Augmented Reality and ARToolkit for Android: the First Steps

Liliya Demidova[1, 2,*]

[1]Ryazan State Radio Engineering University, 390005, Ryazan, Russia
[2]Moscow Technological Institute, 199334, Moscow, Russia

Abstract. The paper describes the aspects of the augmented reality application development on the base of ARToolkit for Android. The steps consistently used for the augmented reality application development have been considered. The simplest Java-based examples of augmented reality have been discussed.

1 Introduction

Augmented reality (AR) 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" [1]. AR is related to a more general concept called mediated reality, in which a view of reality is modified by a computer. AR technology works by strengthening one's current perception of reality, while virtual reality replaces the real world with a simulated one. On the base of AR technology the information about the surrounding real world of the user becomes interactive. Artificial information about the environment and its objects can be imposed on the real world. The virtual objects of AR display information that the user cannot directly detect with his own senses. The information conveyed by these virtual objects helps a user perform real-world tasks in the best way.

In 1997, R. Azuma published a survey [2] that defined the field, described many problems, and summarized the developments up to AR. Since then, the field of AR has grown rapidly. Nowdays we can speak about almost 50 years of research and development in the field of AR [3].

R. Azuma defines AR as systems that have the following three characteristics [2]:
• combines real and virtual;
• interactive in real time;
• registered in 3D.

Registration refers to the accurate alignment of real and virtual objects. Without accurate registration, the illusion that the virtual objects exist in the real environment is severely compromised. Registration is a difficult problem and a topic of continuing research [4].

To enable rapid development of AR application, the software development kits (SDK) have emerged. Some of the well known AR SDKs are offered by ARToolKit, Blippar, Catchoom CraftAR, Layar, Vuforia, etc.

2 ARToolkit for Android

ARToolkit is the most popular freely available AR software development kit. ARToolkit implements the logic necessary for AR, including [5]:
• recognizing markers in the video frames;
• determining the geometry of the scene (i.e., the position and angle of the marker);
• projecting the 3D model into the scene;
• creating the composite video frame (the view plus the 3D model in registration with the marker).

ARToolkit works with ordinary PCs and laptops equipped with inexpensive web cams.

AR applications on the base of ARToolkit for Android can be used for the smart phones. These devices are of interest because they are rapidly becoming ubiquitous, and therefore have the potential to transform science, engineering, and technical education at relatively low cost [5, 6].

AR applications are written in special 3D programs that allow the developer to connect animation or contextual digital information in the computer program to an AR "marker" [7] in the real world. When a computing device's AR application or browser plug-in receives digital information from a known marker, it begins to execute the marker's code and layer the correct image or images [8].

AR applications for smart phones typically include global positioning system (GPS) to pinpoint the user's location and its compass to detect device orientation. Sophisticated AR programs used by the military for training may include machine vision, object recognition and gesture recognition technologies [8].

Markers are the squares that ARToolKit recognises and tracks in a video stream [9]. That is, markers are the physical patterns that we must created or printed out. ARToolKit comes with PDF files for some pre-made markers, e.g. the Hiro marker, which we must print out and affix to card or board (so that they stay flat). Markers are the optical inputs to ARToolKit.

Markers have the following constraints [9]:

* Corresponding author: demidova.liliya@gmail.com

• they must be square;
• they must have a continuous border (generally either full black or pure white) and they must sit on a background of contrasting colour (generally the opposite of the border colour); by default, the border thickness is 25% of the length of an edge of the marker;
• the area inside the border, which we refer to as the marker image, must not be rotationally symmetric (specifically, it must not have rotational symmetry of an even order); the area inside the border can be black and white, or coloured (and ARToolKit provides a means to track with greater accuracy when the marker image is coloured).

ARToolKit includes a number of fully-working examples on Android that demonstrate basic and advanced techniques [10, 11]. We can copy and build on the examples to create our own application.

The examples are divided into 3 sets.

1. All Java-based: examples where all user-developed code is in Java, and is based on the provided ARBaseLib classes.

2. Mixed Java and native C/C++ using Android NDK: examples where user-developed code is split between the Java environment and native C/C++ environment. Code can use the provided ARBaseLib java classes while also addressing ARToolKit in C/C++ via the libARWrapper C/C++ API.

3. AR and rendering code in native C/C++ using Android NDK: examples where most of the user-developed code is native C/C++. These examples offer the greatest power to the AR developer and direct access to the full native ARToolKit API.

