



Qualcomm® Vuforia™

The Vuforia platform enables augmented reality (AR) app experiences that are best in class and creative beyond definition. These experiences reach across most real world environments, giving mobile apps the power to see.

Vuforia is a software platform that uses top-notch, consistent, and technically resourceful computer vision-based image recognition and offers the widest set of features and capabilities, giving developers the freedom to extend their visions without technical limitations. With support for iOS, Android, and Unity 3D, the Vuforia platform allows you to write a single native app that can reach the most users across the widest range of smartphones and tablets.

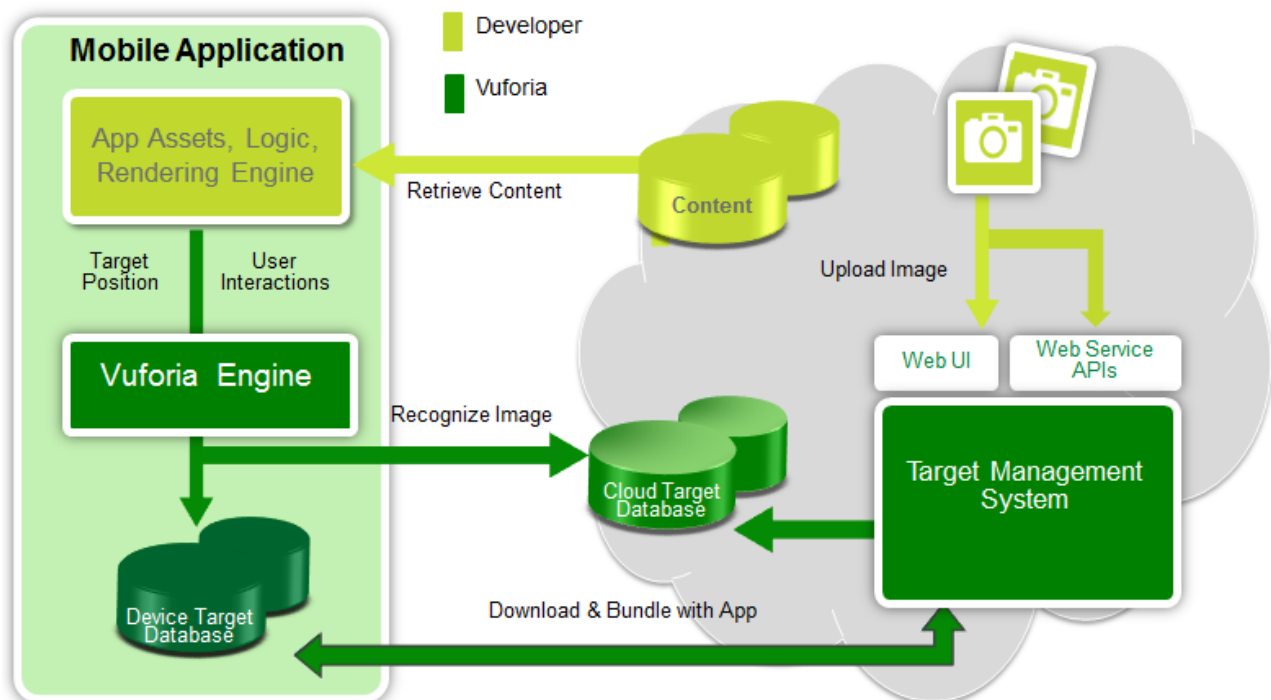
Developing with Vuforia

A Vuforia SDK-based AR application uses the display of the mobile device as a "magic lens" or looking glass into an augmented world where the real and virtual worlds appear to coexist. The application renders the live camera preview image on the display to represent a view of the physical world. Virtual 3D objects are then superimposed on the live camera preview and they appear to be tightly coupled in the real world. For more information see www.vuforia.com.

An application developed for Vuforia will give your users a more compelling experience:

- Faster local detection of targets
- Cloud recognition of up to 1 million targets simultaneously
- User-defined targets for run-time target generation
- Robust tracking – Augmentations stick to the target and are not easily lost as the device moves
- Simultaneous tracking of up to five targets
- Better results in real world conditions – Low light, partially covered target
- Optimizations that ensure better and more realistic graphics rendered on the target

This diagram provides an overview of the application development process with the Vuforia platform. The platform consists of the Vuforia Engine (inside the SDK), the Target Management System hosted on the developer portal ([Target Manager](#)), and optionally, the Cloud Target Database.



Vuforia Components

A developer uploads the input image for the target that he wants to track. The target resources can then be accessed by the mobile app in two ways:

- Accessed from a cloud target database using web services
- Downloaded in a device target database to be bundled with the mobile app

The Vuforia Engine provides a library (shared object - `libQCAR.so` on Android, static library - `libQCAR.a` on iOS) that should be linked to the app.

Vuforia supports your development efforts with the following:

- Getting started guides ([Android](#), [iOS](#), [Unity Extension](#)) to set up development on different platforms (Windows, MacOS, Linux)
- Device-specific [SDKs](#) (Android, iOS) or Extensions ([Unity Extension](#))
- Tools and services ([Target Manager](#) web UI, [Developer Guide](#), [Vuforia Web Services](#))
- [Sample apps](#) and [video tutorials](#)
- Support [forum](#) (dedicated technical support engineers, thousands of posts, FAQs)

Vuforia SDK Architecture

A Vuforia SDK-based AR application is composed of the following core components:

Camera

The camera component ensures that every preview frame is captured and passed efficiently to the tracker. The developer only has to initialize the camera to start and stop capturing. The camera frame is automatically delivered in a device-dependent image format and size.

Image Converter

The pixel format converter converts from the camera format (e.g., YUV12) to a format suitable for OpenGL ES rendering (e.g., RGB565) and for tracking (e.g., luminance) internally. This conversion also includes downsampling to have the camera image in different resolutions available in the converted frame stack.

Tracker

The tracker component contains the computer vision algorithms that detect and track real-world objects in camera video frames. Based on the camera image, different algorithms take care of detecting new targets or markers and evaluating virtual buttons. The results are stored in a state object that is used by the video background renderer and can be accessed from application code. The tracker can load multiple datasets at the same time and activate them.

Video Background Renderer

The video background renderer module renders the camera image stored in the state object. The performance of the background video rendering is optimized for specific devices.

