORMS AND DEBUGGING TOOLS ..................................................... 4</td>
</tr>
<tr>
<td>2.1 Qualcomm Vuforia Augmented Reality Framework ................................................. 4</td>
</tr>
<tr>
<td>2.1.1 Vuforia Frame-Work ............................................................................................... 4</td>
</tr>
<tr>
<td>2.1.2 Vuforia Interoperability ......................................................................................... 6</td>
</tr>
<tr>
<td>2.1.3 AR Frame Work ...................................................................................................... 7</td>
</tr>
<tr>
<td>2.1.4 Vuforia Applications ............................................................................................. 8</td>
</tr>
<tr>
<td>2.1.5 Other Available AR Development SDKs in the Market ....................................... 8</td>
</tr>
<tr>
<td>2.1.6 Metaio vs. Vuforia .................................................................................................. 8</td>
</tr>
<tr>
<td>2.2 Android Platform ...................................................................................................... 8</td>
</tr>
<tr>
<td>2.2.1 Android Framework ............................................................................................... 10</td>
</tr>
<tr>
<td>2.2.2 Advantages of Android Platform ........................................................................... 11</td>
</tr>
<tr>
<td>2.3 Java .......................................................................................................................... 11</td>
</tr>
<tr>
<td>2.3.1 Object-Oriented Features of Java .......................................................................... 13</td>
</tr>
<tr>
<td>2.3.2 JAVA and Augmented Reality (AR) ................................................................. 14</td>
</tr>
<tr>
<td>2.4 Eclipse IDE ............................................................................................................... 14</td>
</tr>
<tr>
<td>2.4.1 Java Development Tools (JDT) ............................................................................ 14</td>
</tr>
<tr>
<td>2.4.2 Plug-In Development Environment (PDE) .................................................... 15</td>
</tr>
</tbody>
</table>
2.4.3 Advantages of Using Eclipse .................................................................15
2.5 XML & HTML ..........................................................................................15
3 SOFTWARE PRODUCT DEVELOPMENT LIFECYCLE .................................17
4 SYSTEM REQUIREMENTS ............................................................................19
5 SETTING UP THE DEVELOPMENT ENVIRONMENT .....................................20
  5.1 Creating Device Database-Set on Vuforia ..............................................20
  5.2 Setting Up the Android Development Environment ...............................21
    5.2.1 JDK (Java Development Tool Kit) Installation .................................22
    5.2.2 Android ADT (Android Development Tools) Bundle Installation ......22
    5.2.3 Setting Up Android SDK Manager ....................................................22
    5.2.4 Setting Up Cygwin Environment ......................................................23
    5.2.5 Setting Up Android NDK (Native Development Tool Kit) ..................24
  5.3 Installing Vuforia SDK ............................................................................24
6 CREATING AN ANDROID APPLICATION USING ECLIPSE IDE ...............26
7 HOW TO RUN AN ANDROID APPLICATION ..............................................33
8 FUTURE ENHANCEMENTS ........................................................................41
9 SUMMARY AND DEVELOPMENT CHALLENGES ......................................42
REFERENCES .................................................................................................43
APPENDICES
  A ABOUTSCREEN.JAVA ..................................................................................45
  B IMAGETARGETS.JAVA ................................................................................47
  C IMAGETARGETSRENDERER.JAVA ..............................................................52
  D IMAGETARGETSSPLASHSCREEN.JAVA .......................................................55
LIST OF TABLES

Table 2.1. Comparison between Metaio and Vuforia AR SDKs.........................................................9
LIST OF FIGURES

Figure 1.1. Sacred Land film project. ................................................................. 2
Figure 1.2. Rio de Janeiro - HD - Brazil - tourist guide. ............................. 2
Figure 2.1. High-level system overview of Vuforia SDK. ........................... 5
Figure 2.2. Block diagram representation of interoperability process in Augmented Reality between provider and user............................ 6
Figure 2.3. Augmented Reality Framework. .................................................... 7
Figure 2.4. Hierarchical view of the android framework.............................. 10
Figure 2.5. Java execution framework- JVM.................................................. 12
Figure 2.6. JVM components. ................................................................. 12
Figure 2.7. Generic framework of AR on the web. ....................................... 16
Figure 3.1. Software Development Life Cycle (SDLC). ............................... 18
Figure 5.1. Vuforia Target manager. ...................................................... 21
Figure 5.2. Android SDK manager. ...................................................... 23
Figure 6.1. Eclipse Workspace launcher.................................................... 27
Figure 6.2. Selecting new android application project. ............................... 27
Figure 6.3. New Application window_1.................................................... 28
Figure 6.4. New Application window_2.................................................... 28
Figure 6.5. New application window_3 .................................................... 29
Figure 6.6. New application window_3  Select blank activity for an android application............................................................................ 29
Figure 6.7. New application window_3  Enter the activity and layout name..... 30
Figure 6.8. ImageTargetsApp package structure. ....................................... 30
Figure 6.9. Represents how to create a new .java file under SRC folder. ......... 31
Figure 6.10. Class Hierarchy in Android application project. ....................... 32
Figure 7.1. Running ImageTargetsApp android application_1....................... 34
Figure 7.2. Running ImageTargetsApp android application_1 – Android device chooser.............................................................................. 35
Figure 7.3. UniversityInfo app has been installed. ........................................ 35
Figure 7.4. UniversityInfo application splash screen.................................................................36
Figure 7.5. Focusing android device camera over the university image on a laptop screen. ........................................................................................................................................37
Figure 7.6. Option page appears after the image is recognized by the camera.........................38
Figure 7.7. Options Screen_1......................................................................................................38
Figure 7.8. Main OptionPage......................................................................................................39
Figure 7.9. ComputerScienceDept OptionPage.........................................................................39
Figure 7.10. Computer Science FacultyInfo InfoPage.................................................................40
Figure 7.11. FacultyInfo page.....................................................................................................40
ACKNOWLEDGEMENTS

I would like to express my special immeasurable appreciation and deepest gratitude to my advisor Professor Dr. Carl F. Eckberg, you have been a tremendous mentor for me. I owe great regards and respect to my advisor for his support, advice, guidance, valuable comments, suggestions and provisions that benefited me much in the completion and success of this work.