The simplest examples are all Java-based. These examples are as follows:
• *ARSimple*: an example that extends the ARActivity class in ARBaseLib to create a simple augmented reality application;
• *ARSimpleInteraction*: an example that adds simple interaction to ARSimple.

3 Android Examples

To work with Android Examples we must install:
• *JDK version 7u79* (http://www.oracle.com/technetwork/java/javase/downloads/index.html);
• *Android Studio with SDK and NDK* (http://developer.android.com/sdk/index.html);
• *ARToolkit5-bin-5.3.1-Android* (http://artoolkit.org/download-artoolkit-sdk).

If we want to be able to run the AR application from Android Studio's built-in emulator, it is necessary to be sure that virtualization is enabled in BIOS settings. Also, it is possible to use any external emulators, for example, *Nox APP Player* (http://en.bignox.com/#p2). Nowdays this emulator is the stablest and powerful Android emulator.

To start with *ARSimple* example we must open *Android Studio* and select "*Import project (Eclipse ADT, Gradle, etc.*" (Figure 1). Then it is necessary to choose the source files to be imported (Figure 2) and specify the directory for them (Figure 3). As a result, the project window should be appeared (Figure 4). Here we can see the project structure (Figure 5).

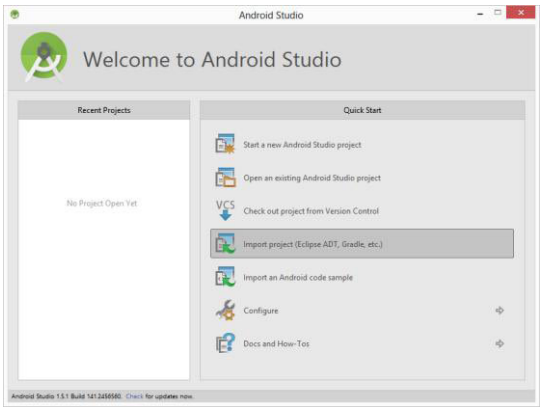


Fig. 1. Android Studio start window

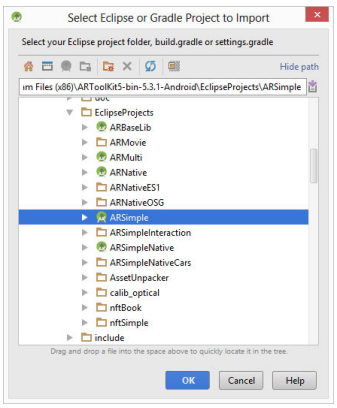


Fig. 2. ARSimple example's selection

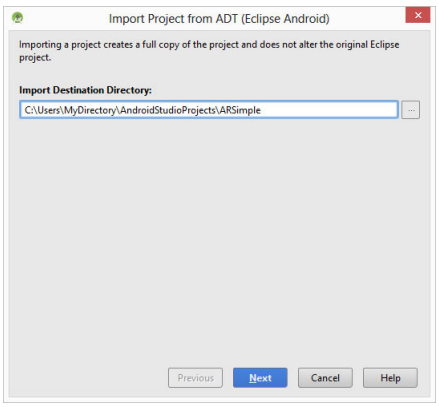


Fig. 3. The destination directory's selection

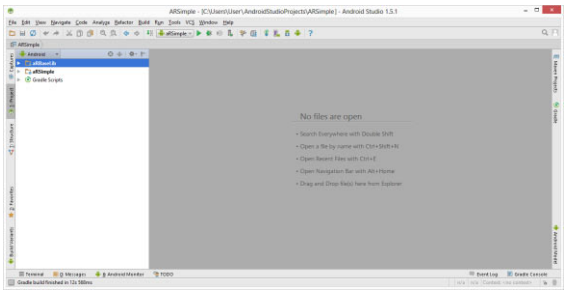


Fig. 4. The project window

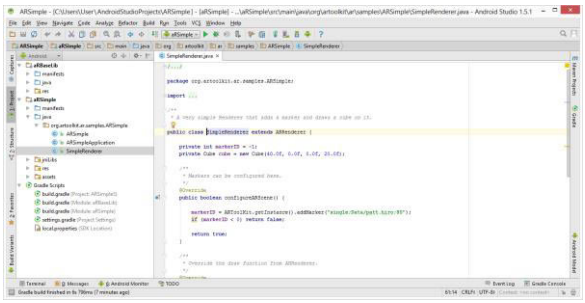


Fig. 5. The project structure

The files *ARSimple.java*, *ARSimpleApplication.java*, *SimpleRenderer.java* are of the main interest in the folder aRsimple. These files contain Java code of AR application. Also, we can see here the *aRBaseLib* folder and the *Gradle Scripts* folder.

