Apress Beginning Android Tablet Programming, Starting with Android Honeycomb for Tablets (2011)

“

Welcome to the world of programming the Android tablet. In this chapter, I’ll introduce you to the basics of setting up your programming environment. I’ll walk you through your first Android program, and we’ll even take a run through the world’s quickest introduction to Java.

But first: a slightly self-indulgent history of the world of handheld computing.

A Short, Personal History of Portable Programming

Handheld computers have been around longer than you might think. For a lot of people, it was only yesterday that the iPhone appeared on the scene, and all the cool kids had to have one.

But I’ve been using such handheld devices on and off since the late 1970s.

My first introduction to something you could call a handheld computer was the Sharp PC-1210. This had a single-line display and a QWERTY keyboard, and could be programmed in BASIC. It was owned by a cousin of mine, and I wanted one so badly I could taste it.

Okay, not sure what I would have used it for—probably generating random encounters in D&D1— but it kicked off a mild obsession with cute handheld devices that persists to this current day.

I looked on with interest as the Apple Newton appeared in 1987, proudly announcing handwriting recognition. I also watched as it crashed and burned, a victim of its own marketing hype, which vastly surpassed its technical capability. However, it was probably my first encounter with a tablet computer.

Then the PalmPilot appeared in 1996, or thereabouts. I got me one of those. It’s still kicking around the house somewhere. The Palm had handwriting recognition of a sort, but much less ambitiously implemented, and as a result, it worked quite nicely.

I loved that thing. The designers had paid a lot of attention to the Newton debacle, and kept a firm eye on what was achievable with the technology of the day. A handheld computer is, of necessity, quite a limited device compared to a laptop computer. Never mind that today’s smartphones have more computing capacity than NASA had when they put a man on the moon; they’re still much more restricted in terms of memory, processing power, and screen real estate than a desktop computer, and, by their very nature, probably always will be.

Therefore, Palm OS was not designed to do everything. It made the simplifying assumption that anyone with a PalmPilot would also have access to a computer, so they let all the stuff requiring heavy processing and storage capacity live on that computer. They then concentrated on maximizing what the user actually saw and used. They also invented the term that is used for this whole class of devices, a personal data assistant, or PDA.

And this is an important thing to remember about programming for a mobile device. Keep it simple, avoid bloat, and concentrate on usability. Pay attention, because this will appear on the snap quiz later.

Then there were devices like the Psion. I’ve owned several models of these. They were particularly nice because they had a small but fully functional keyboard that you could actually touch-type on. The Psion had a simple programming language of its own, and ran the Symbian operating system. This is interesting because, although the Psion is long dead, the Symbian OS is alive and well and living in Nokia phones.

Then I got another Palm device, this time a LifeDrive. This was a PDA only a little thicker than a modern iPhone. It supported Wi-Fi and Bluetooth, and had a touch screen, a fairly comprehensive suite of user applications, and a 4 GB drive. In fact, apart from not actually having a phone built in, it was very close to the modern smartphone: about five years prior to the appearance of the modern smartphone.

The other side of the mobile device coin was, of course, the mobile phone.

In the late 1980s, my company put a car phone in my car rather than risk me being out of touch for more than five minutes at a time.

In the early 1990s, I got my first half-brick, and I’ve had one ever since, getting a new model every couple of years. And over that time, they’ve gotten cleverer, with bigger screens and more functions. Address books, calendars, calculators, games . . .

You see where I’m going here. The mobile phone and the PDA have been on a collision course since forever. And, for all intents and purposes, with the advent of the smartphone, they’ve arrived.

There have, of course, been other small, mobile devices in common use since at least the middle of last century—if you count cameras, even longer. (If you count watches, a couple of centuries.)

Transistor radios. Calculators. Cameras. Watches. Walkmen. MP3 players. DVD players. GPS navigators. eBook readers. They’ve all, one by one, been assimilated by either the PDA or the mobile phone, and now the PDA and the phone have assimilated each other.

Again, this trend is not as new as many people think. Combination mobile/PDAs were available a decade ago, in the form of the Kyocera 6035 (a Palm OS–based phone). There have been the Treo, the Blackberry, and various Windows CE platforms, to name but a few.

But for the most part, these were purchased by either your smart young executive or your die-hard tech geek. They were, when all is said and done, rather expensive and generally less functional than just buying the devices you needed.