Application Code

Application developers must initialize all the above components and perform three key steps in the application code. For each processed frame, the state object is updated and the applications render method is called. The application developer must:

1. Query the state object for newly detected targets, markers or updated states of these elements
2. Update the application logic with the new input data
3. Render the augmented graphics overlay

Device Databases

Device databases are created using the online [Target Manager](#). The downloaded device target database assets contain an XML configuration file that allows the developer to configure certain trackable features and a binary file that contains the trackable database. These assets are compiled by the application developer into the app installer package and used at runtime by the Vuforia AR SDK.

Cloud Databases

Cloud databases can be created using the Target Manager or using the Vuforia Web Services API. Targets are queried at runtime of the application using the cloud recognition feature that performs a visual search in the cloud using sent camera images. In addition to the target data, the provisioned targets can contain metadata which is returned upon query.

User-Defined Targets

A fundamentally different supported approach is the user-defined targets feature. Rather than preparing targets outside of the developer, this feature allows for creating targets on-the-fly from the current camera image. A builder component is called to trigger the creation of a new user-target. The returned target is cached, but retained only for a given AR session.

Getting Started - Overview

Setting up the development environment

The first step for newcomers and experienced AR developers is to follow the Getting Started guide and run a Vuforia sample app. After that, depending on your skill and comfort level with AR, you may go directly to one of the following options:

- If you are familiar with the concepts of vision based augmented reality, you can look at the [Developer Guide](#) to learn more about Vuforia SDK features.
- If you are a "No text - unless it starts with //" type of person, the [API Reference](#) is the right place for you!

Vuforia deployment versions

Vuforia is offered as Android and iOS native SDK and as Unity Extension for both platforms. Development is supported in different development environments. The table below depicts existing combinations.

Development Environment	Development Platform			
	Native SDK		Unity Extension	
	Android	iOS	Android	iOS
Windows	Yes	--	Yes, multi-platform deployment	
MacOS	Yes	Yes	Yes, multi-platform deployment	
Linux	Yes	--	--	--

To set up your development environment for the chosen deployment platform, refer to the individual Getting Started guides below. The beginning of each section specifies exactly which OS versions are supported.

- [Getting Started with the Android Native SDK](#)
- [Getting Started with the iOS Native SDK](#)
- [Getting Started with the Unity Extension](#)

Getting Started with the Android Native SDK

If you are an Android developer and already have the Android SDK and NDK installed, go directly to [Step 2: Installing the Vuforia SDK](#). If you are new to Android software development, start at [Step 1: Setting Up the Development Environment](#).

1. [Setting Up the Development Environment](#)
2. [Installing the Vuforia SDK](#)
3. [Compiling and Running a Vuforia Sample App](#)

Step 1: Setting Up the Development Environment

Android SDK

Supported development platforms

- The Vuforia SDK supports Android OS 2.2 and above.
- The recommended development environment is Microsoft Windows 7 32/64-bit or Windows XP.
- The components to build the actual code (JDK, Eclipse+ADT and gcc) are available across multiple platforms. While building on Linux Ubuntu and Mac OS X environments is possible, we are unable to support those platforms with detailed documentation. However, we have included hints and notes to platform-specific issues that will help you set up your development environment on Linux Ubuntu 10.10 or Mac OS X 10.6 (Snow Leopard). This setup guide has been written for the Win7 32/64-bit platform with special notes for other operating systems.

NOTE: If you have already set up the Android SDK and NDK, go directly to [Step 2: Installing the Vuforia SDK](#).

Set up the Android development environment

The Vuforia SDK requires the Android SDK and the NDK. The Android 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 (JNI).

To set up the development environment, install these components in the following order, using the specified versions of the tools with the Vuforia SDK v2.0:

Component	Version
JDK	Java SE 7u9
Eclipse IDE	Latest version
Android SDK Downloader	Android SDK Tools revision 21
Android ADT	Latest version that is for SDK tools rev 21
Android SDK platform support	Android SDK Platform tools revision 21
Cygwin Environment	Latest version 1.7.17-1
Android NDK	Android NDK r8c

JDK

1. Download the **Java SE Development Kit** (JDK) from this site: <http://www.oracle.com/technetwork/java/javase/downloads/>
2. Click **Download** from the JDK section of the 'Java Platform, Standard Edition' table.
3. Install the JDK environment with default settings.

Detailed installation instructions and system requirements can be found at <http://www.oracle.com/technetwork/java/javase/index-137561.html>

MAC: The JDK is already integrated into the Mac OS X operation system.

Eclipse IDE

To install the Eclipse IDE:

1. Download the latest version of Eclipse IDE for Java Developers from: <http://www.eclipse.org/downloads/>
2. Unpack the downloaded ZIP package, and copy the contents of the archive starting with the subdirectory 'eclipse' to your program directory path in C:\Program Files\eclipse. You can also create a shortcut to eclipse.exe on your desktop or start menu.
3. Start `eclipse.exe`.

The first time that Eclipse is started, the IDE asks you to assign storage space for your workspace. This directory contains ONLY IDE-specific settings and information. This is not your application development workspace (despite the notice in this window). This type of information is typically stored in the user's home directory, which is the default value here: C:\Users\USERNAME\workspace. Check **Use this as the default and do not ask again**.

MAC: For MAC OS X, we recommend using the 32-bit version of Eclipse.

Linux: Eclipse is not always able to determine the location of the JVM, despite being in the path. To fix this, insert into `<path-to-eclipse-dir>/eclipse.ini` the following line at the top:

```
-vm <path-to-JVM>
```

For example:

```
-vm /user/bin/java
```

Android SDK downloader

The Android SDK is distributed through an SDK starter package containing the SDK tools.

1. Download the starter package from: <http://developer.android.com/sdk/index.html>
2. Unzip the archive and copy the contents into a directory, e.g., C:\Development\Android\android-sdk-windows\.

Throughout the Getting Started guide, we refer to the base directory of your development environment as `<DEVELOPMENT_ROOT>= c:\Development\Android`.

NOTE: Do not use pathnames with spaces.

3. Add the `tools\` directory to your Windows path.

- Right-click **Computer** on the desktop and select **Properties**.
- Use the Advanced system settings to open the System Properties window, and select **Environment Variables** on the Advanced tab.
- Look for the variable `Path` in the System variables window. Press **Edit**.
- Scroll to the end of the Variable value, and add the full path to the `tools\` directory to the end of the path, separated by a semicolon from the path before. For example:

```
;C:\Development\Android\android-sdk-windows\tools\
```

NOTE: The last `"\"` at the end of the `Path` variable has to be included.

Android ADT installation in the next step uses this path to identify the Android development environment.

