Hello, Android

Whether you're an experienced mobile engineer, a desktop or web developer, or a complete programming novice, Android represents an exciting new opportunity to write innovative applications for mobile devices.

Despite the name, Android will not help you create an unstoppable army of emotionless robot warriors on a relentless quest to cleanse the earth of the scourge of humanity. Instead, *Android* is an open source software stack that includes the operating system, middleware, and key applications along with a set of API libraries for writing mobile applications that can shape the look, feel, and function of mobile handsets.

Small, stylish, and versatile, modern mobile phones have become powerful tools that incorporate cameras, media players, GPS systems, and touch screens. As technology has evolved, mobile devices have become about more than simply making calls, but their software and development platforms have struggled to keep pace.

Until recently, mobile phones were largely closed environments built on proprietary operating systems that required proprietary development tools. The phones themselves often prioritized native applications over those written by third parties. This has introduced an artifi cial barrier for developers hoping to build on increasingly powerful mobile hardware. In Android, native and third-party applications are written using the same APIs and executed on the same run time. These APIs feature hardware access, location-based services, support for background services, map-based activities, relational databases, interdevice peer-to-peer messaging, and 2D and 3D graphics.

Using this book, you will learn how to use these APIs to create your own Android applications. In this chapter, you'll learn some mobile development guidelines and be introduced to the features available from the Android development platform.

Android has powerful APIs, excellent documentation, a thriving developer community, and no development or distribution costs. As mobile devices continue to increase in popularity, this is an exciting opportunity to create innovative mobile phone applications no matter what your development background.

A Little Bac kground

In the days before Twitter and Facebook, when Google was still a twinkle in its founders' eyes and dinosaurs roamed the earth, mobile phones were just that — portable phones small enough to fi t inside a briefcase, featuring batteries that could last up to several hours; they offered the freedom to make calls without being physically connected to a landline.

Increasingly small, stylish, and powerful mobile phones are now as ubiquitous as they are indispensable. Hardware advancements have made mobiles smaller and more effi cient while including an increasing number of peripherals. Beginning with cameras and media players, mobiles now include GPS systems, accelerometers, and touch screens. While these hardware innovations should prove fertile ground for software development, the applications available for mobile phones have generally lagged behind the hardware.

The Not So Distant Past

Historically, developers, generally coding in low-level C or C++, have needed to understand the specifi c hardware they were coding for, generally a single device or possibly a range of devices from a single manufacturer. As hardware technology has advanced, this closed approach has struggled to keep pace. More recently, platforms like Symbian have been created to provide developers a wider target audience.

These systems have proved more successful in encouraging mobile developers to provide rich applications that better leverage the hardware available. These platforms offer some access to the device hardware, but require writing complex C/C++ code and making heavy use of proprietary APIs that are notoriously diffi cult to use. This difficulty is amplifi ed when developing applications that must work on different hardware implementations and is particularly true when developing for a particular hardware feature like GPS. In recent years, the biggest advance in mobile phone development has been the introduction of Javahosted MIDlets. MIDlets are executed on a Java virtual machine, abstracting the underlying hardware and letting developers create applications that run on the wide variety of hardware that supports the Java run time. Unfortunately, this convenience comes at the price of restricted access to the device hardware.

In mobile development, it's considered normal for third-party applications to receive different hardware access and execution rights compared to native applications written by the phone manufacturers, with MIDlets often receiving few of either.

The introduction of Java MIDlets has expanded developers' audiences, but the lack of low-level hardware access and sandboxed execution have meant that most mobile applications are desktop programs designed to run on a smaller screen rather than take advantage of the inherent mobility of the handheld platform.

The Future

Android sits alongside a new wave of mobile operating systems designed for increasingly powerful mobile hardware. Windows Mobile and Apple's iPhone now provide a richer, simplifi ed development environment for mobile applications. However, unlike Android, they're built on proprietary operating systems that often prioritize native applications over those created by third parties and restrict communication among applications and native phone data. Android offers new possibilities for mobile applications by offering an open development environment built on an open source Linux kernel. Hardware access is available to all applications through a series of API libraries, and application interaction, while carefully controlled, is fully supported.