With the release of Apple’s iPhone, everything seemed to come together. Functionality, price point, and marketing push all combined to make the smartphone a must-have, mass-market appliance.

There is yet another branch of the mobile computing tree that I haven’t mentioned—the laptop. The first hands-on experience I ever had with a laptop was the KayPro II (way, *way* back in 1982). This was a CP/M Z80–based machine with two floppy disks, and a built-in screen (a cathode ray tube). It wasn’t a called a laptop. It was a “luggable” computer, in that it had a handle, and you could carry it, if you were fairly strong.

The next one I had access to was a Commodore-SX64, which was a Commodore 64, built into a box, with a floppy disk, a 5-inch screen, and a handle. I thought it was pretty neat.

Real laptops appeared after that, and have steadily become more powerful, smaller, and with better batteries. Touch screen models appeared about five or six years ago, but while nice, they didn’t set the world on fire.

A few years ago, I got my first netbook—an extra small laptop with a solid state drive. I used this as my mobile device until recently, when I acquired my HTC Desire Android phone.

And then Apple announced the iPad, basically a big iPhone with a bigger screen, which neatly bridged the gap between the laptop and the smartphone.

This brings us down to two complementary mobile devices, the smartphone—something that fits into your pocket—and the tablet, something still light and portable, which has the screen size and peripherals to do serious work.

The Advent of Android

Nice as the iPhone and the iPad were technically, they were emphatically *not* open source. In fact, many people had issues with Apple’s restrictions on what could and could not be put in your smartphone.

Plus, as I’ve just laid out, the smartphone has been arriving for quite some time, with at least three well-established mobile operating systems in the marketplace, so the iPhone was soon to have competition.

Android Inc. was founded in 2003, and acquired by Google in 2005. The Open Handset Alliance—a consortium of companies including Google, Motorola, HTC, and a bunch of other handset manufacturers and carriers—was announced in November 2007 (a mere two months after the launch of the iPhone). That same day, the Android platform was revealed.

Android was built on a Linux kernel, and is fundamentally open source (with some exceptions). Android apps are primarily written in Java. The Android version of Java runs on what is called the Dalvik virtual machine, which is optimized specifically for handheld devices.

Fast-forward a few years to 2011, which I like to call “the present.” There have been a few Android tablets released, but these are still fundamentally upsized mobile phones as far as their operating system is concerned. It’s only with the release of Android 3.0 (Honeycomb) that we have a version that takes full advantage of the expanded capabilities of a full-sized tablet.

Which brings us to the whole point of this book.

”

Addison-Wesley Android Wireless Application Development 2nd (2011)

“

**A Brief History of Mobile Software Development**

To understand what makes Android so compelling, we must examine how mobile devel- opment has evolved and how Android differs from competing platforms.

**Way Back When**

Remember way back when a phone was just a phone? When we relied on fixed land- lines? When we ran for the phone instead of pulling it out of our pocket? When we lost our friends at a crowded ballgame and waited around for hours hoping to reunite? When we forgot the grocery list and had to find a payphone or drive back home again?

Those days are long gone.Today, commonplace problems such as these are easily solved with a one-button speed dial or a simple text message like “WRU?” or “20?” or “Milk and?”

Our mobile phones keep us safe and connected. Now we roam around freely, relying on our phones not only to keep in touch with friends, family, and coworkers, but also to tell us where to go, what to do, and how to do it. Even the most domestic of events seem to revolve around my mobile phone.

Consider the following true story, which has been slightly enhanced for effect:

Once upon a time, on a warm summer evening, I was happily minding my own business cooking dinner in my new house in rural New Hampshire when a bat swooped over my head, scaring me to death.

The first thing I did—while ducking—was to pull out my cell phone and send a text mes- sage to my husband, who was across the country at the time. I typed, “There’s a bat in the house!”

Mobile phones can solve just about *anything*—and we rely on them for *everything* these days.

***Use in the introduction***

**You notice that I used half a dozen different mobile applications over the course of this story. Each application was developed by a different company and had a different user interface. Some were well designed; others not so much. I paid for some of the applica- tions, and others came on my phone.**

**As a user, I found the experience functional, but not terribly inspiring. As a mobile de- veloper, I wished for an opportunity to create a more seamless and powerful application that could handle all I’d done and more. I wanted to build a better bat trap, if you will.**

Before Android, mobile developers faced many roadblocks when it came to writing applications. Building the better application, the unique application, the competing appli- cation, the hybrid application, and incorporating many common tasks such as messaging and calling in a familiar way were often unrealistic goals.