Troubleshooting

For troubleshooting Android-related issues and for more detailed instructions on the Android SDK setup, refer to:

- [Installing the Vuforia SDK](#)
- [Get the Android SDK](#)

MAC: Update the `PATH` variable to point to the 'make' utility and the Android SDK tools directory in the `/etc/rc.common` or `~/bash_profile` file:

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:~/Development/Android/android-sdk-macosx/tools;export PATH
```

LINUX: Update your `PATH` to point to the 'make' utility and the Android SDK tools directory. If you use bash shell, add the following to the `~/bashrc` :

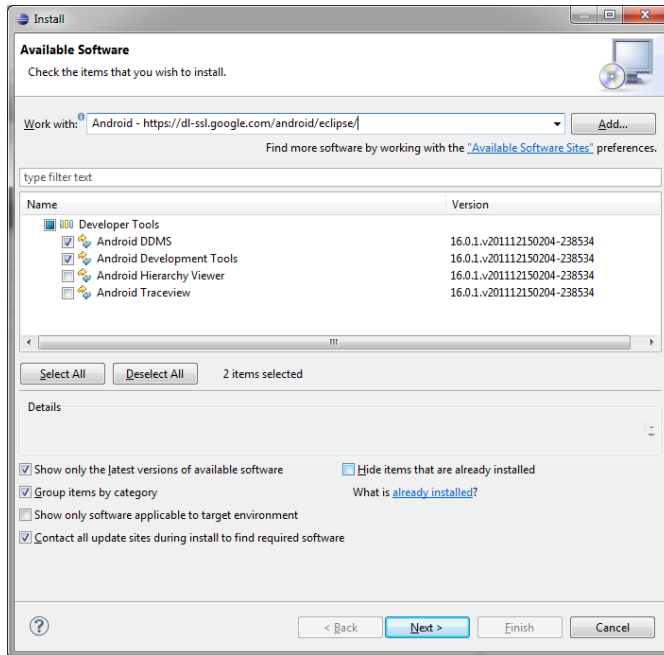
```
export PATH=/opt/android-sdk-linux/tools/;$PATH
```

Android ADT

Android Development Tools (ADT) are a powerful extension to Eclipse that connect it with the Android SDK and help with app development. This package is installed from within Eclipse.

1. Select `Help->Install New Software...`
2. Add the url: <https://dl-ssl.google.com/android/eclipse/> into the `Work with` field. Eclipse will ask you to provide an arbitrary name for the update site. Soon, `Developer Tools` appears in the field.

3. At a minimum, select **Android Development Tools** and **Android DDMS**, which adds debugging support, from the list.
4. Click **Next >**.



Eclipse component selection

After reviewing this selection and accepting the license terms, the downloader fetches the files and puts them in your Eclipse directory. After accepting the certificate and an automatic restart of Eclipse, the installation of the Android ADT is complete.

Android SDK platform support

To develop for Android, support for the appropriate Android platform must be installed. The Android SDK Manager is used to install additional components and support for different platforms.

1. In Eclipse, select the menu Window->Android SDK Manager. If the Android SDK location was not set up correctly within Eclipse, go to Windows-> Preferences->Android, and set the SDK location field to the root of your SDK install directory.
2. In the Android SDK manager window, sort by API level, click **Deselect All** and check the following boxes:

From Tools:

- Android SDK Platform-tools

From Android 4.1 (API 16):

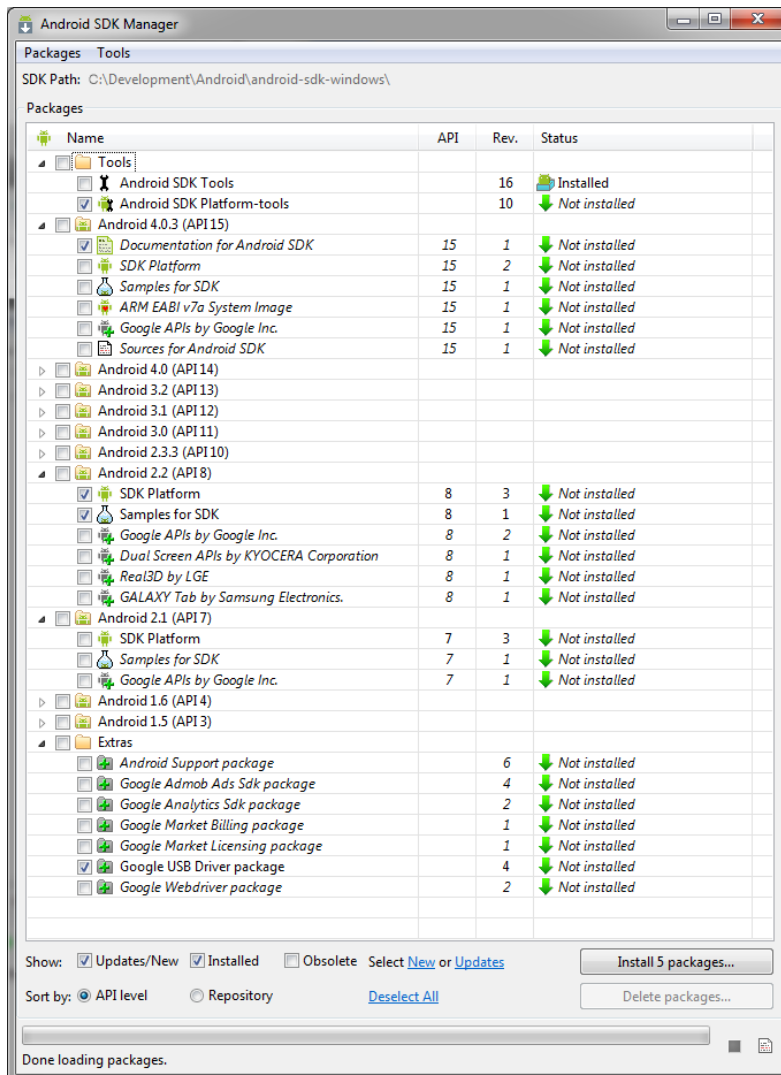
- Documentation for Android SDK

From Android 2.2 (API 8):

- SDK Platform
- Samples for SDK (optional)

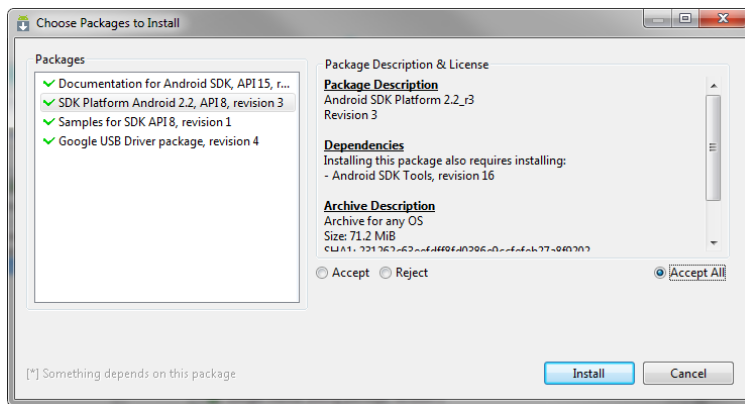
From Extras:

- Google USB Driver package (not compatible with Linux)



Android SDK component selection

3. To install the selected options, click **Install 5 packages...** and accept all licenses on the next window. Use `Accept All` as a shortcut and click **Install**.



Android component license acceptance

4. Add the platform-tools\ directory to your Windows path: ;C:\Development\Android\android-sdk-windows\platform-tools\

NOTE: The last "\" at the end of the Path variable has to be included.

MAC: Update the PATH variable to point to the Android SDK Platform-tools directory in the /etc/rc.common file or ~/.bash_profile:

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:~/Development/Android/android-sdk-macosx/tools:~/Development/Android/android-sdk-macosx/platform-tools:export PATH
```