In Android, all applications have equal standing. Third-party and native Android applications are written using the same APIs and are executed on the same run time. Users can remove and replace any native application with a third-party developer alternative; even the dialer and home screens can be replaced.

What It Is n't

As a disruptive addition to a mature fi eld, it's not hard to see why there has been some confusion about what exactly Android is. Android is **not**:

❑ **A Java ME implementation** Android applications are written using the Java language, but they are not run within a Java ME virtual machine, and Java-compiled classes and executables will not run natively in Android.

❑ **Part of the Linux Phone Standards Forum (LiPS) or the Open Mobile Alliance (OMA)** Android runs on an open source Linux kernel, but, while their goals are similar, Android's complete software stack approach goes further than the focus of these standards defining organizations.

❑ **Simply an application layer (like UIQ or S60)** While it does include an application layer, "Android" also describes the entire software stack encompassing the underlying operating system, API libraries, and the applications themselves.

❑ **A mobile phone handset** Android includes a reference design for mobile handset manufacturers, but unlike the iPhone, there is no single "Android Phone." Instead, Android has been designed to support many alternative hardware devices.

❑ **Google's answer to the iPhone** The iPhone is a fully proprietary hardware and software platform released by a single company (Apple), while Android is an open source software stack produced and supported by the Open Handset Alliance and designed to operate on any handset that meets the requirements. There's been a lot of speculation regarding a Google-branded Android phone, but even should Google produce one, it will be just one company's hardware implementation of the Android platform.

An Open Platform for Mobile Development

Google describes Android as:

The fi rst truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.

http://googleblog.blogspot.com/2007/11/wheres-my-gphone.html

Android is made up of several necessary and dependent parts including the following:

❑ A hardware reference design that describes the capabilities required of a mobile device in order to support the software stack

❑ A Linux operating system kernel that provides the low-level interface with the hardware, memory management, and process control, all optimized for mobile devices

❑ Open source libraries for application development including SQLite, WebKit, OpenGL, and a media manager

❑ A run time used to execute and host Android applications, including the Dalvik virtual machine and the core libraries that provide Android specific functionality. The run time is designed to be small and effi cient for use on mobile devices.

❑ An application framework that agnostically exposes system services to the application layer, including the window manager, content providers, location manager, telephony, and peer-to-peer services

❑ A user interface framework used to host and launch applications

❑ Preinstalled applications shipped as part of the stack

❑ A software development kit used to create applications, including the tools, plug-ins, and documentation

At this stage, not all of the Android stack has been released as open source, although this is expected to happen by the time phones are released to market. It's also worth noting that the applications you develop for Android do not have to be open source.

What really makes Android compelling is its open philosophy, which ensures that any defi ciencies in user interface or native application design can be fi xed by writing an extension or replacement. Android provides you, as a developer, the opportunity to create mobile phone interfaces and applications designed to look, feel, and function exactly as you image them.

Nativ e Android Ap plication s

Android phones will normally come with a suite of preinstalled applications including, but not limited to:

❑ An e-mail client compatible with Gmail but not limited to it

❑ An SMS management application

❑ A full PIM (personal information management) suite including a calendar and contacts list, both tightly integrated with Google's online services

❑ A fully featured mobile Google Maps application including StreetView, business fi nder, driving directions, satellite view, and traffi c conditions

❑ A WebKit-based web browser

❑ An Instant Messaging Client

❑ A music player and picture viewer

❑ The Android Marketplace client for downloading thied-party Android applications.

❑ The Amazon MP3 store client for purchasing DRM free music.

All the native applications are written in Java using the Android SDK and are run on Dalvik. The data stored and used by the native applications — like contact details — are also available to thirdparty applications. Similarly, your applications can handle events such as an incoming call or a new SMS message. The exact makeup of the applications available on new Android phones is likely to vary based on the hardware manufacturer and/or the phone carrier or distributor. This is especially true in the United States, where carriers have signifi cant infl uence on the software included on shipped devices.

