1

Introduction to Android

Objectives

In this chapter you'll be introduced to:

- The history of Android and the Android SDK.
- The Android Market for apps.
- A review of basic object-technology concepts.
- Key software for Android app development, including the Android SDK, the Java SDK and Eclipse integrated development environment.
- Important Android documentation.
- Test-driving an Android app that enables you to draw on the screen.
- The Deitel online Android Resource Centers.

Outline

1.1 Introduction
1.2 Android Overview
1.3 Android 2.2 (Froyo)
1.4 Android 2.3 (Gingerbread)
1.5 Android 3.0 (Honeycomb)
1.6 Android Ice Cream Sandwich
1.7 Downloading Apps from the Android Market
1.8 Packages
1.9 Android Software Development Kit (SDK)
1.10 Object Technology: A Quick Refresher
1.11 Test-Driving the **Doodlz** App in an Android Virtual Device (AVD)
1.12 Deitel Resources
1.13 Android Development Resources
1.14 Wrap-Up

1.1 Introduction

Welcome to Android app development! We hope that you'll find working with *Android for Programmers: An App-Driven Approach* to be an informative, challenging, entertaining and rewarding experience. This book is geared toward Java programmers. We use only complete working apps, so if you don't know Java but have object-oriented programming experience in another language, such as C#, Objective-C/Cocoa or C++ (with class libraries), you should be able to master the material quickly, learning Java and Java-style object-oriented programming as you learn Android app development.

The book uses an ***app-driven approach***—we discuss each new technology in the context of complete working Android apps, with one app per chapter. We describe the app and test-drive it. Next, we briefly overview the key ***Eclipse*** (integrated development environment), Java and ***Android SDK (Software Development Kit)*** technologies we'll use to implement the app. For apps that require it, we walk through designing the GUI visually using Eclipse. Then we provide the complete source-code listing, using line numbers, syntax shading (to mimic the syntax coloring used in the Eclipse IDE) and code highlighting to emphasize the key portions of the code. We also show one or more screen shots of the running app. Then we do a detailed code walkthrough, emphasizing the new programming concepts introduced in the app. The source code for all of the book's apps can be downloaded from `www.deitel.com/books/AndroidFP/`. Figure 1.1 lists key online Android documentation.

Title	URL
Android Developer Guide	`developer.android.com/guide/index.html`
Using the Android Emulator	`developer.android.com/guide/developing/devices/emulator.html`
Android Package Index	`developer.android.com/reference/packages.html`
Android Class Index	`developer.android.com/reference/classes.html`
User Interface Guidelines	`developer.android.com/guide/practices/ui_guidelines/index.html`

Fig. 1.1 | Key online documentation for Android developers. (Part 1 of 2.)

Title	URL
Data Backup	`developer.android.com/guide/topics/data/backup.html`
Security and Permissions	`developer.android.com/guide/topics/security/security.html`
Managing Projects from Eclipse with ADT	`developer.android.com/guide/developing/projects/projects-eclipse.html`
Debugging Tasks	`developer.android.com/guide/developing/debug-tasks.html`
Tools Overview	`developer.android.com/guide/developing/tools/index.html`
Publishing Your Apps	`developer.android.com/guide/publishing/publishing.html`
Android Market Getting Started	`market.android.com/support/bin/topic.py?hl=en&topic=15866`
Android Market Developer Distribution Agreement	`www.android.com/us/developer-distribution-agreement.html`

Fig. 1.1 | Key online documentation for Android developers. (Part 2 of 2.)

Read the Before You Begin section following the Preface for information on downloading the software you'll need to build Android apps. The Android Developer site provides free downloads plus documentation, how-to videos (Fig. 1.38), coding guidelines and more. To publish your apps to Google's app marketplace—***Android Market***—you'll need to create a developer profile at `market.android.com/publish/signup`. There's a registration fee and you must agree to the Android Market Developer Distribution Agreement. We discuss publishing your apps in more detail in Chapter 2, Android Market and App Business Issues.

As you dive into Android app development, you may have questions about the tools, design issues, security and more. There are several Android developer newsgroups and forums where you can get the latest announcements or ask questions (Fig. 1.2).

Title	Subscribe	Description
Android Discuss	*Subscribe using Google Groups:* `android-discuss` *Subscribe via e-mail:* `android-discuss-subscribe@googlegroups.com`	A general Android discussion group where you can get answers to your app-development questions.
Stack Overflow	`stackoverflow.com/questions/tagged/android`	Use this list for beginner-level Android app-development questions, including getting started with Java and Eclipse, and questions about best practices.

Fig. 1.2 | Android newsgroups and forums. (Part 1 of 2.)