LINUX: Update your PATH to point to the Android SDK Platform-tools directory. If you use bash shell, add the following to ~/.bashrc :

```
export PATH=/opt/android-sdk-linux/tools:/opt/android-sdk-linux/platform-tools:PATH
```

The Eclipse environment is now ready for Android development.

Cygwin environment

LINUX: This section is not relevant for Linux users who have GNU make installed and in their path.

MAC: This section is not relevant for Mac users who have Apple Developer Tools (XCode) installed. Install XCode if necessary from <http://developer.apple.com/xcode/>.

A GNU compiler is required to compile dynamic applications as shared libraries for the Android NDK. Android make files are designed to run with gcc4. On Windows, a convenient way to have the complete environment prepared for this, is to install Cygwin.

Cygwin uses an installer helper to manage the installation process.

1. Go to <http://www.cygwin.com/setup.exe> and select "Install from the Internet!" when prompted at "Choose A Download Source" in the installer. We recommend not changing the Root Directory in the next window, and leaving it at "C:\cygwin." The "Local Package Directory" holds the downloaded packages. You may want to keep them with the

downloaded `Setup.exe` file in the same directory to have a Cygwin installer directory. Choose a download site with a known fast connection near you.

When the package information is downloaded you will see a hierarchical browser to select packages.

2. Select the following package from the hierarchy for download: All -> Devel -> "make: The GNU version of the 'make' utility"
3. Select the word **skip** to change it to the actual version number, which is currently 3.82-90.
4. Finish the installation by clicking **Next**.

Your Cygwin environment is fully set up to work with the QCAR SDK. If you have other similar environments installed, make sure to set your Windows path variable to point to "C:\cygwin\bin" so that bash uses this version of GNU's `make.exe`.

Android NDK

The Android NDK is an extension of the Android SDK that lets Android developers build performance-critical parts of their applications in native code.

1. Download the NDK package from <http://developer.android.com/sdk/ndk/index.html>
2. Unzip the archive, and copy the contents into a directory. To be consistent with our previous setup, we recommend putting the contents in "C:\Development\Android\android-ndk-r8\." Thus, Android SDK and Android NDK share the same parent directory. Later, we will add the Vuforia SDK and your project files.

NDK requires the above directory to be added to the Windows path.

3. Right-click **My Computer** on the desktop and select **Properties**.
4. On the Advanced tab, select **Environment Variables** and look for the Variable "Path" in the System variables window.
5. After pressing **Edit**, scroll to the end of "Variable value:" and add the full path to the directory to the end of the path, separated by a semicolon from the previous path. In the above example, you would add:

```
;C:\Development\Android\android-ndk-r8\
```

NOTE: Path has a semicolon at the beginning. **Do not use pathnames with spaces.** Alternatively, you can set a User variable with the name `Path`, but this is valid only for the current user. The last "\" at the end of the `Path` variable has to be included.

6. To test your installation, compile any of the NDK sample applications. Using a Cygwin bash shell, navigate to the root directory of any demo application (e.g., for the 'san-angeles' sample app without the installation path above):

```
cd /cygdrive/c/Development/Android/android-ndk-r8/samples/san-angeles
ndk-build
```

The compiler should produce a dynamically linked library `libsanangeles.so` and write it to `/libs/armeabi` within the application directory. NDK includes support for different architectures so you might find different subdirectories in `/libs`.

Now your development environment is ready to host Vuforia SDK-related content.

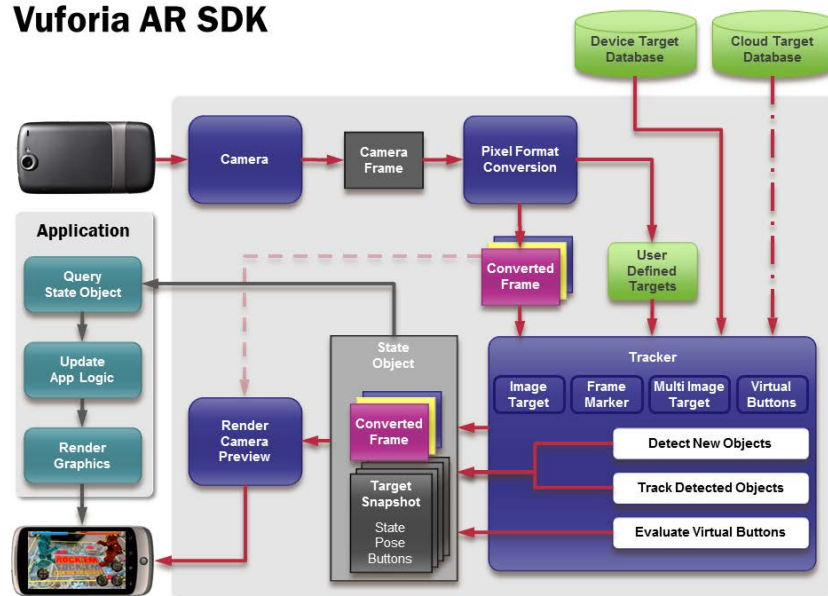
MAC: Update the `PATH` variable to point to the Android NDK directory in the `/etc/rc.common` file or `~/ .bash_profile`:

```
PATH=/bin:/sbin:/usr/bin:/usr/sbin:/usr/libexec:/System/Library/CoreServices:/Developer/usr/bin:~/Development/Android/android-sdk-macosx/tools:~/Development/Android/android-sdk-macosx/platform-tools:~/Development/Android/android-ndk-r8:export PATH
```

LINUX: Update your `PATH` to point to the Android NDK Platform directory. If you use bash shell, add the following to the `~/.bashrc` file:

```
export PATH=~/bin:/opt/android-sdk-linux_x86/tools/:/opt/android-ndk-r8:$PATH
```

Vuforia AR SDK



Data flow diagram of the Vuforia AR SDK in an application environment

Step 2: Installing the Vuforia SDK

Clean installation

The Vuforia SDK is distributed using installers for the following platforms: Windows, Mac OS X, and Linux.