I would also like to thank my committee members, Professor William A. Root and Professor Dr. Robert D. Grone, for serving as my committee members even at hardship. I also want to thank you for letting my defense be a great moment, and for your brilliant comments and suggestions, thanks to you.

Professors Eckberg, Root and Grone are people you will never forget once you meet them. They are the most adorable advisors and are among the smartest people I know. I hope that I could be as lively, enthusiastic, and energetic as them.
CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Android devices are making people’s lives easier and convenient with the blend of hardware and software tools. Augmented reality is one of the android platform features that combine virtual world with real world entities. This self-guided University tour application allows students to get to know about their department, faculty and course related information through a single camera focus over their respective department building i.e., if any student want to know about the computer science department information, then he/she only have to focus their android device camera over the computer science department building or over an image of that building on your computer or phone screen, that’s it!! All department related information such as, department contact information, its hours of operation, faculty and their hours of availability, courses offered by the department and events that have been organized will pop up on the android device screen provided with buttons to access particular information in more easier and faster way than ever. This app doesn’t need the internet to access any desired information. Other potential users of this application would be student parents or other student visitors, and prospective students

1.2 MOTIVATION

We can find many similar android applications that provide such facilities to the tourists for self-guided tours, but usage is very limited. We can find advanced level tourist guide apps along with location based services that help to find nearby attractions.

The Figure 1.1 [1] and Figure 1.2 [2] are examples of self-guided tourist guide android Augmented Reality apps. While doing some research on these kinds of products, I thought of designing a similar app for my University campus, to help students to get their desired information on their finger tips. Every time inside or outside the campus if I have to get any information about the campus map and about the professor with respect to their hours
of operation, it was very difficult to find it through an online department website. But now this app makes it easier without the need of internet connection and no need of peeking through different internal website links.

Some people can aboard a campus map (or shopping mall map) quickly, but many cannot. Being told you are looking at the physics building, even when facing an unlabeled side, can be helpful. The smart phone knowledge of where it is, plus its orientation can allow an app like this to say: directly behind you is the business building. The alternative of using a full campus map, requiring a live internet connection can be less helpful.
1.3 ABOUT APPLICATION

This application uses JAVA as a programming language and Qualcomm Vuforia as a software platform. This application uses Eclipse IDE as a debugger for this application. Along with the Java SDK, this app requires Android NDK (Native Development Kit), ADB (Android Debug Bridge) debugger, ADT (Android Development Tools) and AVD (Android Virtual Devices) manager tools. HTML is used as a web technology language for front end appearance. This application needs a camera-equipped android mobile or tablet devices.

1.4 DOCUMENT ORGANIZATION

- Chapter 2 discusses the technologies used in this project.
- Chapter 3 discusses the development phases that every software product goes through.
- Chapter 4 discusses the hardware and software requirements need to develop this application.
- Chapter 5 details the setting up of software packages that is used in this applications.
- Chapter 6 and Chapter 7 explains how to create and run android applications.
- Chapter 8 discusses the future enhancements that can be added to this application.
- Chapter 9 includes the final conclusion part.
CHAPTER 2
SOFTWARE PLATFORMS AND DEBUGGING TOOLS

This chapter includes the detailed information of the software platforms, programming language, debugging tool and the development tool kits used in designing this android application. This also includes the system requirements information.

2.1 QUALCOMM VUFORIA AUGMENTED REALITY FRAMEWORK

The Vuforia platform is a product of Qualcomm Technologies, Inc. This is one of the best and most creative branded AR software platforms that use technically efficient and superior quality of computer vision-based image recognition technology in the devices to connect the physical world with the digital environment. It is basically a Software Development Kit (SDK) that provides tools to design Augmented Reality applications.

Vuforia SDK provides the following features:

- Callbacks for events (Example: A new camera image is available)
- High-level access to hardware units (Example: Camera start/stop)
- Multiple trackables (tracking types):
  - Image Targets
  - Multi Targets
  - Cylinder Targets
  - Word Targets
  - Frame Markers
- Real-world Interactions
  - Virtual Buttons

2.1.1 Vuforia Frame-Work

Figure 2.1[3] represents the high-level framework of Vuforia SDK.
The key components of AR [4]:

- Camera Data
- Location Data
- Sensor Data
- Graphics Overlay

Camera data is the live feed display data from the android camera. This data can be available by using android APIs within the camera package i.e., under `android.hardware.Camera`.

Similarly, few android applications requires the mobile device’s current location data, based on which the related information will be displayed. These are achievable through android location APIs under the `android.location` package and with the help of the `LocationManager` class.

Sensor data is one of the major key components of AR applications. When the android device camera is used to get the focus of a particular image or an object, then the important factor to be considered would be the orientation of the android device. So, these sensors related APIs are provided under the `android.hardware.SensorManager` package. This package offers several APIs that allow the user to move the device as desired and see the changes.

So, bringing all the data together forms a Graphics Overlay of drawing something over the camera feed.
2.1.2 Vuforia Interoperability

Figure 2.2 [5] represents the block diagram representation of the AR systems Interoperability process between AR provider and AR user agent. If AR applications are going to be deployed on a massive scale, there are several key areas of technology that are needed [6]:

1. A low cost platform that combines AR display, tracking and processing
2. Mobility as key requirement for the platform to realize AR in a global space
3. Backend infrastructure for distributing of AR content and applications
4. Easy to use authoring tools for creating AR content
5. Large scale AR tracking solutions which work in real time.


Interoperability issues are one of the major challenges in AR. This includes:

1. Composite AR Contents Markup & Format
2. AR contents Transport/Interaction method
3. Representing 3D Interactive AR/MR Contents
4. Event Scripting
5. Local Caching method
6. Additional function ability of AR User Agent
7. Device Capability Access APIs
8. AR Data Mashup method
9. AR Data Format – POI(images, text, 3D models, URLs), Person..
10. AR Data Service API
11. Open Marker Database
12. Security & Privacy Key

2.1.3 AR Frame Work

Figure 2.3 [7] represents AR framework. In the AR framework, we can see the process involved in execution of an android application through any of your android devices.