To understand why, let’s take a brief look at the history of mobile software development.

**“The Brick”**

The Motorola DynaTAC 8000X was the first commercially available cell phone. First marketed in 1983, it was 13 × 1.75 × 3.5 inches in dimension, weighed about 2.5 pounds, and allowed you to talk for a little more than half an hour. It retailed for $3,995, plus hefty monthly service fees and per-minute charges.

We called it “The Brick,” and the nickname stuck for many of those early mobile phones we alternatively loved and hated.About the size of a brick,with a battery power just long enough for half a conversation, these early mobile handsets were mostly seen in the hands of traveling business execs, security personnel, and the wealthy. First-generation mobile phones were just too expensive.The service charges alone would bankrupt the av- erage person, especially when roaming.

Early mobile phones were not particularly full featured. (Although, even the Motorola DynaTAC, shown in Figure 1.2, had many of the buttons we’ve come to know well, such as the SEND, END, and CLR buttons.) These early phones did little more than make and receive calls and, if you were lucky, there was a simple contacts application that wasn’t im- possible to use.

The first-generation mobile phones were designed and developed by the handset manufacturers. Competition was fierce and trade secrets were closely guarded. Manufac- turers didn’t want to expose the internal workings of their handsets, so they usually devel- oped the phone software in-house. As a developer, if you weren’t part of this inner circle, you had no opportunity to write applications for the phones.

It was during this period that we saw the first “time-waster” games begin to appear. Nokia was famous for putting the 1970s video game Snake on some of its earliest mono- chrome phones. Other manufacturers followed suit, adding games such as Pong,Tetris, and Tic-Tac-Toe.

These early phones were flawed, but they did something important—they changed the way people thought about communication.As mobile phone prices dropped, batteries improved, and reception areas grew, more and more people began carrying these handy devices. Soon mobile phones were more than just a novelty.

Customers began pushing for more features and more games. But there was a problem. The handset manufacturers didn’t have the motivation or the resources to build every ap- plication users wanted.They needed some way to provide a portal for entertainment and information services without allowing direct access to the handset.

What better way to provide these services than the Internet?

**Wireless Application Protocol (WAP)**

As it turned out, allowing direct phone access to the Internet didn’t scale well for mobile. By this time, professional websites were full color and chock full of text, images, and

other sorts of media.These sites relied on JavaScript, Flash, and other technologies to en- hance the user experience, and they were often designed with a target resolution of 800x600 pixels and higher.

When the first clamshell phone, the Motorola StarTAC, was released in 1996, it merely had an LCD 10-digit segmented display. (Later models would add a dot-matrix type dis- play.) Meanwhile, Nokia released one of the first slider phones, the 8110—fondly referred to as “The Matrix Phone” because the phone was heavily used in films.The 8110 could display four lines of text with 13 characters per line. Figure 1.3 shows some of the com- mon phone form factors.

          

Figure 1.3 Various mobile phone form factors: the candy bar, the slider, and the clamshell.

With their postage stamp-sized low-resolution screens and limited storage and process- ing power, these phones couldn’t handle the data-intensive operations required by tradi- tional web browsers.The bandwidth requirements for data transmission were also costly to the user.

The Wireless Application Protocol (WAP) standard emerged to address these concerns. Simply put,WAP was a stripped-down version of HTTP, which is the backbone protocol of the Internet. Unlike traditional web browsers,WAP browsers were designed to run within the memory and bandwidth constraints of the phone.Third-party WAP sitesserved up pages written in a markup language called Wireless Markup Language (WML). These pages were then displayed on the phone’s WAP browser. Users navigated as they would on the Web, but the pages were much simpler in design.

The WAP solution was great for handset manufacturers.The pressure was off—they could write one WAP browser to ship with the handset and rely on developers to come up with the content users wanted.

The WAP solution was great for mobile operators.They could provide a custom WAP portal, directing their subscribers to the content they wanted to provide, and rake in the data charges associated with browsing, which were often high.