To start developing with the Vuforia SDK:

- Download and install the Vuforia SDK under `<DEVELOPMENT_ROOT>`
- Adjust the Vuforia Environment settings in Eclipse

After accepting the license agreements, the installer creates a directory structure in your Android development environment. This structure ensures that sample apps can be easily built and deployed using the Android NDK and the Eclipse Java developer environment.

Upgrading from a previous version

- When installing an updated version of the Vuforia SDK, we recommend installing it under a new directory tree, as the installer recommends.
- To compile the new set of sample applications it is required to adjust the Eclipse workspace variable setting.
- Finally, you need to move your projects under the new `<DEVELOPMENT_ROOT>` (see below). See the [Transition/Migration Guide](#) for API changes and migration instructions.

Resulting directory structure

To streamline development we have designed a directory structure that keeps the Vuforia SDK and your applications in separate trees. This will ensure easier updates to the SDK, leaving your application source-trees untouched.

In the previous steps we used a starting directory for the SDK and the NDK installation that we called `<DEVELOPMENT_ROOT>= C:\Development\Android`

The downloaded installer creates a directory structure that will integrate into your Android development environment. Following the recommended installation location the development environment should finally result in the structure below. Here `xx-yy-zz` stands for the version number of the Vuforia SDK.

```
<DEVELOPMENT_ROOT>\
```

```
android-ndk-r8\
```

```
android-sdk-windows\
```

```
vuforia-sdk-android-xx-yy-zz\
```

```
    build\
```

Vuforia Augmented Reality SDK

```
    licenses\
```

License Agreements

```
    samples\
```

Sample applications with full source code

assets\
readme.txt

Additional assets required to use Vuforia SDK
Starting read-me document

Install the Vuforia SDK

Download

The Vuforia SDK is distributed through the [Vuforia Developer Portal](#). To access the installer, follow these instructions.

Note: Although we distribute installers for the platforms listed below, support for Vuforia SDK development is limited to the Win 7 32/64-bit platform.

Windows

Although we recommend that you develop on Windows 7 32/64-bit or Windows XP 32-bit environments, the Vuforia SDK has also been successfully run on Windows XP 64-bit. However, we currently do not offer specific support for this platform.

1. Download the installer EXE-file from the Downloads page.
2. Run the installer.

Mac OS X

Installation has been tested on Mac OS X 10.6 and OS X 10.7.

1. Download the archive file from the Downloads page.
2. Unarchive and run the installer.

Linux

Installation has been tested on Ubuntu 11.10.

1. Make sure JVM is installed.
2. Download the installer file from the Downloads page.
3. Open a terminal window and change the directory to the installer file location.
4. Change the mode of the installer to be executable.
5. Execute the installer file on the command line.

Set the QCAR environment variable

Our recommended directory structure allows for SDK upgrades independent of application development. Your future AR projects will be in the recommended structure under <DEVELOPMENT_ROOT> . One workspace variable must be set in Eclipse so it is aware of this hierarchy:

1. In Eclipse, go to Window->Preferences.
2. Navigate to Java->Build Path->Classpath Variables using the hierarchy browser.
3. Create a new variable by selecting **New . . .**
4. Add:

QCAR_SDK_ROOT

into the `Name:` field, and navigate using `Folder...` to the `<DEVELOPMENT_ROOT>\vuforia-sdk-android-xx-yy-zz` directory we defined in the Vuforia SDK setup section. In our example above, the variable value is:

`C:/Development/Android/vuforia-sdk-android-xx-yy-zz`

where `xx-yy-zz` denotes the current Vuforia SDK version number.

This setting is essential as the classpath settings in the sample files use this reference to include common shared JAR files.

MAC: The preferences menu on Mac OS X is under `Eclipse->Preferences`.

Set up the build path

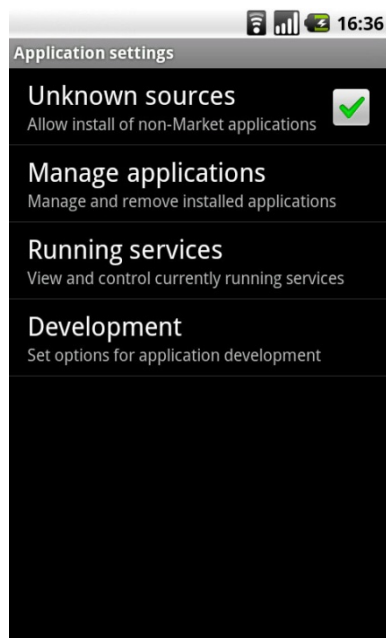
Navigate in the package properties (right-click on package -> Properties) to the **“Order and Export”** tab after selecting the **“Java Build Path”** page. Verify whether the checkmark is included before the `QCAR/SDK_ROOT/build/java/QCAR/QCAR.jar`. If not, set it to ensure `QCAR.jar` is packaged with your app.

Prepare test device for development

Developer settings on the device

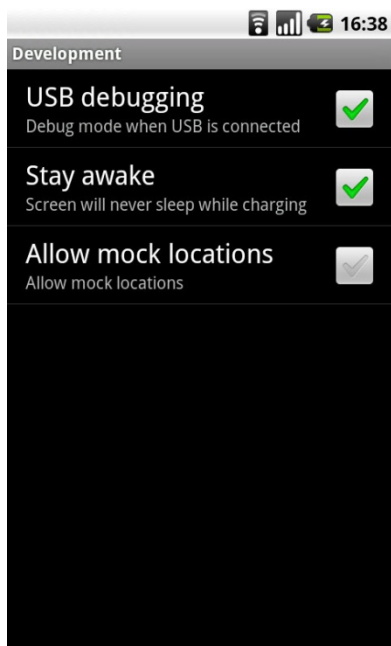
Android devices require special settings for development. In this step, we will:

- Allow installing apps from unknown sources
 - Enable USB debugging
1. On the device, go to `Settings->Applications` and choose **Unknown sources** as shown here. Accept the warning shown on the right. This setting allows direct installation of unsigned APKs from within Eclipse.