Android SDK F eatures

The true appeal of Android as a development environment lies in the APIs it provides. As an application-neutral platform, Android gives you the opportunity to create applications that are as much a part of the phone as anything provided out of the box. The following list highlights some of the most noteworthy Android features:

❑ No licensing, distribution, or development fees

❑ Wi-Fi hardware access

❑ GSM, EDGE, and 3G networks for telephony or data transfer, allowing you to make or receive calls or SMS messages, or to send and retrieve data across mobile networks

❑ Comprehensive APIs for location-based services such as GPS

❑ Full multimedia hardware control including playback and recording using the camera and microphone

❑ APIs for accelerometer and compass hardware

❑ IPC message passing

❑ Shared data stores

❑ An integrated open source WebKit-based browser

❑ Full support for applications that integrate Map controls as part of their user interface

❑ Peer-to-peer (P2P) support using Google Talk

❑ Mobile-optimized hardware-accelerated graphics including a path-based 2D graphics library and support for 3D graphics using OpenGL ES

❑ Media libraries for playing and recording a variety of audio/video or still image formats

❑ An application framework that encourages reuse of application components and the replacement of native applications

Access to Hardware including Camera, GPS, and Accelerometer

Android includes API libraries to simplify development involving the device hardware. These ensure that you don't need to create specific implementations of your software for different devices, so you can create Android applications that work as expected on any device that supports the Android software stack.
The Android SDK includes APIs for location-based hardware (such as GPS), camera, network connections, Wi-Fi, Bluetooth, accelerometers, touch screen, and power management. You can explore the possibilities of some of Android's hardware APIs in more detail in Chapter 10.

Native Google Maps, Geocoding, and Location-Based Services

Native map support lets you create a range of map-based applications that leverage the mobility of Android devices. Android lets you create activities that include interactive Google Maps as part of your user interface with full access to maps that you can control programmatically and annotate using Android's rich graphics library.

Android's location-based services manage technologies like GPS and Google's GSM cell-based location technology to determine the device's current position. These services enforce an abstraction from specific location-detecting technology and let you specify minimum requirements (e.g., accuracy or cost) rather than choosing a particular technology. It also means that your location-based applications will work no matter what technology the host handset supports.

To combine maps with locations, Android includes an API for forward and reverse geocoding that lets you fi nd map coordinates for an address, and the address of a map position.

You'll learn the details of using maps, the geocoder, and location-based services in Chapter 7.

Background Services

Android supports applications and services designed to run invisibly in the background.

Modern mobiles are by nature multifunction devices; however, their limited screen size means that generally only one interactive application can be visible at any time. Platforms that don't support background execution limit the viability of applications that don't need your constant attention.

Background services make it possible to create invisible application components that perform automatic processing without direct user action. Background execution allows your applications to become eventdriven and to support regular updates, which is perfect for monitoring game scores or market prices,

generating location-based alerts, or prioritizing and pre-screening incoming calls and SMS messages.

Learn more about how to get the most out of background services in Chapter 8.

SQLite Database for Data Storage and Retrieval

Rapid and effi cient data storage and retrieval are essential for a device whose storage capacity is limited by its compact nature.

Android provides a lightweight relational database for each application using SQLite. Your applications can take advantage of the managed relational database engine to store data securely and efficiently.

By default, each application database is *sandboxed* — its content is available only to the application that created it — but Content Providers supply a mechanism for the managed sharing of these application databases.

Databases, Content Providers, and other data persistence options available in Android are covered in detail in Chapter 6.

Shared Data and Interapplication Communication

Android includes three techniques for transmitting information from your applications for use elsewhere: Notifi cations, Intents, and Content Providers.

Notifi cations are the standard ways in which a mobile device traditionally alerts users. Using the API, you can trigger audible alerts, cause vibration, and flash the device's LED, as well as control status bar notifi cation icons as shown in Chapter 8.

Intents provide a mechanism for message passing within and between applications. Using Intents, you can broadcast a desired action (such as dialing the phone or editing a contact) system-wide for other applications to handle. Intents are an important core component of Android and are covered in depth in Chapter 5.