Developers and content providers didn’t deliver. For the first time, developers had a chance to develop content for phone users, and some did so, with limited success.

Most of the early WAP sites were extensions of popular branded websites, such as CNN.com and ESPN.com, which were looking for new ways to extend their reader- ship. Suddenly phone users accessed the news, stock market quotes, and sports scores on their phones.

Commercializing WAP applications was difficult, and there was no built-in billing mechanism. Some of the most popular commercial WAP applications that emerged dur- ing this time were simple wallpaper and ringtone catalogues that enabled users to person- alize their phones for the first time. For example, a user browsed a WAP site and requested a specific item. He filled out a simple order form with his phone number and his handset model. It was up to the content provider to deliver an image or audio file compatible with the given phone. Payment and verification were handled through various premium- priced delivery mechanisms such as Short Message Service (SMS), Enhanced Messaging Service (EMS), Multimedia Messaging Service (MMS), and WAP Push.

WAP browsers, especially in the early days, were slow and frustrating.Typing long URLs with the numeric keypad was onerous.WAP pages were often difficult to navi- gate. Most WAP sites were written one time for all phones and did not account for indi- vidual phone specifications. It didn’t matter if the end user’s phone had a big color screen or a postage stamp-sized monochrome screen; the developer couldn’t tailor the user’s ex- perience.The result was a mediocre and not very compelling experience for everyone involved.

Content providers often didn’t bother with a WAP site and instead just advertised SMS short codes on TV and in magazines. In this case, the user sent a premium SMS message with a request for a specific wallpaper or ringtone, and the content provider sent it back. Mobile operators generally liked these delivery mechanisms because they received a large portion of each messaging fee.

WAP fell short of commercial expectations. In some markets, such as Japan, it flour- ished, whereas in others, such as the United States, it failed to take off. Handset screens were too small for surfing. Reading a sentence fragment at a time, and then waiting sec- onds for the next segment to download, ruined the user experience, especially because every second of downloading was often charged to the user. Critics began to call WAP “Wait and Pay.”

Finally, the mobile operators who provided the WAP portal (the default home page loaded when you started your WAP browser) often restricted which WAP sites were ac- cessible.The portal enabled the operator to restrict the number of sites users could browse and to funnel subscribers to the operator’s preferred content providers and exclude com- peting sites.This kind of walled garden approach further discouraged third-party develop- ers, who already faced difficulties in monetizing applications, from writing applications.

**Proprietary Mobile Platforms**

It came as no surprise that users wanted more—they will always want more. Writing robust applications with WAP, such as graphic-intensive video games, was nearly impossible.The 18-year-old to 25-year-old sweet-spot demographic—the kids

with the disposable income most likely to personalize their phones with wallpapers and ringtones—looked at their portable gaming systems and asked for a device that was both a phone and a gaming device or a phone and a music player.They argued that if devices such as Nintendo’s Game Boy could provide hours of entertainment with only five but- tons, why not just add phone capabilities? Others looked to their digital cameras, Palms, BlackBerries, iPods, and even their laptops and asked the same question.The market seemed to be teetering on the edge of device convergence.

Memory was getting cheaper, batteries were getting better, and PDAs and other em- bedded devices were beginning to run compact versions of common operating systems such as Linux and Windows.The traditional desktop application developer was suddenly a player in the embedded device market, especially with smartphone technologies such as Windows Mobile, which they found familiar.

Handset manufacturers realized that if they wanted to continue to sell traditional handsets, they needed to change their protectionist policies pertaining to handset design and expose their internal frameworks to some extent.

A variety of different proprietary platforms emerged—and developers are still actively creating applications for them. Some smartphone devices ran Palm OS (now Garnet OS) and RIM BlackBerry OS. Sun Microsystems took its popular Java platform and J2ME emerged (now known as Java Micro Edition [Java ME]). Chipset maker Qualcomm de- veloped and licensed its Binary Runtime Environment for Wireless (BREW). Other plat- forms, such as Symbian OS, were developed by handset manufacturers such as Nokia, Sony Ericsson, Motorola, and Samsung.The Apple iPhone OS (OS X iPhone) joined the ranks in 2008.

Figure 1.4 shows several different phones, all of which have different devel- opment platforms.

Many of these platforms have associated developer programs.These programs keep the developer communities small, vetted, and under contractual agreements on what they can and cannot do.These programs are often required and developers must pay for them.