Allowing unknown sources for software installation

2. Go to the `Development` dialog on the screen above, and choose the two settings as shown here:



USB debugging is mandatory; the stay awake setting helps with development

Install the USB driver

Connect your device to the development PC using the USB cable.

On initial connection, Windows recognizes the new device and attempts to locate a compatible driver. The Android SDK already includes some USB drivers - others can be obtained directly from the device manufacturer. SDK pre-packaged drivers can be located in the following directory:

```
<DEVELOPMENT_ROOT>\android-sdk-windows\extras\google\usb_driver
```

When the device installation finishes, you are ready to use your test device.

On some machines the USB driver installation asks you to reboot the machine. You can skip this step and the device should be accessible without a reboot.

NOTE: Contact the device manufacturer to ensure that you are using the correct USB driver.

Step 3: Compiling and Running a Vuforia Sample App

You are now going to build a sample application included in the Vuforia SDK package.

The `ImageTargets` application is a good place to start learning about the SDK, as it shows detection and tracking of natural features using common images. This section explains how to:

- Build the native C++ source files with the NDK package of the Android SDK
- Use Eclipse to build the Java sources and create the APK package that can be deployed to the device.



Compile the shared object

Compile the shared object with application code

Android NDK applications are deployed as shared objects with a Java bootloader. In this step we create the binary `*.so` file that is later packaged through Eclipse. For each sample app you will need to build the application `*.so` file separately.

Build ImageTargets shared library

Change directory to `ImageTargets` in the `<DEVELOPMENT_ROOT>\vuforia-sdk-android-xx-yy-zz\samples` directory and execute:

ndk-build

The output displayed on the console should look like:

```
1  Gdbserver      : [arm-linux-androideabi-4.4.3] libs/armeabi/gdbserver
2  Gdbsetup       : libs/armeabi/gdb.setup
3  Gdbserver      : [arm-linux-androideabi-4.4.3] libs/armeabi-v7a/gdbserver
4  Gdbsetup       : libs/armeabi-v7a/gdb.setup
5  Compile++ arm  : ImageTargets <= ImageTargets.cpp
6  Compile++ arm  : ImageTargets <= SampleUtils.cpp
7  Compile++ arm  : ImageTargets <= Texture.cpp
8  StaticLibrary  : libstdc++.a
9  Prebuilt       : libQCAR.so <= jni/../../../../build/lib/armeabi/
10 SharedLibrary  : libImageTargets.so
11 Install        : libImageTargets.so => libs/armeabi/libImageTargets.so
12 Install        : libQCAR.so => libs/armeabi/libQCAR.so
13 Compile++ arm  : ImageTargets <= ImageTargets.cpp
14 Compile++ arm  : ImageTargets <= SampleUtils.cpp
15 Compile++ arm  : ImageTargets <= Texture.cpp
16 StaticLibrary  : libstdc++.a
17 Prebuilt       : libQCAR.so <= jni/../../../../build/lib/armeabi-v7a/
18 SharedLibrary  : libImageTargets.so
19 Install        : libImageTargets.so => libs/armeabi-v7a/libImageTargets.so
20 Install        : libQCAR.so => libs/armeabi-v7a/libQCAR.so
```

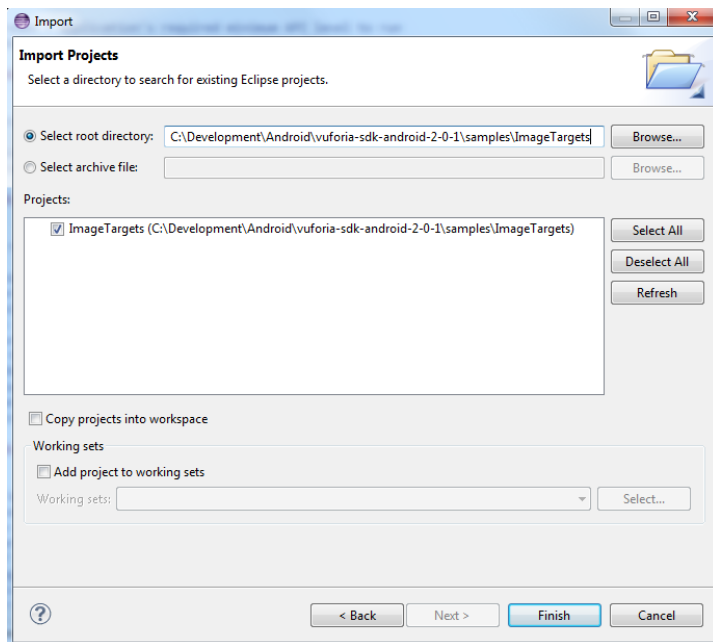
You are now ready to compile the Java bootloader and deploy the application.

Create the Android APK

In the previous step, we created the shared object for the `ImageTargets` sample app. We must now compile and build the Android application APK using the Eclipse IDE environment. The sample applications have several Java classes to create the bootloader, provide GUI functionality, read the orientation sensor of the device, allow user interaction, and provide video background for the app.

Build the ImageTargets application

1. Create a new project in Eclipse by selecting **File->Import->General->Existing Projects intoWorkspace....**
2. Click **Next**.
3. Select the **<DEVELOPMENT_ROOT>\vuforia-sdk-android-xx-yy-
zz\samples\ImageTargets** directory in the “**Select root directory**” field.
4. Click **Finish**.



Eclipse "Import project" panel




The standard installation of Eclipse has automatic compilation turned on. The first time you compile the app, you may receive some error messages from Eclipse. `Project->Clean...->Clean All Projects` should fix these errors, as Eclipse has to build some standard directories. This also creates the APK package for deployment.

If a project still looks like it is in an Error state, right-click the project itself, and select **Properties->Android Tools->Fix project properties...**

Run the sample application

Print the image target

1. Print all image target PDFs in `<DEVELOPMENT_ROOT>\vuforia-sdk-android-xx-yy-zz\samples\ImageTargets\media` from any of the formats onto US Letter or A4-size paper with the page scaling '**none**' option.
2. Make sure you keep the original aspect ratio of the image intact.

	
“chips” image target	“stones” image target
	
“tarmac” image target	

Deploy and run the application

1. With the device connected, select the **"ImageTargets"** project from the Package Explorer in Eclipse.
2. Choose **Run->Run As->Android Application**.

Eclipse automatically installs the app to a connected Android device using ADB and starts running it on the device.