Title	Subscribe	Description
Android Developers	*Subscribe using Google Groups:* `android-developers` *Subscribe via e-mail:* `android-developers-subscribe@googlegroups.com`	Experienced Android developers use this list for troubleshooting apps, GUI design issues, performance issues and more.
Android Market Help Forum	`www.google.com/support/forum/p/Android+market`	Ask questions and find answers regarding Android Market.
Android Forums	`www.androidforums.com/`	Ask questions, share tips with other developers and find forums targeting specific Android devices.

Fig. 1.2 | Android newsgroups and forums. (Part 2 of 2.)

1.2 Android Overview

The first-generation Android phones were released in October 2008. According to Gartner, North American sales of Android-based phones increased 707% in the first quarter of 2010 over the previous year.[1] By March 2011, a Nielsen study showed that Android had 37% of the U.S. smartphone market share, compared to 27% for Apple's iPhone and 22% for Blackberry.[2] In August 2010, more than 200,000 Android smartphones were being activated each day, up from 100,000 per day only two months earlier.[3] As of May 2011, more than 400,000 Android devices were being activated daily. There are now over 300 different Android devices worldwide.

The Android operating system was developed by Android, Inc., which was acquired by Google in July 2005. In November 2007, the Open Handset Alliance™—a consortium of 34 companies initially and 81 now (Fig. 1.3)—was formed to develop Android, driving innovation in mobile technology and improving the user experience while reducing costs. Android is used in numerous smartphones, e-reader devices and tablet computers.

Open Handset Alliance Members		
Mobile Operators		
Bouygues Telecom	China Telecommunications Corporation	KDDI Corporation
China Mobile Communications Corporation	China United Network Communications	NTT Docomo, Inc.
		Softbank Mobile Corp.
		Sprint Nextel

Fig. 1.3 | Open Handset Alliance members (`www.openhandsetalliance.com/oha_members.html`). (Part 1 of 2.)

1. `www.gartner.com/it/page.jsp?id=1372013`.
2. `blog.nielsen.com/nielsenwire/online_mobile/u-s-smartphone-market-whos-the-most-wanted/`.
3. `www.wired.com/gadgetlab/2010/08/google-200000-android-phones/`.

Open Handset Alliance Members		
T-Mobile	Telefónica	Vodafone
Telecom Italia	Telus	
Semiconductor Companies		
AKM Semiconductor Inc.	Freescale Semiconductor	Qualcomm Inc.
Audience	Gemalto	Renesas Electronics Corporation
ARM	Intel Corporation	ST-Ericsson
Atheros Communications	Marvell Semiconductor, Inc.	Synaptics, Inc.
Broadcom Corporation	MediaTek, Inc.	Texas Instruments
CSR Plc.	MIPS Technologies, Inc.	Via Telecom
Cypress Semiconductor Corporation	NVIDIA Corporation	
Handset Manufacturers		
Acer Inc.	Haier Telecom (Qingdao) Co., Ltd.	Motorola, Inc.
Alcatel mobile phones	HTC Corporation	NEC Corporation
ASUSTeK Computer Inc.	Huawei Technologies	Samsung Electronics
CCI	Kyocera	Sharp Corporation
Dell	Lenovo Mobile Communication Technology Ltd.	Sony Ericsson
Foxconn International Holdings Limited	LG Electronics, Inc.	Toshiba Corporation
Garmin International, Inc.		ZTE Corporation
Software Companies		
Access Co., Ltd.	Myriad	PacketVideo (PV)
Ascender Corp.	Motoya Co., Ltd.	SkyPop
Cooliris, Inc.	Nuance Communications, Inc.	SONiVOX
eBay Inc.	NXP Software	SVOX
Google Inc.	OMRON Software Co., Ltd.	VisualOn Inc.
LivingImage Ltd.		
Commercialization Companies		
Accenture	Sasken Communication Technologies Limited	Teleca AB
Aplix Corporation	SQLStar International Inc.	Wind River
Borqs	TAT—The Astonishing Tribe AB	Wipro Technologies
L&T Infotech		
Noser Engineering Inc.		

Fig. 1.3 | Open Handset Alliance members (`www.openhandsetalliance.com/oha_members.html`). (Part 2 of 2.)

Openness and Open Source

One benefit of developing Android apps is the openness of the platform. The operating system is *open source* and free. This allows you to view Android's source code and see how its features are implemented. You can also contribute to Android by reporting bugs (see `source.android.com/source/report-bugs.html`) or by participating in the Open Source Project discussion groups (`source.android.com/community/index.html`). Numerous open-source Android apps from Google and others are available on the Internet (Fig. 1.4). Figure 1.5 shows you where you can get the Android source code, learn about the philosophy behind the open-source operating system and get licensing information.