Each platform has benefits and drawbacks. Of course, developers love to debate about which platform is “the best.” (Hint: It’s usually the platform we’re currently developing for.) The truth is that no one platform has emerged victorious. Some platforms are best

suited for commercializing games and making millions—if your company has brand

backing. Other platforms are more open and suitable for the hobbyist or vertical market applications. No mobile platform is best suited for all possible applications. As a result, the mobile phone has become increasingly fragmented, with all platforms sharing part of the pie.

Figure 1.4 Phones from various mobile device platforms.

For manufacturers and mobile operators, handset product lines quickly became complicated. Platform market penetration varies greatly by region and user demographic. In- stead of choosing just one platform, manufacturers and operators have been forced to sell phones for all the different platforms to compete in the market.We’ve even seen some handsets supporting multiple platforms. (For instance, Symbian phones often also support J2ME.)

The mobile developer community has become as fragmented as the market. It’s nearly impossible to keep track of all the changes in the market. Developer specialty niches have formed.The platform development requirements vary greatly. Mobile software developers work with distinctly different programming environments, different tools, and different programming languages. Porting among the platforms is often costly and not straightfor- ward. Keeping track of handset configurations and testing requirements, signing and certi- fication programs, carrier relationships, and application marketplaces have become complex spin-off businesses of their own.

It’s a nightmare for the ACME Company that wants a mobile application. Should it develop a J2ME application? BREW? iPhone? Windows Mobile? Everyone has a differ- ent kind of phone. ACME is forced to choose one or, worse, all of the platforms. Some platforms allow for free applications, whereas others do not.Vertical market application opportunities are limited and expensive.

As a result, many wonderful applications have not reached their desired users, and many other great ideas have not been developed at all.

**The Open Handset Alliance**

Enter search advertising giant Google. Now a household name, Google has shown an in- terest in spreading its vision, its brand, its search and ad-revenue-based platform, and its suite of tools to the wireless marketplace.The company’s business model has been amaz- ingly successful on the Internet and, technically speaking, wireless isn’t that different.

**Google Goes Wireless**

The company’s initial forays into mobile were beset with all the problems you would ex- pect.The freedoms Internet users enjoyed were not shared by mobile phone subscribers. Internet users can choose from the wide variety of computer brands, operating systems, Internet service providers, and web browser applications.

Nearly all Google services are free and ad driven. Many applications in the Google Labs suite directly compete with the applications available on mobile phones.The appli- cations range from simple calendars and calculators to navigation with Google Maps and the latest tailored news from News Alerts—not to mention corporate acquisitions such as Blogger andYouTube.

When this approach didn’t yield the intended results, Google decided to a different ap- proach—to revamp the entire system upon which wireless application development was based, hoping to provide a more open environment for users and developers: the Internet model.The Internet model allows users to choose between freeware, shareware, and paid software.This enables free market competition among services.

**Forming the Open Handset Alliance**

With its user-centric, democratic design philosophies, Google has led a movement to turn the existing closely guarded wireless market into one where phone users can move be- tween carriers easily and have unfettered access to applications and services.With its vast resources, Google has taken a broad approach, examining the wireless infrastructure from the FCC wireless spectrum policies to the handset manufacturers’ requirements, applica- tion developer needs, and mobile operator desires.

Next, Google joined with other like-minded members in the wireless community and posed the following question:What would it take to build a better mobile phone?

The Open Handset Alliance (OHA) was formed in November 2007 to answer that very question.The OHA is a business alliance comprised of many of the largest and most

successful mobile companies on the planet. Its members include chip makers, handset manufacturers, software developers, and service providers.The entire mobile supply chain is well represented.

Andy Rubin has been credited as the father of the Android platform. His company, Android Inc., was acquired by Google in 2005.Working together, OHA members, includ- ing Google, began developing a nonproprietary open standard platform based upon tech- nology developed at Android Inc. that would aim to alleviate the aforementioned problems hindering the mobile community.The result is the Android project.To this day, most Android platform development is completed by Rubin’s team at Google, where he acts as VP of Engineering and manages the Android platform roadmap.