NOTE: If you get an error at application startup, such as “Application has stopped unexpectedly”, and if you are using Android SDK Tools r17 (or above), you may need to execute the following steps in Eclipse:

1. Right-click the project, and select **Properties > Java Build Path > Order and Export**.

2. Activate the checkbox next to the QCAR JAR file.
3. Close the project properties.
4. Clean and rebuild the project.

The `ImageTarget` sample shows a splash screen on your device, followed by an introductory screen with some basic instructions and a **Start** button at the bottom.

1. Click **Start** on the screen to move on with the application; at this point the live camera image should appear on your device screen.
2. Hold your device up against the printed "**chips**" target to see this image.



You have successfully deployed your first application with the Vuforia SDK!

3. Use your device to look at the printed target. You should see a textured teapot centered on top of the target, registered to the plane.
4. Press **menu**, and select **Switch to Tarmac dataset**.
5. Hold your device up against the **tarmac** printed target. You will see the teapot, but now it is orange.

The `ImageTargets` app shows how the SDK can detect and track a single image from a pre-defined set of images. The app also shows how to switch between datasets without too much effort, as well as how to deal with camera settings, such as autofocus. With a small code change, it can also be compiled to detect and track multiple images simultaneously. Check the [Knowledge Base Articles](#) section of the Developer Guide for more information.

Troubleshooting

1. If you encounter problems on the installation, check the device connection settings in **Prepare Test Device for Development**. In Eclipse, you can see if the device is connected correctly via the ADB in the DDMS panel (Window->Open Perspective->DDMS). The device must be listed under **Devices**.
2. Alternatively, open a Cygwin bash shell and execute:

```
adb devices
```

The output should show the attached device:

```
$ adb devices
```

```
List of devices attached  
HT012P123456 device
```

3. If the device list is empty, or a given device is not listed, kill the ADB server by executing:

```
adb kill-server
```

4. Execute 'adb devices' again to restart the server and re-detect devices.

Getting Started with the iOS Native SDK

If you are an iOS developer and have already set up your iOS development environment, go directly to [Step 2: Installing the Vuforia SDK](#). Note that iOS SDK 6.0 or above with XCode 4.5 or later is recommended for use with Vuforia.

If you are new to iOS software development, start at [Step 1: Setting up the Development Environment](#).

1. [Setting up the Development Environment](#)
2. [Installing the Vuforia SDK](#)
3. [Installing the Vuforia Samples](#)
4. [Compiling & Running a Vuforia sample app](#)

If you were part of the Vuforia SDK Beta program, please review the following sections of the DevGuide – [Publish with Vuforia](#) and [Transition/Migration Guide](#).

Step 1: Setting up the Development Environment iOS SDK

The Vuforia SDK for iOS currently supports the majority of iOS devices including:

- iPhone 3GS
- iPhone 4/4S
- iPhone 5
- iPad 2, 3 & 4
- iPad mini
- iPod Touch 4G and later

The recommended development platform is Mac OS X Lion or Mountain Lion

Follow the steps below to:

1. Become an iOS Developer
2. Install XCode and the iOS SDK
3. Set up a Provisioning Profile

Become an iOS developer

Vuforia applications must be deployed to a device to run; they cannot be run in the iOS simulator. You must enroll in the iOS Developer program to deploy applications to an iOS device. You can enroll here: <http://developer.apple.apple.com/programs/ios>

Install XCode and the iOS SDK

Once you have enrolled in the iOS Developer program, you should have access to the iOS Dev Center: <http://developer.apple.com/devcenter/ios>

Go to the Dev Center or the Mac AppStore to download XCode. The download should include the latest version of the iOS SDK.

We have tested Vuforia with XCode 4.5 and iOS SDK 6.0 or above; however, it should be possible to set the Deployment Target to 4.3 or above from within XCode.

Set up a Provisioning Profile

From the iOS Dev Center, you can go to your iOS Provisioning Portal, where you can follow the steps to set up your development machine to build and deploy iOS applications. Also, follow the process to:

- Obtain a certificate
 - Assign a device
 - Create an App ID
 - Create a provisioning profile
1. When choosing an App ID, pick something generic enough to work throughout your development process. See the Apple Developer Provisioning Portal for more details about choosing App IDs.
 2. Create a Development Provisioning Profile using the App ID you created, and be sure to add the devices you wish to test.
 3. Download the provisioning profile to your computer and double-click it to install.
 4. Test that a simple non-Vuforia application, possibly one of the samples in the iOS SDK, can run on the device. If you experience problems, it is a good idea to search the web for solutions, as there is plenty of advice available.

Step 2: Installing the Vuforia SDK iOS

The Vuforia SDK has been tested on 10.7 Lion and 10.8 Mountain Lion.

1. Download the archive file from the Downloads page.
2. Unarchive and run the installer.

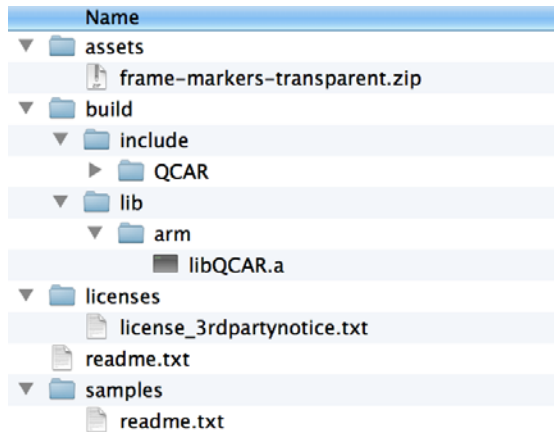
If you are installing on Mountain Lion, it is possible that the MacOS GateKeeper may object, saying that there are problems with the download. To overcome this:

1. Go to Preferences → Security&Privacy → General.
2. Unlock the key and allow applications downloaded from "anywhere."
3. You may encounter the JavaVM being disabled. Enable it because the installer is a Java component.

Resulting directory structure

The directory structure resulting from installation is shown in this image where the SDK has been installed in this directory:

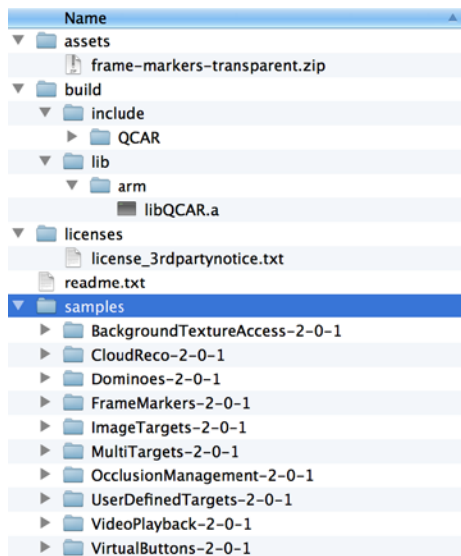
```
<DEVELOPMENT_ROOT>/Vuforia-ios-xx-yy-zz.
```



We recommend developing your own applications outside of the SDK directory, to make it easier to upgrade to future versions of the SDK.