Description	URL
Extensive list of open-source apps, organized by category (e.g., games, utilities, etc.).	`en.wikipedia.org/wiki/List_of_open_source_Android_applications`
Google's sample apps for the Android platform.	`code.google.com/p/apps-for-android/`
Thirty sample apps demonstrating several Android features.	`developer.android.com/resources/browser.html?tag=sample`
Lists 12 open-source Android apps.	`www.techdrivein.com/2010/11/12-open-source-android-applications.html`
Provides links to a selection of open-source Android games.	`www.techdrivein.com/2010/12/15-nice-and-simple-open-source-android.html`

Fig. 1.4 | Open-source Android apps resource sites.

Title	URL
Get Android Source Code	`source.android.com/source/download.html`
Philosophy and Goals	`source.android.com/about/philosophy.html`
Licenses	`source.android.com/source/licenses.html`
FAQs	`source.android.com/faqs.html#aosp`

Fig. 1.5 | Android source code and documentation resources.

Java

Android apps are developed with Java—the world's most widely used programming language. Java—the world's most widely used programming language—was a logical choice for the Android platform, because it's powerful, free and open source. Java is used to develop large-scale enterprise applications, to enhance the functionality of web servers, to provide applications for consumer devices (e.g., cell phones, pagers and personal digital assistants) and for many other purposes.

Java enables you to develop apps that will run on a variety of devices without any platform-specific code. Experienced Java programmers can quickly dive into Android development, using the Android APIs (Application Programming Interfaces) and others available from third parties.

The openness of the platform spurs rapid innovation. Android is available on devices from dozens of original equipment manufacturers (OEMs) in 48 countries through 59 carriers.[4] The intense competition among OEMs and carriers benefits customers.

4. `code.google.com/events/io/2010/`.

Java is object oriented and has access to powerful class libraries that help you develop apps quickly. GUI programming in Java is event driven—in this book, you'll write apps that respond to various user-initiated events such as screen touches and keystrokes. In addition to directly programming portions of your apps, you'll also use Eclipse to conveniently drag and drop predefined objects such as buttons and textboxes into place on your screen, and label and resize them. Using Eclipse with the Android Development Tools (ADT) Plugin, you can create, run, test and debug Android apps quickly and conveniently, and you can visually design your user interfaces.

Multitouch Screen

Many Android smartphones wrap the functionality of a mobile phone, Internet client, MP3 player, gaming console, digital camera and more into a handheld device with full-color ***multitouch screens***. These allow you to control the device with ***gestures*** involving one touch or multiple simultaneous touches (Fig. 1.6).

Gesture name	Physical action	Used to
Touch	Tap the screen once.	Open an app, "press" a button or a menu item.
Double tap	Tap the screen twice.	Zoom in and then back out on pictures, Google Maps and web pages.
Long press	Touch the screen and hold finger in position.	Open a context menu or grab app icons or objects to move by dragging.
Drag	Touch and drag your finger across the screen.	Move objects or icons, or scroll precisely on a web page or list.
Fling	Touch and quickly flick your finger across the screen in the direction you'd like to move.	Scroll through a **List View** (e.g., Contacts) or a **DatePicker View** and **TimePicker View** (e.g., dates and times in the **Calendar**).
Pinch zoom	Using two fingers, touch and pinch your fingers together, or spread them apart.	Zoom in and then back out on the screen (e.g., enlarging text and pictures).

Fig. 1.6 | Android gestures.

Using the multitouch screen, you can navigate easily between your phone, apps, music library, web browsing, and so on. The screen can display a keyboard for typing e-mails and text messages and entering data in apps (some Android devices also have hard keyboards). Using two fingers, you can zoom in (moving your fingers apart) and out (pinching your fingers together) on photos, videos and web pages. You can scroll up and down or side to side by just swiping your finger across the screen.

Built-in Apps

Android devices come with several built-in apps, which may vary depending on the device. These typically include **Phone**, **Contacts**, **Mail**, **Browser** and more (Fig. 1.7). Many manufacturers customize the default apps; we'll show you how to interact with the apps regardless of how they've been changed.

Android Naming Convention

Each new version of Android is named after a dessert, going in alphabetical order:

- Android 1.6 (Donut)
- Android 2.0–2.1 (Eclair)
- Android 2.2 (Froyo)
- Android 2.3 (Gingerbread)
- Android 3.0 (Honeycomb)

1.3 Android 2.2 (Froyo)

Android 2.2 (also called ***Froyo***, released in May 2010) included several new features and enhancements (Fig. 1.7). In subsequent sections we'll discuss Android 2.3 (Gingerbread) and Android 3.0 (Honeycomb).