1. When you start your application, first the camera will be on and then
2. It starts to focus on the desired object or an image
3. Then it identifies the object that it acquired from the camera.
4. Then this image is compared with the stored database; the data base is either stored within a local device or in a cloud database (In our application this is
achieved by Vuforia device database, where we store images on to Vuforia database and then download it to our local device).

5. During the comparison stage, if the image matches with one of the images stored in the database then the related information will be displayed on your android device.

### 2.1.4 Vuforia Applications

The applications of mobile AR can be found in the following areas:

- Navigation
- Gaming
- Location overlay
- Geo-Information services

### 2.1.5 Other Available AR Development SDKs in the Market

Other than Qualcomm’s Vuforia, there are few AR SDKs that are available in the market. They are as follows:

- Metaio AR SDK by Metaio, Munich, Germany.
- Wikitude AR SDK by Wikitude

### 2.1.6 Metaio vs. Vuforia

The Table 2.1 [7] represents the comparison of the features provide by the Metaio and Vuforia SDKs. The SDKs for developing AR applications can be chosen according to the features required in the app.

### 2.2 ANDROID PLATFORM

Android refers to either android mobile devices or android operating system. Android Operating system is the powerhouse of all the android devices available in today’s market. This OS gives instructions to the devices on how to behave, very similar to other operating system. The primary language used here is JAVA.

Android is standing as a king in the universe of mobile phones and tablets. In this creative android world with the combination of well equipped cell phones, increasing social networking, efficient web technology services and demands for location based products are all responsible to provide an efficient and versatile platform for Augmented Reality (AR).
Table 2.1. Comparison between Metaio and Vuforia AR SDKs

<table>
<thead>
<tr>
<th>Platform</th>
<th>Metaio</th>
<th>Vuforia</th>
</tr>
</thead>
<tbody>
<tr>
<td>iOS</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Android</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Web</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Feature</th>
<th>Metaio</th>
<th>Vuforia</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>IMU Sensors</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Cloud Support</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Content API</td>
<td>OpenGL, in-house 3d renderer</td>
<td>OpenGL only</td>
</tr>
<tr>
<td>Unity (3D)</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Markers</th>
<th>Metaio</th>
<th>Vuforia</th>
</tr>
</thead>
<tbody>
<tr>
<td>ID Marker</td>
<td></td>
<td>Image</td>
</tr>
<tr>
<td>Picture Marker</td>
<td></td>
<td>Cylinder</td>
</tr>
<tr>
<td>Markerless</td>
<td></td>
<td>Text Recognition</td>
</tr>
<tr>
<td>Markerless 3D</td>
<td></td>
<td>Frame Marker</td>
</tr>
<tr>
<td>LLA Markers</td>
<td></td>
<td>Custom</td>
</tr>
<tr>
<td>QR &amp; Barcode</td>
<td></td>
<td>Virtual Button</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Non-Optical Tracking</th>
<th>Metaio</th>
<th>Vuforia</th>
</tr>
</thead>
<tbody>
<tr>
<td>GPS/ Inertial Sensors</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Dummy</td>
<td>No</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>License</th>
<th>Metaio</th>
<th>Vuforia</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Free + Commercial</td>
<td>Free + Cloud Service Charges</td>
</tr>
</tbody>
</table>


Augmented Reality is the one of the most creative and intelligent evolution from android developers.

The apps we usually download on our android mobile phones or tablets are the android applications that are specifically written for the android operating system. This app can be installed on windows desktop computer or laptops, only if there is an Android Emulator running in your windows system. This emulator helps to create a virtual feel of android mobile device on your desktop pc. Android is an independent OS designed only for mobile device which is robust in nature.
Android Operating system versions are named for candies. Android 1.6 – Donut, Android 2.0 – Éclair, Android 2.2 – Froyo, Android 2.3 – Gingerbread, Android 3.0 – Honeycomb, Android 4.0 – Ice Cream Sandwich, Android 4.1 – Jelly Bean, Android 4.4 – KitKat and Android 5.0 – Lollipop.

2.2.1 Android Framework

Android Software Development Tool Kit (SDK) includes the application framework and the Dalvik Virtual Machine. Dalvik is a memory optimized runtime environment which is responsible for executing class files. In Java JVM is used to execute class files.

Android SDK includes webkit, 2-D and 3-D graphics library, SQLite and supports a variety of media formats such as MPEG-4, MP3, AMR, JPG, PNG and GIF). All of these functionalities are achievable through Eclipse IDE.

Figure 2.4 [8] represent the overall structure of an android framework.


The android framework comprises of four key layers [8]:

- Applications
- Application Framework
- Libraries
- Dalvik VM
- Core (Google) Java Libraries
- Linux 2.6.x (Kernel)
• **Applications Layer**: This is the end layer which includes the fully furnished android applications.

• **Application Framework Layer**: This framework includes the system libraries and associated application code.

• **Library and runtime Layer**: This layer includes the core system libraries, Dalvik virtual Machine, and Google APIs for Java.

• **Kernel layer**: Responsible for all the core operations of the android OS, such as device driver, schedulers, memory management etc.

### 2.2.2 Advantages of Android Platform

• Android is completely focused on application development and not on peripheral requirements such as kernel compilation and installation [8].

• Android OS is very customizable (like changing UI, file system, Data transfer etc is easier) in comparison to IOS, where everything is blackboxed and does not allow the users to modify these configuration settings.

• One of the key features of android is that it provides multitasking; android can run more than one application simultaneously.

### 2.3 Java

“Java is a general-purpose computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible” [9].

This programming language serves as the base for most of the client and server side network related web applications. The main reason for this kind of popularity for the language is because of its nature- “Write once, run anywhere” i.e., in Java, the code compiled in one machine platform can run on any other computer platform without having to recompile it. The reason behind this nature is JVM (Java Virtual machine). Figure 2.5 [10] shows the execution process of a java source code.