Finally, *Content Providers* are a way to give managed access to your application's private database. The data stores for native applications, such as the Contact Manager, are exposed as Content Providers so you can create your own applications that read or modify these data stores. Chapter 6 covers Content Providers in detail, including the native providers and demonstrating how to create and use providers of your own.

P2P Services with Google Talk

Based on earlier SDK versions, it's expected that in later releases you will once again be able to send structured messages from your application to any other Android mobile using Android's peer-to-peer (P2P) communications service.

The Android P2P service uses a specialized version of XMPP (Extensible Messaging and Presence Protocol). Based on Google's Google Talk instant messaging service, it creates a persistent socket connection between your device and any other online Android handset that ensures communication with low latency and rapid response times.

When made available, you'll be able to use the Google Talk service for conventional instant messaging, or an interface to send data between application instances on separate devices. This is strong sauce for creating interactive applications that involve multiple users, such as real-time multiplayer games or social applications.

The P2P service also offers presence notifi cation, which is used to see if a contact is online. While the P2P service is very attractive in itself, it also plays very well with other Android features. Imagine a background service that transmits locations between friends and a corresponding mapping application that displays these locations or alerts you when friends are nearby.

Owing to security concerns, sending data messages with Google Talk isn't possible in Android 1.0. An instant messaging client is available, and it's expected that XMPP-compatible IM and data messaging will be made available to developers in a future SDK release.

Extensive Media Support and 2D/3D Graphics

Bigger screens and brighter, higher-resolution displays have helped make mobiles multimedia devices. To make the most of the hardware available, Android provides graphics libraries for 2D canvas drawing and 3D graphics with OpenGL.

Android also offers comprehensive libraries for handling still images, video, and audio fi les including the MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, and GIF formats.

2D and 3D graphics are covered in depth in Chapter 11, while Android media management libraries are covered in Chapter 10.

Optimized Memory and Process Management

Android's process and memory management is a little unusual. Like Java and .NET, Android uses its own run time and virtual machine to manage application memory. Unlike either of these frameworks, the Android run time also manages the process lifetimes. Android ensures application responsiveness by stopping and killing processes as necessary to free resources for higher-priority applications.

In this context, priority is determined depending on the application with which the user is interacting. Ensuring that your applications are prepared for a swift death but are still able to remain responsive and update or restart in the background if necessary, is an important consideration in an environment that does not allow applications to control their own lifetimes.

You will learn more about the Android application life cycle in Chapter 3.

Introducing the Open Handset Alliance

The *Open Handset Alliance (OHA)* is a collection of more than 30 technology companies including hardware manufacturers, mobile carriers, and software developers. Of particular note are the prominent mobile technology companies Motorola, HTC, T-Mobile, and Qualcomm. In their own words, the OHA represents:

A commitment to openness, a shared vision for the future, and concrete plans to make the vision a reality. To accelerate innovation in mobile and offer consumers a richer, less expensive, and better mobile experience.

http://www.openhandsetalliance.com/oha_faq.html

The OHA hopes to deliver a better mobile software experience for consumers by providing the platform needed for innovative mobile development at a faster rate and a higher quality without licensing fees for software developers or handset manufacturers.

Ultimately the success of Android as a mobile platform will depend largely on the success of OHA partners in releasing desirable handsets and mobile services that encourage the widespread adoption of Android phones. Developers meanwhile have the opportunity to create innovative new mobile applications for Android to encourage more mobile technology companies to become part of the OHA.

What Does Android Run On?

The fi rst Android mobile handset, the T-Mobile G1, was released in the US in October 2008 and in the UK in November 2008. The Open Handset Alliance has further committed to deploying additional handsets and services that support Android early in 2009.

Rather than a mobile OS created for a single hardware implementation, Android is designed to support a large variety of hardware platforms, from touch-screen phones to devices with no screens at all.

Beyond that, with no licensing fees or proprietary software, the cost to handset manufacturers for providing Android-compatible variations of their handsets is comparatively low. It's hoped that once demand for hardware capable of running popular Android applications reaches a critical mass, more device manufacturers will produce increasingly tailored hardware to meet that demand.