ARBaseLib is an Android library. Android applications can reference this library, and Android Studio will take care of including the necessary files when the AR application (APK – Android Package) is built and deployed. This allows reusable components to be placed in the library and used in many different examples and applications.

Gradle is an open source build automation system that builds upon the concepts of Apache Ant and Apache Maven and introduces a Groovy-based domain-specific language (DSL) instead of the XML form used by Apache Maven of declaring the project configuration. Gradle uses a directed acyclic graph to determine the order in which tasks can be run.

It is necessary say, that Android application package (APK) is the package file format used by the Android operating system for distribution and installation of mobile applications and middleware.

To make APK it necessary to carry out the following commands:

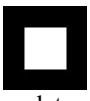
• *Build* > *Make Project*;
• *Build* > *Build APK*.

In case of success the files such as *aRSimple-debug-unaligned.apk* and *aRSimple-debug.apk* in the directory *C:\Users\MyDirectory\AndroidStudioProjects\ARSimple \aRSimple\build\outputs\apk* will be created.

The aligned APK is optimized for RAM usage so it will consumes less RAM in the devices. Therefore *aRSimple-debug.apk* should be used to test and run APK. Later (when the testing of any APK was carried out successfully), it is expedient to create the file *aRSimple-release.apk*. This file can be used for distribution.

Then we must print out the standard marker *Hiro.pdf* from the directory *C:\Program Files\ARToolkit5-bin-5.3.1-Android\doc\patterns* (Figure 6).

The Hirro marker



The template marker

Fig. 6. The markers

Later, we can create our own marker, using the template file *Blank pattern.png* (Figure 6) from the same directory. Then, we must train it [9].

To test APK we can use *Nox APP Player* emulator (Figure 7). It is necessary to install ARSimple APK and run it. As soon as the Hiro marker appears in front of the webcam (Figure 8) the colorful cube should be placed on it (Figure 9). Now we can rotate the marker. The colorful cube will rotate too (Figure 10). To stop APK we can press Esc.

Fig. 7. Nox APP Player

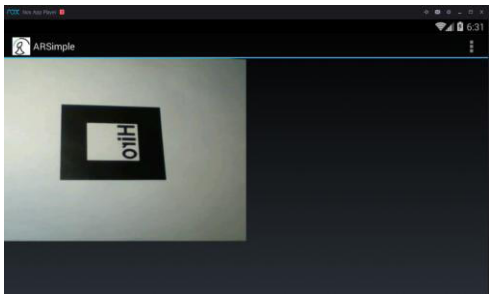


Fig. 8. The Hiro marker in front of the webcam

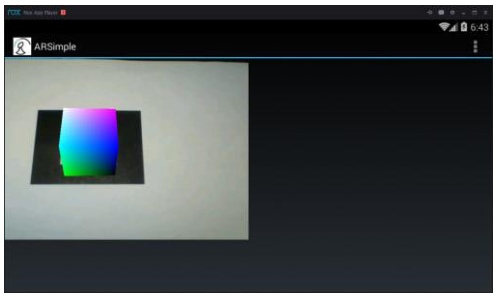


Fig. 9. The colorful cube on the marker

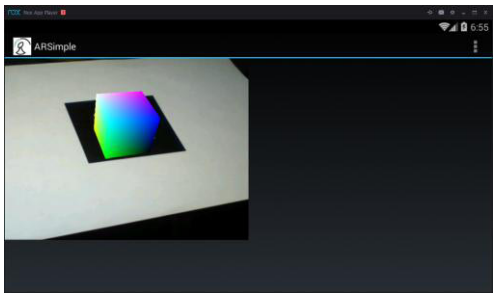


Fig. 10. The colorful cube rotates with the marker

The more interesting file within the ARSimple project is SimpleRenderer.java class. Here we can see how the marker definition is loaded

```
markerID = ARToolKit.getInstance().
addMarker("single;Data/patt.hiro;80");
```

and how the cube is placed on the image

```
gl.glLoadMatrixf(ARToolKit.getInstance().
queryMarkerTransformation(markerID, 0);
```

and drawn

```
cube.draw(gl);
```

Also, we can see the cube definition

```
private Cube cube = new Cube(40.0f, 0.0f, 0.0f,
20.0f);
```

The first instruction defines the single marker patt.hiro from the directory *C:\Users\MyDirectory\Android StudioProjects\ARSimple\aRSimple\scr\main\assets\Data* with the edge size of 80 mm.