In Java, the source code is written in .java file. This source code includes classes, methods, variables, objects etc. Javac is used to compile this code and generate .class file (also known as byte code). This .class file containing byte code is an input to the JVM. JVM reads this bytecode and interpret it and executes the program.
Basically JVM is a virtual computer that runs as a software process and resides in the real computer. This virtual machine is responsible for Java to act as a platform independent programming language. Figure 2.6 [10] represents the different components of JVM.

Stack in JVM is often called the Stack Frame. This stack in the memory segment stores the local variables and method related arguments of all the methods within a java source code implementation. The registers within a stack, namely- vars, frame, and optop are
used during stack manipulation. This runtime stack is used, for example, for storage of local variables of a function invocation.

The bytecode resides in the Method Area of the JVM. Method area keeps track of the program counter (PC) during the one by one execution of the instructions.

Other objects in Java programs are stored in the Heap memory i.e., whenever we create objects using new operator, then the memory for this object is allocated in the heap memory. In C++ we do have free operator to free these memory allocation after each program execution. But in java this process of freeing up the object’s memory allocation is done automatically using the Garbage Collection mechanism. One of the past algorithms used for this garbage collection mechanism was mark and sweep.

2.3.1 Object –Oriented Features of Java

Object Oriented Programming (OOPs) is a methodology or paradigm to design a program using classes and objects. It simplifies the software development and maintenance by providing the following few concepts [11]:

- **Object**: Any entity that has state and behavior is known as an object. Entity can be either physical or logical.

- **Class**: Collection of objects is called class. It is a logical entity. Objects “instantiates” their defining class.

- **Inheritance**: An object that acquires all the properties and behaviors of another parent object is known as inheritance. The advantages of inheritance includes code reusability and runtime polymorphism

- **Abstraction**: Hiding internal details and showing functionality is known as abstraction. Abstraction in java is enhanced through abstract class and interface.

- **Polymorphism**: Polymorphism by name means that a task can have many forms. Eg: if we take GeometricalObjects as a class and shape as a method inside that class, then the shape method can have different forms, such as square, rectangle, triangle etc.

- **Encapsulation**: Wrapping up of data and code in to a single unit is known as encapsulation. All java classes are example of encapsulation.
2.3.2 JAVA and Augmented Reality (AR)

Android prefers JAVA as a programming language for application development. The reason for using JAVA is because

- It is a commonly used programming language which provides enormous Google APIs that support building more efficient and sophisticated Android application components.
- It has a platform independent nature. Since java can run on any virtual machine, this reduces the recompilation task on different mobile devices.
- Java provides better security features along with many development tool options.
- Java is known to be an industrial language which is compatible with most mobile phones in the market.
- It has efficient object oriented features.

Android uses the Dalvik Virtual machine (DVM) to execute class files instead of JVM, and was provided by GOOGLE through a Java API.

2.4 ECLIPSE IDE

“Eclipse is a popular open source IDE (Integrated Development Environment) initially developed by IBM. It is primarily used for Java development, but it supports a wide variety of other programming languages” [12].

Eclipse platform is designed using Java as a primary programming language. This requires JVM to run the java source code. This software product is built by plugins which are considered to be small units of functionality. These plugins are JAR files along with manifest files which describes the plugins. “When Eclipse is started, the runtime platform scans the manifests of the plugins in your install, and builds a plugin registry that is stored in memory” [13].

2.4.1 Java Development Tools (JDT)

JDT provides editors, debugger, compiler and builder for Java. Java SDK (Software Development Tool Kit) was not provided with Eclipse software, it was available separately and users can choose the required SDK for their system.

JDT provides the tool plugins that implement a Java IDE supporting the development of any java application [13].
2.4.2 Plug-In Development Environment (PDE)

PDE provides tools for developing, building, deploying and testing plugins. PDE build examines the dependencies of the plugins and generates Ant scripts to construct the build artifacts [13].

2.4.3 Advantages of Using Eclipse

- More sophisticated and user friendly interface.
- Efficient debugging tools used for step-by-step code inspection by applying break points.
- Refactoring. This concept in Eclipse will come in handy while doing renames of particular variables, methods, or class.
- Assistance in creating JAR files.

2.5 XML & HTML

XML stands for Extensible Markup Language, much like HTML. But the difference between XML and HTML language is that XML was designed to describe data, with focus on what data is; on the other hand HTML was designed to display data, with focus on how the data looks [14].

In some web application both XML and HTML are used together, where XML describes data, and HTML formats and display the data.

In Figure 2.7 [5] we can see the benefits of using web technology for AR. Basically network-based AR refers to mobile technology that can overlay information from the web on top of objects in the real world.

Since, android AR applications are leveraging in the mobile industry, a native application was developed. The reason for this development was because of increasing mobile integration capabilities and the ability to access such functionalities (e.g.: camera). The key drawback of this native development is that it was not possible to transfer these applications across multiple devices and platforms.

So, to overcome this drawback, native applications started using web –based components. This solved the cross platform issue in native development; this can be observed from the Figure 2.7 [5].
CHAPTER 3

SOFTWARE PRODUCT DEVELOPMENT
LIFECYCLE

The software development includes the following phases in order to shape a final product to work effectively and efficiently as desired. This lifecycle includes the following phases as shown in Figure 3.1 [15]:

- **Requirement Analysis:** This phase includes the gathering of all the software and hardware requirements needed to develop an application. This can be memory requirements, software packages need to develop, debug and execute the application product.

- **Design:** This phase is to develop the architectural design of a software application based on the requirement analysis phase. So, this is actually the web view or block diagram representation of how the final product will look.

- **Implementation:** This phase is where actually we start implementing the source code for our application by using one of the available programming languages (such as Java, C, and C++ etc.)

- **Testing:** This phase is to analyze whether the programming modules are working as desired.

- **Evolution:** This is the product release phase where the product will be introduced to the market after all the necessary tests are performed.
CHAPTER 4

SYSTEM REQUIREMENTS

Below is a bullet point style summary of hardware and software requirements for this project.