Google’s involvement in the Android project has been so extensive that the line be- tween who takes responsibility for the Android platform (the OHA or Google) has blurred. Google hosts the Android open source project and provides online Android doc- umentation, tools, forums, and the Software Development Kit (SDK) for developers. All major Android news originates at Google.The company has also hosted a number of events at conferences and the Android Developer Challenge (ADC), a contest to encour- age developers to write killer Android applications—for $10 million dollars in prizes to spur development on the platform.The winners and their apps are listed on the Android website.

**Manufacturers: Designing the Android Handsets**

More than half the members of the OHA are handset manufacturers, such as Samsung, Motorola, HTC, and LG, and semiconductor companies, such as Intel,Texas Instruments, NVIDIA, and Qualcomm.These companies are helping design the first generation of An- droid handsets.

The first shipping Android handset—the T-Mobile G1—was developed by handset manufacturer HTC with service provided by T-Mobile. It was released in October 2008. Many other Android handsets were slated for 2009 and early 2010.The platform gained momentum relatively quickly. Each new Android device was more powerful and exciting than the last. Over the following 18 months, 60 different Android handsets (made by 21 different manufacturers) debuted across 59 carriers in 48 countries around the world. By June 2010, at an announcement of a new, highly anticipated Android handset, Google an- nounced more than 160,000 Android devices were being activated each day (for a rate of nearly 60 million devices annually).The advantages of widespread manufacturer and car- rier support appear to be really paying off at this point.

The Android platform is now considered a success. It has shaken the mobile market- place, gaining ground steadily against competitive platforms such as the Apple iPhone, RIM BlackBerry, and Windows Mobile.The latest numbers (as of Summer 2010) show BlackBerry in the lead with a declining 31% of the smartphone market.Trailing close be- hind is Apple’s iPhone at 28%. Android, however, is trailing with 19%, though it’s gaining ground rapidly and, according to some sources, is the fastest-selling smartphone platform. Microsoft Windows Mobile has been declining and now trails Android by several percent- age points.

**Mobile Operators: Delivering the Android Experience**

After you have the phones, you have to get them out to the users. Mobile operators from North, South, and Central America; Europe, Asia, India, Australia, Africa, and the Middle East have joined the OHA, ensuring a worldwide market for the Android movement. With almost half a billion subscribers alone, telephony giant China Mobile is a founding member of the alliance.

Much of Android’s success is also due to the fact that many Android handsets don’t come with the traditional “smartphone price tag”—quite a few are offered free with acti- vation by carriers. Competitors such as the Apple iPhone have no such offering as of yet. For the first time, the average Jane or Joe can afford a feature-full phone. I’ve lost count of the number of times I’ve had a waitress, hotel night manager, or grocery store checkout person tell me that they just got an Android phone and it has changed their life.This phe- nomenon has only added to the Android’s rising underdog status.

In the United States, the Android platform was given a healthy dose of help from car- riers such asVerizon,who launched a $100 million dollar campaign for the first Droid handset. Many other Droid-style phones have followed from other carriers. Sprint re- cently launched the Evo 4G (America’s first 4G phone) to much fanfare and record one- day sales (http://j.mp/cNhb4b).

**Content Providers: Developing Android Applications**

When users have Android handsets, they need those killer apps, right? Google has led the pack, developing Android applications, many of which, such as the

email client and web browser, are core features of the platform. OHA members are also working on Android application integration. eBay, for example, is working on integration with its online auctions.

The first ADC received 1,788 submissions, with the second ADC being voted upon by 26,000 Android users to pick a final 200 applications that would be judged profession- ally—all newly developed Android games, productivity helpers, and a slew of location- based services (LBS) applications.We also saw humanitarian, social networking, and mash-up apps. Many of these applications have debuted with users through the Android Market—Google’s software distribution mechanism for Android. For now, these chal- lenges are over.The results, though, are still impressive.

For those working on the Android platform from the beginning, handsets couldn’t come fast enough.The T-Mobile G1 was the first commercial Android device on the market, but it had the air of a developer pre-release handset. Subsequent Android handsets have had much more impressive hardware, allowing developers to dive in and design awe- some new applications.

As of October 2010, there are more than 80,000 applications available in the Android Market, which is growing rapidly.This takes into account only applications published through this one marketplace—not the many other applications sold individually or on other markets.This also does not take into account that, as of Android 2.2, Flash applica- tions can run on Android handsets.This opens up even more application choices for An- droid users and more opportunities for Android developers.