The method queryMarkerTransformation of the second instruction allows translating and rotating the cube.

The fourth instruction defines the edge size of the cube (40 mm) and the coordinates of it center concerning the marker center (0, 0, 20). In this case the cube is raised over the marker on 20 mm.

In the simplest case we can change the marker name in the first instruction and the cube parameters in the fourth instruction. Then, it is necessary, firstly, to carry out such commands as *Make Project* and *Build APK*. and, secondly, to update the APK in the emulator and run it.

ARInteractionSimple example adds simple interaction to ARSimple. In this case the APK reproduces the spinning cube on the marker. The spinning is toggled when the user taps the screen:

```
public void click() {
   spinning = !spinning;
}
```

The following instructions

```
private Cube cube = new Cube(40.0f, 0.0f, 0.0f,
20.0f);
private float angle = 0.0f;
private boolean spinning = false;
```

define the cube parameters, the initial turn angle of the cube and the initial value of the rotation indicator correspondingly.

The marker is defined as in *ARSimple* example.

The instruction

```
gl.glRotatef(angle, 0.0f, 0.0f, 1.0f);
```

defines the turn with the predefined `angle` around the vertical axis Z counterclockwise. The last three parameters define the axis of the turn cube.

Then, the cube is drawn

```
cube.draw(gl);
```

The following instruction

```
angle += 5.0f;
```

defines increase in the turn angle by 5 degrees.

When the screen is tapped, the APK is informed and the sound of click is distributed in the emulator (the phone vibrates correspondingly):

```
Vibrator vib =
(Vibrator)getSystemService(VIBRATOR_SERVICE);
            vib.vibrate(100);
```

We can change the installations mentioned above and test the new APK. Also, we can replace the cube by other figure and so on.

4 Conclusions

The first steps of AR application development on the base of the simplest examples have been discussed. These examples can form the basis during the creation of the complex AR applications. These examples can be used as the basis during the creation of the complex AR applications, for example, for the educational purposes. In particular, they can be applied for visualization of the work principles of the swarm intelligence algorithms (step by step) under decision of many important optimization problems [12, 13].

References

1. https://en.wikipedia.org/wiki/Augmented_reality
2. R. T. Azuma, Teleoperators and Virtual Environments, **6**(4), 355-385 (1997)
3. M. Billinghurs, A. Clark, G. Lee, Foundations and Trends in Human-Computer Interaction, **8**(2-3), 73-272 (2015)
4. R. Azuma, Y. Baillo, R. Behringer, S. Feiner, S. Julier, B. MacIntyre, IEEE Computer Graphics and Applications, **21**(6), 34-47 (2001)
5. A. Craig, R.E. McGrath, A. Gutierrez, *Technical Note: Augmented Reality Software Kits for Smart Phones* (2011)
6. G. Milsap, E. Bourland, *Advanced Rendering for Augmented Reality on Mobile Devices* (2011)
7. J. Kohler, A. Pagani, D. Stricker, Detection and Identification Techniques for Markers Used in Computer Vision, Visualization of Large and Unstructured Data Sets, IRTG Workshop, pp. 36-44 (2010)
8. http://whatis.techtarget.com/definition/augmented-reality-AR
9. https://www.artoolworks.com/support/library/Creating_and_training_new_ARToolKit_markers
10. http://artoolkit.org/documentation/doku.php?id=4_Android:android_examples
11. https://www.artoolworks.com/support/library/ARToolKit_for_Android_examples#ARNative
12. L. Demidova, Yu. Sokolova, Modification of Particle Swarm Algorithm for the Problem of the SVM Classifier Development, 2015 International Conference “Stability and Control Processes“ in Memory of V.I. Zubov (SCP), 623–627 (2015)
13. L. Demidova, E. Nikulchev, Yu. Sokolova, International Journal of Advanced Computer Science and Applications, **7**(2), 16-24 (2016)