H/W System Requirements: (Minimum H/W System Requirements):

- Processor : Pentium –Core(i3)
- Speed : 2.0 GHz
- RAM : 2GB
- Hard Disk : 20 GB
- Key Board : Standard Windows Keyboard
- Mouse : Two or Three Button Mouse
- Monitor : SVGA

Supporting Devices (Phone Specifications):

- OS : Android 2.0 and above
- Inbuilt Camera : Yes
- Camera Specifications : 3Megapixel Camera

S/W System Requirements: (Minimum):

- Debugging Tools : Eclipse IDE
- Front End : Android XML
- Database : Vuforia
- Tool sets : NDK, QCAR, ADT, and AVD
CHAPTER 5

SETTING UP THE DEVELOPMENT ENVIRONMENT

In order to develop an android AR application there is a need for the installation of many software packages. The following are the software packages to be installed on your computer before starting development of this application:

1. First step in any AR application development is that all the images used in the product must be uploaded to the database. This database can be stored on to our local device (Device Database) or on to the cloud (Cloud Database). In our application we are storing the university campus images on to Vuforia device database.

2. Setting up the Android Development Environment.

3. Last step is to install Vuforia Android SDK.

5.1 CREATING DEVICE DATABASE-SET ON VUFDRIA

Initially the pictures of university campus buildings are captured in all the angles and orientation and then these images are to be uploaded to the Vuforia database. Following are the steps involved in creating Device Database through Vuforia Target Manager [15]:

- Go to the Target manager Home page https://developer.vuforia.com/target-manager and create an account to register to Vuforia.

- Then under Target Manager tab click on Create Database button. Then select Device Database option and enter the unique database name and click on create button.

- Now a window will pop up to upload the image targets (university campus pictures). Here enter the target name and upload the desired image by selecting the browse button to browse within your local system. This file must me 8 or 24-bit PNG or JPG. A JPG file must be RGB or grayscale with maximum image size 2.25MB.

- After uploading an image click Add. Then after processing, the details of the target uploaded will be displayed i.e., the rating, picture quality and if required, improvement recommendations display along with the target image.

- After creating a database. Click on the Download Database button as shown in Figure 5.1. This will download a <database_name>.zip folder on your working computer system. Then extract this zip folder. This folder contains the
<database_name>.dat and <database_name>.xml file. This folder in later stages will be stored under the asset folder while implementing the application on Eclipse IDE.

![Vuforia Target manager](image)

**Figure 5.1. Vuforia Target manager.**

### 5.2 Setting up the Android Development Environment

Vuforia SDK requires supporting software packages such as Android SDK and Android NDK. NDK is an extension of the Android SDK that lets Android developers build performance-critical parts of their applications in native code. The SDK and NDK communicate over the Java Native Interface [16].

In order to setup a development environment for Android, then following components need to be installed with recent versions [16].

1. JDK
2. Android ADT (Android Developer Tools) Bundle. This includes:
   - Eclipse IDE with the ADT plugin
   - Android SDK tools
   - Android Platform tools
• Latest Android SDK platform
4. Android NDK

5.2.1 JDK (Java Development Tool Kit) Installation
Follow the instructions below to install the latest JDK version:
• Go to the following link http://www.oracle.com/technetwork/java/javase/downloads/ and then download the JDK package depending upon whether you system is 32-bit or 64-bit windows operating system.
• Install the JDK environment with default environment settings.

5.2.2 Android ADT (Android Development Tools) Bundle Installation
Follow the instructions below to download and install ADT bundle:
• Download the ADT bundle from the android developer official website:
  • http://developer.android.com/sdk/index.html/
• This download will be the .zip folder. So, extract this zip folder after the download is complete. This extracted ADT bundle folder contains the following sub folders
  • Eclipse
  • sdk
  • S/Dk Manager.exe
• Now store the above extracted folder to the following path within your computer: C:\Development\Android

5.2.3 Setting Up Android SDK Manager
• After storing the folder in the above specified file path, then go to the Eclipse folder and run eclipse.exe
• Once the Eclipse is opened, goto Window -> Android SDK Manager. As soon as you click on Android SDK Manager, a popup window will open as shown in Figure 5.2.
• Then select the required component in the given check boxes, click Install packages at the corner right bottom of the pop-up window and then accept the license to start the installation. But the essential packages are: Android API level which should be above Android 2.2 for our application to work properly and the other is the Google USB Driver package.
After this, add the directory path of Android SDK platform tools to the windows environment variable named PATH; to do this right-click on my computer -> select Properties -> Select Advanced system settings -> under Advanced tab, select Environment Variables -> Under System variables section look for PATH variable and then add C:\Development\Android\android-sdk-windows\platform-tools\ and then click ok to save the changes that have been made.

5.2.4 Setting Up Cygwin Environment

In order to build an android application with NDK build, then there is a requirement for GNU compiler to compile dynamic applications in the form of shared libraries. On windows, a convenient way to create such a compiler environment is through Cygwin.

Follow the below steps to install Cygwin from its official website:
• The run the setup.exe file.
• Then it asks to select whether to “install from the internet” or through the file stored in the local device.

Select install from the internet and then it will ask to choose the path to save the installer. The default path is “C:\cygwin”.

### 5.2.5 Setting Up Android NDK (Native Development Tool Kit)

In our AR application we are making use of both java and C++ APIs. So, in order to enable native C++ programming on Android, Android NDK needs to be installed. This installation is not required if we use only Java APIs. Below are the steps involved in the NDK installation process.

• First download the NDK package from http://developer.android.com/sdk/ndk/index.html
• This downloads the .zip folder. Unzip this folder. Now for the ease of convenience, locate this folder in to “C:\Development\Android\”.
• This directory path needs to be added to the system variables. For this, copy the NDK folder location and then follow the steps mentioned in the previous section to store the PATH for Android SDK.
• Now in order to build your application, open Cygwin and change the root path to the path where your application is stored. Then execute the command statement below to build your application:
  “ndk-build”
• The result of this build will create dynamic link libraries .so files and store them under /libs/armeabi folder of your application.