There are now more than 180,000 Android developers writing interesting and exciting applications. By the time you finish reading this book, you will be adding your expertise to this number.

**Taking Advantage of All Android Has to Offer**

Android’s open platform has been embraced by much of the mobile development com- munity—extending far beyond the members of the OHA.

As Android phones and applications have become more readily available, many other mobile operators and handset manufacturers have jumped at the chance to sell Android phones to their subscribers, especially given the cost benefits compared to proprietary platforms.The open standard of the Android platform has resulted in reduced operator costs in licensing and royalties, and we are now seeing a migration to open handsets from proprietary platforms such as RIM,Windows Mobile, and the Apple iPhone.The market has cracked wide open; new types of users are able to consider smartphones for the first time.Android is well suited to fill this demand.

**Android: A Next-Generation Platform**

Although Android has many innovative features not available in existing mobile plat- forms, its designers also leveraged many tried-and-true approaches proven to work in the wireless world. It’s true that many of these features appear in existing proprietary

platforms, but Android combines them in a free and open fashion while simultaneously addressing many of the flaws on these competing platforms.

The Android mascot is a little green robot, shown in Figure 1.5.This little guy (girl?) is often used to depict Android-related materials.

Android is the first in a new generation of mobile platforms, giving its platform devel- opers a distinct edge on the competition.Android’s designers examined the benefits and drawbacks of existing platforms and then incorporated their most successful features.At the same time,Android’s designers avoided the mistakes others suffered in the past.

Since the Android 1.0 SDK was released,Android platform development has continued at a fast and furious pace. For quite some time, there was a new Android SDK out every couple of months! In typical tech-sector jargon, each Android SDK has had a project name. In Android’s case, the SDKs are named alphabetically after sweets (see Figure 1.6).

The latest version of Android is codenamed Gingerbread.

Figure 1.5 The Android mascot and logo.

Figure 1.6 Some Android SDKs and their codenames.

Android Platform Differences 19

   

20 Chapter 1 Introducing Android

**Free and Open Source**

Android is an open source platform. Neither developers nor handset manufacturers pay royalties or license fees to develop for the platform.

The underlying operating system of Android is licensed under GNU General Public License Version 2 (GPLv2), a strong “copyleft” license where any third-party improve- ments must continue to fall under the open source licensing agreement terms.The An- droid framework is distributed under the Apache Software License (ASL/Apache2), which allows for the distribution of both open- and closed-source derivations of the source code. Commercial developers (handset manufacturers especially) can choose to en- hance the platform without having to provide their improvements to the open source community. Instead, developers can profit from enhancements such as handset-specific improvements and redistribute their work under whatever licensing they want.

Android application developers have the ability to distribute their applications under whatever licensing scheme they prefer. Developers can write open source freeware or tra- ditional licensed applications for profit and everything in between.

**Familiar and Inexpensive Development Tools**

Unlike some proprietary platforms that require developer registration fees, vetting, and expensive compilers, there are no upfront costs to developing Android applications.

**Freely Available Software Development Kit**

The Android SDK and tools are freely available. Developers can download the Android SDK from the Android website after agreeing to the terms of the Android Software De- velopment Kit License Agreement.

**Familiar Language, Familiar Development Environments**

Developers have several choices when it comes to integrated development environments (IDEs). Many developers choose the popular and freely available Eclipse IDE to design and develop Android applications. Eclipse is the most popular IDE for Android develop- ment, and there is an Android plug-in available for facilitating Android development. An- droid applications can be developed on the following operating systems:

n Windows XP (32-bit) orVista (32-bit or 64-bit) n Mac OS X 10.5.8 or later (x86 only) n Linux (tested on Linux Ubuntu 8.04 LTS, Hardy Heron)

**Reasonable Learning Curve for Developers**

Android applications are written in a well-respected programming language: Java. The Android application framework includes traditional programming constructs, such as threads and processes and specially designed data structures to encapsulate objects com-

monly used in mobile applications. Developers can rely on familiar class libraries, such as java.net and java.text. Specialty libraries for tasks such as graphics and database

management are implemented using well-defined open standards such as OpenGL Em- bedded Systems (OpenGL ES) or SQLite.