Step 3: Installing the Vuforia Samples

1. Download one or more of the samples that come in a compressed archive and expand these into their own directory.
2. Copy the samples into the Vuforia installation directory so it looks like this:



Step 4: Compiling and Running a Vuforia Sample App

We are now going to build a sample application included in the Vuforia SDK package.

The `ImageTargets` application is a good place to start learning about the SDK as it shows detection and tracking of natural features using common images. This section shows you how to build the project in XCode and deploy to the device.

Configuring the project in XCode

1. Open the `ImageTargets` XCode project, located in the `ImageTargets` directory as shown above.

Note: Vuforia applications do not build or work with the Simulator.

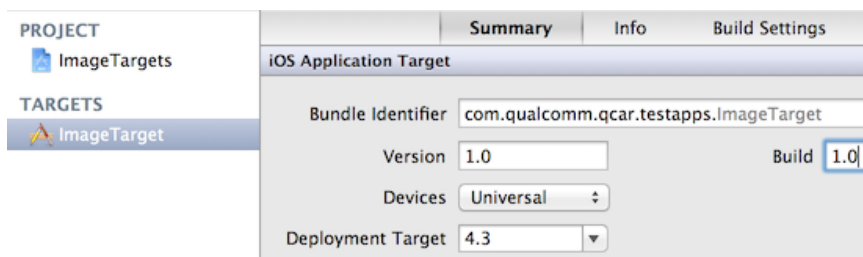
2. You must select a device to build and run any Vuforia application.
3. Ensure that your iOS device is connected to your computer via a USB cable and that XCode can recognize it as being usable for development (as highlighted in the Organizer window by a green light next to it).
4. Select the device via the dropdown selector at the top left of the XCode window as shown here:



Set the bundle identifier

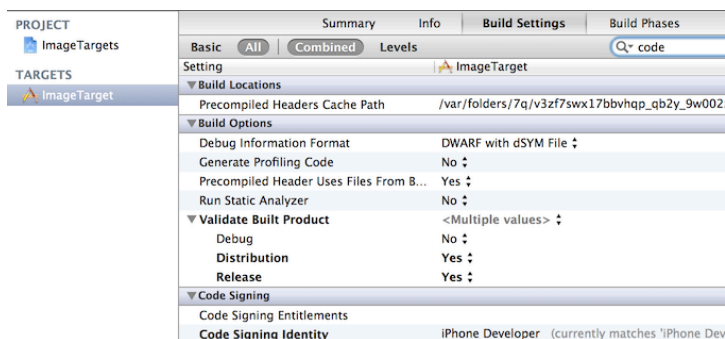
To set your bundle identifier so it matches the provisioning profile you have set up on the Apple Developer Provisioning Portal:

1. Click the **ImageTargets** target to make it active.
2. Replace `com.qualcomm.qcar.testapps` with your own bundle identifier, as shown here:



Set the Code Signing Identity

1. In Xcode, click the **Active Target**, and then click the **Build Settings** tab, as shown here:



2. Under Code Signing Identity, ensure that the right identity is selected. This usually happens by default, and if everything is OK, the message “currently matches” will display. This validates that the provisioning profile matches the code signing identity as set on the Apple Developer Portal.

Run the sample application

Print the image target

Print all image targets in `<DEVELOPMENT_ROOT>/vuforia-sdk-ios-xx-yy-
zz/samples/ImageTargets/media` from any of the formats onto a US Letter or A4-size paper with the page scaling 'none' option, keeping the original aspect ratio of the image intact.

Typically there are three image targets as shown here.



Chips image target



Stones image target



Tarmac image target

Deploy and run the application

1. Connect your iOS device to your computer via the USB connector.
2. Deploy the application to the device using the 'Build and Run' button in XCode, which will do exactly this and install it on the device, assuming there are no problems.

After a splash screen, the live camera image is shown.

3. Use your device to look at the 'chips' or 'stones' printed target, and you should see a textured teapot centered on top of the target, grounded to the plane.



You have successfully deployed your first application with the Vuforia SDK!

4. Double-tap the screen to reveal a menu, and select **Select Target→Tarmac** to switch datasets. The teapot will no longer appear on the Stones target. This is because the Tarmac image is stored in a different database. Now, point your device towards the **tarmac** printed target and you will see the teapot again.

The `ImageTargets` app shows how the SDK can detect and track a single image from a pre-defined set of images. The app also shows how to switch between databases without too much effort, as well as how to deal with camera settings, such as autofocus. With a small code change, it can also be compiled to detect and track multiple images simultaneously.

Developer Workflow

After creating an account in the Target Manager UI, the developer is presented with a choice: use device-based databases or cloud based databases. You will not be able to convert a database to the other type later on, so be clear on which capabilities you need.

Device databases provide Vuforia applications with a locally accessible database of image targets.

If you know what the application will be expected “see,” and the target set of images is fewer than 100, then a local device database is the right solution. Device database use cases include promotional campaigns for a specific set of products or print material, i.e., anything that will be identified and scanned with a lower set of images. Device databases receive more immediate responses since the images are stored locally.

Cloud databases provide Vuforia applications with a large number of targets. Cloud databases are stored in the Internet and support over a million image targets.

If the application supports a retail or catalog use case where there are more than 100 images, or the images are updated frequently, then a cloud database is the right solution for you. Cloud image targets may take a little longer to identify (depending on network connectivity), but they provide a strong image recognition capability.

To help make a decision, consider the following table:

Device Database	Cloud Database
Limited to 100 targets per downloaded device target database	>1 million targets in the cloud database
Allows downloading targets in different combinations	One database with all images and metadata
Downloaded targets are only for detection, no metadata support	Targets retrieved by cloud recognition can carry up to 150kB of metadata
Network connection on end user device not required for detection	Network connection on end user device required for recognition, network traffic for cloud recognition
Run-time detection response time within 2	Response time up to 3 seconds,

Device Database

to 3 frames

Multiple databases can be active (each with maximum 100 targets)

Free

Cloud Database

depending on network connectivity

Recognition on maximum 1 million targets active at any time

Free and Paid, depending on usage

Once you have decided between the two methods, use this chart to guide you in creating your Vuforia Augmented Reality application.