### 5.3 INSTALLING VUFORIA SDK

• To install Vuforia SDK, go to the link https://developer.vuforia.com/downloads/sdk and then click on the “Download SDK for Android”.
• This downloads the .zip folder. Unzip this folder and store it under “C:\Development\Android\”.
• Now in eclipse, right click on your project root and select properties -> Java Build Path -> under Libraries -> click on “Add External Jars” -> now browse for the location where your Vuforia SDK’s QCAR.jar file is stored, this should be found in C:\Development\Android\Android-SDK/build/java. Now once this
library is added, clean your project after all the settings are saved by selecting *Project* tab -> *clean* in eclipse.
CHAPTER 6

CREATING AN ANDROID APPLICATION USING ECLIPSE IDE

This chapter explains the step by step process to create and execute an Android application in Eclipse. Below are the steps involved right from the creation to execution of an android application.

1. First in order to start eclipse, go-to C:\Development\Android\ADT-Bundle\Eclipse, and then double click on eclipse.exe. It asks for the workspace to process the application. The default path is C:\Users\admin\workspace. And then click OK (see Figure 6.1).

2. Then Eclipse will open. Now go to file menu and select New -> Android Application Project (see Figure 6.2).

3. After selecting the Android Application Project a pop-up window will appear as shown in Figure 6.3. Now enter your application name and select the desired Android API level (Android SDK) as shown below and click next until you reach the final stage and then click finish at the last (follow steps from Figure 6.3 through Figure 6.7). Figure 6.3 through Figure 6.7 are the step by step snapshots.

4. Now the application package ImageTargetsApp is created as shown in Figure 6.8:

5. Now the Figure 6.9, shows how to create a .java file:
Here, under ImageTargetsApp we can find the source folder. When you right click on that folder, select the New option and then select Class. This is shown in Figure 6.9.

6. In Figure 6.10, we can observe that, under my application ImageTargetsApp, we can find many .java classes. Each class performs different functions and all are included under one package called ImageTargetsApp. The .java classes in my application include:
   - AboutScreen.java (Appendix A)
   - ContactDealer.java
   - ContactListAdapter.java
   - DebugLog.java
   - GridAdaptor.java
   - Home.java
   - ImageTargets.java (Appendix B)
Figure 6.1. Eclipse Workspace launcher.

Figure 6.2. Selecting new android application project.

- ImagetargetsRenderer.java (Appendix C)
- ImageTargetSplashScreen (Appendix D)
- QCARSampleOpenGL
- Texture

This can be observed in the table of contents in the Figure 6.10.
Figure 6.3. New Application window_1.

Figure 6.4. New Application window_2.
Figure 6.5. New application window_3.

Figure 6.6. New application window_3 → Select blank activity for an android application.
Figure 6.7. New application window - Enter the activity and layout name.

Figure 6.8. ImageTargetsApp package structure.
Figure 6.9. Represents how to create a new .java file under SRC folder.
Figure 6.10. Class Hierarchy in Android application project.
CHAPTER 7

HOW TO RUN AN ANDROID APPLICATION

There are a few settings that need to be done before executing an android application. These are as follows:

1. First we need to enable the USB Debugging inside the android devices. This is needed if you are installing your android application from your desktop PC or laptop on to your android device through a USB cable. Here are the steps to enable debugging mode.

   Navigate to Settings -> About Phone -> scroll to the bottom and tap Build number seven (7) times. You'll get a short pop-up in the lower area of your display saying that you're now a developer. 2. Go back and now access the Developer options menu, check 'USB debugging' and click OK on the prompt.

2. Now connect your android device to your Desktop PC or Laptop through USB cable.

3. Now in eclipse right click on the root of our application and select Run as and then select Android Application as shown in Figure 7.1.

4. Now after selecting Run as android application, you can see the Android Device Chooser pop up window as shown in Figure 7.2. Here this window lists all the android devices connected to your desktop PC or to your laptop. Now select one of the devices and click OK. Now the installation of an ImageTargetsApp application in to one of the connected android devices will start.

5. After the completion of the installation of an ImageTargetsApp, now you can use your android application. Figure 7.3 shows that our android application UniversityInfo” has been installed successfully on our android tablet.

6. Now click on UniversityInfo app. Then we get a splash screen that displays one of the university campus buildings images randomly from the database. Figure 7.4 shows the flash screen that appears at the first of the application launch.

7. After 2 secs, the splash screen control is redirected to the android device camera. Now hold the android device camera on to a particular university campus building image or on to a live building. Now the camera starts to focus the image and if it matches with an image in the database, then related information is displayed. Figure 7.5 shows how to focus the android device camera on to a university building.

8. The camera will still focus on the image until if one of the images is matched with the database. Once there is a match, then the following screen appears shown in Figure 7.6.

9. In Figure 7.7 we show the options available after the image has been recognized:
10. Now if you click any of the above displayed options, the related information will be displayed. In Figure 7.8, Figure 7.9, and Figure 7.10 we see the images for FacultyInfo options:

11. Now if you click on any professor’s image, then their respective contact information will be displayed. This is shown in Figure 7.11.
Figure 7.2. Running ImageTargetsApp android application_1 – Android device chooser.

Figure 7.3. UniversityInfo app has been installed.
Figure 7.4. UniversityInfo application splash screen.
Figure 7.5. Focusing android device camera over the university image on a laptop screen.
Figure 7.6. Option page appears after the image is recognized by the camera.

Figure 7.7. Options Screen_1.
Figure 7.8. Main OptionPage.

Figure 7.9. ComputerScienceDept OptionPage.
Figure 7.10. Computer Science FacultyInfo InfoPage.

Figure 7.11. FacultyInfo page.
CHAPTER 8

FUTURE ENHANCEMENTS

This University Campus AR application serves the basic purpose of the need to design an android application in order to make university related information available to the students in an easier and faster way.

The most obvious enhancement is to take many more campus pictures and load them in to the database. Ideally the application would be tested by having the “naïve” users take the android pictures of buildings to see if the database has adequate pictures for building identification in reasonable time.