**Enabling Development of Powerful Applications**

In the past, handset manufacturers often established special relationships with trusted third-party software developers (OEM/ODM relationships).This elite group of software developers wrote native applications, such as messaging and web browsers, which shipped on the handset as part of the phone’s core feature set.To design these applications, the manufacturer would grant the developer privileged inside access and knowledge of a handset’s internal software framework and firmware.

On the Android platform, there is no distinction between native and third-party appli- cations, enabling healthy competition among application developers. All Android applica- tions use the same libraries.Android applications have unprecedented access to the underlying hardware, allowing developers to write much more powerful applications. Ap- plications can be extended or replaced altogether. For example, Android developers are now free to design email clients tailored to specific email servers, such as Microsoft Ex- change or Lotus Notes.

**Rich, Secure Application Integration**

Recall from the bat story I previously shared that I accessed a variety of phone applica- tions in the course of a few moments: text messaging, phone dialer, camera, email, picture messaging, and the browser. Each was a separate application running on the phone— some built-in and some purchased. Each had its own unique user interface. None were truly integrated.

Not so with Android. One of the Android platform’s most compelling and innovative features is well-designed application integration.Android provides all the tools necessary to build a better “bat trap,” if you will, by allowing developers to write applications that seamlessly leverage core functionality such as web browsing, mapping, contact manage- ment, and messaging. Applications can also become content providers and share their data among each other in a secure fashion.

Platforms such as Symbian have suffered from setbacks due to malware.Android’s vig- orous application security model helps protect the user and the system from malicious software.

**No Costly Obstacles to Publication**

Android applications have none of the costly and time-intensive testing and certification programs required by other platforms such as BREW and Symbian.

Android Platform Differences 21

22 Chapter 1 Introducing Android

**A “Free Market” for Applications**

Android developers are free to choose any kind of revenue model they want.They can develop freeware, shareware, or trial-ware applications, ad-driven, and paid applications. Android was designed to fundamentally change the rules about what kind of wireless ap- plications could be developed. In the past, developers faced many restrictions that had lit- tle to do with the application functionality or features:

n Store limitations on the number of competing applications of a given type n Store limitations on pricing, revenue models, and royalties n Operator unwillingness to provide applications for smaller demographics

With Android, developers can write and successfully publish any kind of application they want. Developers can tailor applications to small demographics, instead of just large-scale money-making ones often insisted upon by mobile operators.Vertical market applications can be deployed to specific, targeted users.

Because developers have a variety of application distribution mechanisms to choose from, they can pick the methods that work for them instead of being forced to play by oth- ers’ rules.Android developers can distribute their applications to users in a variety of ways:

n Google developed the Android Market (see Figure 1.7), a generic Android applica- tion store with a revenue-sharing model.

Figure 1.7 The Android market.

n Handango.com added Android applications to its existing catalogue using their billing models and revenue-sharing model.

n Developers can come up with their own delivery and payment mechanisms.

Mobile operators are still free to develop their own application stores and enforce their own rules, but it will no longer be the only opportunity developers have to distribute their applications.

**A New and Growing Platform**

Android might be the next generation in mobile platforms, but the technology is still in its early stages. Early Android developers have had to deal with the typical roadblocks as- sociated with a new platform: frequently revised SDKs, lack of good documentation, and market uncertainties.

On the other hand, developers diving into Android development now benefit from the first-to-market competitive advantages we’ve seen on other platforms such as BREW



and Symbian. Early developers who give feedback are more likely to have an impact on the long-term design of the Android platform and what features will come in the next version of the SDK. Finally, the Android forum community is lively and friendly. Incen- tive programs, such as the ADC, have encouraged many new developers to dig into the platform.

Each new version of the Android SDK has provided a number of substantial improve- ments to the platform. In recent revisions, the Android platform has received some much- needed UI “polish,” both in terms of visual appeal and performance.Although most of these upgrades and improvements were welcome and necessary, new SDK versions often cause some upheaval within the Android developer community. A number of published applications have required retesting and resubmission to the Android Marketplace to con- form to new SDK requirements, which are quickly rolled out to all Android phones in the field as a firmware upgrade, rendering older applications obsolete.

Some older Android handsets are not capable of running the latest versions of the plat- form.This means that Android developers often need to target several different SDK ver- sions to reach all users. Luckily, the Android development tools make this easier than ever.

“

Android UI Fundamentals - Develop & Design

“

use for chapter 3

“