This was a “prototype”, in that only a handful of buildings were cataloged into the database.

Add a website containing the listing application, and instructions for communicating with the thesis rather about additions to the picture database.

Since, the AR is leveraging in the android world, there is a very good scope of enhancements to this application in the future. Below are the few enhancements that would make this application look and feel more efficient.

• One of the features that needs to be added and will be of great help is the “location based services”. So, this helps students to go to the particular office or to the building or to a particular classroom by enabling the location identification service within the app.

• Links to access the university websites.

• Improvements in the User-Interface by making it more user-friendly.
CHAPTER 9

SUMMARY AND DEVELOPMENT CHALLENGES

“SDSU University Campus Tour” is a free application, which is easy to use, universally available, simple, and a thorough Application that helps students to refer to the campus related information without the use of the Internet and saves time in peeking through the university links for gathering university related information.

The students will get to know their department, faculty, courses and favorite restaurants information merely by focusing an android device camera on to the live campus building or on to respective images. The success of a system like the one proposed here would depend on the acceptance by students who gets complete university information in one place.

The key challenge faced during the design of this algorithm was gathering the images of the buildings in each and every angle and orientation and the next challenge was the image recognition algorithm, which is the heart of this application. So, this application required very stable, efficient and effective image recognition algorithms.

Taking the pictures from many angles aroused interest in campus security, but this turned out to be a minor challenge.

Note that the software packages used in developing this Android application is intended to be public domain and can be obtained from the supervisor’s website for free of cost.
REFERENCES


APPENDIX A

ABOUTSCREEN.JAVA
package com.qualcomm.QCARSamples.ImageTargets;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.text.Html;
import android.text.method.LinkMovementMethod;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

public class AboutScreen extends Activity implements OnClickListener {
    private TextView mAboutText;
    private Button mStartButton;

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.about_screen);

        mAboutText = (TextView) findViewById(R.id.about_text);
        mAboutText.setText(Html.fromHtml(getString(R.string.about_text)));
        mAboutText.setMovementMethod(LinkMovementMethod.getInstance());
        // Setup the link color
        mAboutText.setLinkTextColor(getResources().getColor(R.color.holo_light_blue));

        mStartButton = (Button) findViewById(R.id.button_start);
        mStartButton.setOnClickListener(this);
    }

    /** Starts the ImageTargets main activity */
    private void startARActivity() {
        Intent i = new Intent(this, ImageTargets.class);
        startActivity(i);
    }

    public void onClick(View v) {
        switch (v.getId()) {
            case R.id.button_start:
                startARActivity();
                break;
        }
    }
}
APPENDIX B

IMAGETARGETS.JAVA
import android.app.Activity;
import android.app.AlertDialog;
import android.content.DialogInterface;
import android.content.pm.ActivityInfo;
import android.content.res.Configuration;
import android.graphics.Color;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.DisplayMetrics;
import android.view.GestureDetector;
import android.view.KeyEvent;
import android.view.LayoutInflater;
import android.view.MenuItem;
import android.view.MotionEvent;
import android.view.View;
import android.view.ViewGroup.LayoutParams;
import android.view.WindowManager;
import android.widget.RelativeLayout;
import android.widget.Toast;
import com.qualcomm.QCAR.QCAR;

/** The main activity for the ImageTargets sample. */
public class ImageTargets extends Activity {

    // Focus mode constants:
    private static final int FOCUS_MODE_NORMAL = 0;
    private static final int FOCUS_MODE_CONTINUOUS_AUTO = 1;

    // Application status constants:
    private static final int APPSTATUS_UNINITED = -1;
    private static final int APPSTATUS_INIT_APP = 0;
    private static final int APPSTATUS_INIT_QCAR = 1;
    private static final int APPSTATUS_INIT_TRACKER = 2;
    private static final int APPSTATUS_INIT_APP_AR = 3;
    private static final int APPSTATUS_LOAD_TRACKER = 4;
    private static final int APPSTATUS_INITED = 5;
    private static final int APPSTATUS_CAMERA_STOPPED = 6;
    private static final int APPSTATUS_CAMERA_RUNNING = 7;

    // Name of the native dynamic libraries to load:
    private static final String NATIVE_LIB_SAMPLE = "ImageTargets";
    private static final String NATIVE_LIB_QCAR = "QCAR";
// Constants for Hiding/Showing Loading dialog
static final int HIDE_LOADING_DIALOG = 0;
static final int SHOW_LOADING_DIALOG = 1;

private View mLoadingDialogContainer;

// Our OpenGL view:
private QCARSampleGLView mGlView;

// Our renderer:
private ImageTargetsRenderer mRenderer;

// Display size of the device:
private int mScreenWidth = 0;
private int mScreenHeight = 0;

// Constant representing invalid screen orientation to trigger a query:
private static final int INVALID_SCREEN_ROTATION = -1;

// Last detected screen rotation:
private int mLastScreenRotation = INVALID_SCREEN_ROTATION;

// The current application status:
private int mAppStatus = APPSTATUS_UNINITED;

// The async tasks to initialize the QCAR SDK:
private InitQCARTask mInitQCARTask;
private LoadTrackerTask mLoadTrackerTask;

// An object used for synchronizing QCAR initialization, dataset loading and
// the Android onDestroy() life cycle event. If the application is destroyed
// while a data set is still being loaded, then we wait for the loading
// operation to finish before shutting down QCAR:
private Object mShutdownLock = new Object();

// QCAR initialization flags:
private int mQCARFlags = 0;

// The textures we will use for rendering:
private Vector<Texture> mTextures;

// Detects the double tap gesture for launching the Camera menu
private GestureDetector mGestureDetector;

// Contextual Menu Options for Camera Flash - Autofocus
private boolean mFlash = false;
private boolean mContAutofocus = false;

// The menu item for swapping data sets:
MenuItem mDataSetMenuItem = null;
boolean mIsStonesAndChipsDataSetActive = false;

private RelativeLayout mUILayout;

/** Static initializer block to load native libraries on start-up. */
static {
    loadLibrary(NATIVE_LIB_QCAR);
    loadLibrary(NATIVE_LIB_SAMPLE);
}

/**
* Creates a handler to update the status of the Loading Dialog from an UI Thread
*/
static class LoadingDialogHandler extends Handler {
    private final WeakReference<ImageTargets> mImageTargets;

    LoadingDialogHandler(ImageTargets imageTargets) {
        mImageTargets = new WeakReference<ImageTargets>(imageTargets);
    }

    public void handleMessage(Message msg) {
        ImageTargets imageTargets = mImageTargets.get();
        if (imageTargets == null) {
            return;
        }

        if (msg.what == SHOW_LOADING_DIALOG) {
            imageTargets.mLoadingDialogContainer.setVisibility(View.VISIBLE);
        } else if (msg.what == HIDE_LOADING_DIALOG) {
            imageTargets.mLoadingDialogContainer.setVisibility(View.GONE);
        }
    }
}

private Handler loadingDialogHandler = new LoadingDialogHandler(this);

/** An async task to initialize QCAR asynchronously. */
private class InitQCARTask extends AsyncTask<Void, Integer, Boolean> {
    // Initialize with invalid value:
    private int mProgressValue = -1;

    protected Boolean doInBackground(Void... params) {
        }
    }

// Prevent the onDestroy() method to overlap with initialization:
synchronized (mShutdownLock)
{
    QCAR.setInitParameters(ImageTargets.this, mQCARFlags);

    do
    {
        // QCAR.init() blocks until an initialization step is
        // complete, then it proceeds to the next step and reports
        // progress in percents (0 ... 100%).
        // If QCAR.init() returns -1, it indicates an error.
        // Initialization is done when progress has reached 100%.
        mProgressValue = QCAR.init();

        // Publish the progress value:
        publishProgress(mProgressValue);

        // We check whether the task has been canceled in the
        // meantime (by calling AsyncTask.cancel(true)).
        // and bail out if it has, thus stopping this thread.
        // This is necessary as the AsyncTask will run to completion
        // regardless of the status of the component that
        // started is.
        while (!isCancelled() && mProgressValue >= 0
                && mProgressValue < 100);

        return (mProgressValue > 0);
    }

}
APPENDIX C

IMAGETARGETSRENDERER.JAVA
ImageTargetsRenderer.java

package com.qualcomm.QCARSamples.ImageTargets;

import javax.microedition.khronos.egl.EGLConfig;
import javax.microedition.khronos.opengles.GL10;

import android.content.Context;
import android.content.Intent;
import android.opengl.GLSurfaceView;
import android.os.Handler;
import android.util.Log;
import android.widget.Toast;
import com.qualcomm.QCAR.QCAR;

/** The renderer class for the ImageTargets sample. */
public class ImageTargetsRenderer implements GLSurfaceView.Renderer {
    public boolean mIsActive = false;

    /** Reference to main activity **/
    public ImageTargets mActivity;

    /** Native function for initializing the renderer. */
    public native void initRendering();

    /** Native function to update the renderer. */
    public native void updateRendering(int width, int height);

    public static Handler MainActivityHandler;

    public static Context context;
    public static int flag = 0;

    public void onImageTracked(String trackableName) {
        Log.d("tour","in onImageTracked....."+trackableName);
        // Toast.makeText(context, trackableName, Toast.LENGTH_LONG).show();

        Intent intent = new Intent(context,Home.class);
        //intent.putExtra("name",trackableName);

        if(trackableName.startsWith("gmcs")) {
            Log.d("tour","tracked....."+trackableName);
            intent.putExtra(Home.SELECTED_CAR, Home.GMCS);
        } else if(trackableName.startsWith("ps")) {
        }
    }
}
intent.putExtra(Home.SELECTED_CAR, Home.PHYSICS);
}
else if(trackableName.startsWith("ec"))
{
    intent.putExtra(Home.SELECTED_CAR, Home.EASTCOMMONS);
}
context.startActivity(intent);

/** Called when the surface is created or recreated. */
public void onSurfaceCreated(GL10 gl, EGLConfig config)
{
    DebugLog.LOGD("GLRenderer::onSurfaceCreated");
    // Call native function to initialize rendering:
    initRendering();
    // Call QCAR function to (re)initialize rendering after first use
    // or after OpenGL ES context was lost (e.g. after onPause/onResume):
    QCAR.onSurfaceCreated();
}

/** Called when the surface changed size. */
public void onSurfaceChanged(GL10 gl, int width, int height)
{
    DebugLog.LOGD("GLRenderer::onSurfaceChanged");
    // Call native function to update rendering when render surface
    // parameters have changed:
    updateRendering(width, height);
    // Call QCAR function to handle render surface size changes:
    QCAR.onSurfaceChanged(width, height);
}

/** The native render function. */
public native void renderFrame();

/** Called to draw the current frame. */
public void onDrawFrame(GL10 gl)
{
    if (!mIsActive)
        return;
    // Update render view (projection matrix and viewport) if needed:
    mActivity.updateRenderView();
    // Call our native function to render content
    renderFrame();
}
APPENDIX D

IMAGETARGETSSPLASHSCREEN.JAVA
package com.qualcomm.QCARSamples.ImageTargets;

import android.app.Activity;
import android.content.Intent;
import android.content.res.Configuration;
import android.os.Bundle;
import android.os.Handler;

public class ImageTargetsSplashScreen extends Activity {

    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // Sets the Splash Screen Layout
        setContentView(R.layout.splash_screen);

        // Generates a Handler to launch the About Screen
        // after 2 seconds
        final Handler handler = new Handler();
        handler.postDelayed(new Runnable() {
            public void run() {
                // Starts the About Screen Activity
                startActivity(new Intent(ImageTargetsSplashScreen.this,
                                            ImageTargets.class));
            }
        }, 2000L);
    }

    public void onConfigurationChanged(Configuration newConfig) {
        // Manages auto rotation for the Splash Screen Layout
        super.onConfigurationChanged(newConfig);
        setContentView(R.layout.splash_screen);
    }

}