Feature	Description
Improved memory and performance	Upgrades include: • Dalvik Virtual Machine enhancements made it two to five times faster than in Android 2.1. • Chrome V8 engine quickly loads JavaScript web pages. • Kernel memory-management boost improves device performance.
Auto-discovery	Allows Exchange users to enter a username and password to quickly sync their Exchange accounts with their Android devices.
Calendar	Users can sync their Exchange Calendar with the **Calendar** app.
Global Address Lists (GAL) look-up	Accesses addresses for e-mail users and distribution lists in the user's Microsoft Exchange e-mail system, enabling auto-complete of recipients' contact names when creating a new e-mail.
Passwords	Users can add alphanumeric passwords to unlock a device. This enhances data security by preventing anyone from accessing information on the locked device.
Remote Wipe	If you're unable to find your Android device, the Remote Wipe feature restores it to the factory settings (removing all personal data), thus protecting the privacy of your information. Once you Remote Wipe the phone, any data that you haven't backed up will be lost. [Note: Availability of Remote Wipe varies by manufacturer and device policy managers.]

Fig. 1.7 | Android 2.2 user features (`developer.android.com/sdk/android-2.2-highlights.html`). (Part 1 of 3.)

Feature	Description
Contacts and accounts	The **Quick Contact** for Android gives users easy access to contact information and modes for communicating with their contacts, such as e-mail, SMS or phone. A user can tap a contact's photo (e.g., in the contacts list, image gallery, e-mail or calendar), bringing up the **Quick Contact** widget with the various communication modes. As a developer, you can incorporate **Quick Contact** into your apps.
Camera	The camera controls in Android 2.2 include camera flash support and digital zoom. Users can adjust the camera settings to account for their environment (e.g., night, sunset, action), add effects (e.g., sepia, red tint, blue tint) and more. You can program the camera's preview and capture settings and retrieve and encode video.
Android virtual keyboard	The keyboard layout has been improved, making typing on the multitouch screen easier, and ensuring that keyboard touches aren't missed when typing with two fingers.
Improved dictionary	The more sophisticated dictionary learns from the user's word usage and includes the user's contacts in the suggestions.
Browser	The browser's improved user interface features a new address bar that the user can tap for search and navigation, and double-tap to zoom in and back out on a web page. It also supports HTML5, which includes features such as video playback and drag and drop that were previously available only through third-party plugins, such as Adobe Flash. [*Note:* The Browser also supports Flash.]
Multiple-languages keyboard	Users can add keyboards in other languages and easily switch among them by "flinging" from right to left across the space bar on the keyboard. To add keyboards, either on a device or in the emulator, go to **Settings > Language & keyboard > Android keyboard > Input languages.**
Media framework	Android's ***Stagefright media framework*** enables video playback and HTTP progressive streaming—i.e., sending video over the Internet using the HyperText Transfer Protocol to a browser and playing the video even while it's still downloading. The previous media framework, OpenCORE, is still supported in Android.
Bluetooth	Users can now wirelessly connect their Android devices to other Bluetooth-enabled devices such as headsets and car docks (for connecting the phone to the car's hands-free phone system), share contact information with Bluetooth-enabled phones and voice dial.

Fig. 1.7 | Android 2.2 user features (`developer.android.com/sdk/android-2.2-highlights.html`). (Part 2 of 3.)

Feature	Description
Tethering and Wi-Fi hotspot support	Android 2.x included built-in tethering and Wi-Fi hotspot support, enabling users to connect their phone to their Windows or Linux computer with a USB cable to use the phone's 3G service to connect to the Internet (`www.engadget.com/2010/05/13/android-2-2-froyo-to-include-usb-tethering-wifi-hotspot-funct/`).

Fig. 1.7 | Android 2.2 user features (`developer.android.com/sdk/android-2.2-highlights.html`). (Part 3 of 3.)

New Developer Features in Android 2.2

The ***Android Cloud to Device Messaging (C2DM)*** service allows app developers to send data from their servers to their apps installed on Android devices, even when the apps are not currently running. The server notifies the apps to contact the server directly to receive updated app or user data.[5] ***Android Application Error Reports***, which can be accessed by logging into your Android Market publisher account, enable you to receive app-crash and app-freeze reports from your apps' users.

Android 2.2 also includes several new APIs that allow you to easily add functionality into your apps (Fig. 1.8). We use some of these new frameworks in this book. We also use ***web services***. With these, you can create ***mashups***, which enable you to rapidly develop apps by combining the complementary web services of several organizations, possibly with information feeds of various types (such as RSS, Atom, XML, JSON and others) (Fig. 1.9). For example, `www.housingmaps.com` uses web services to combine Craigslist (`www.craigslist.org`) real-estate listings with the capabilities of Google Maps—the most widely used API for mashups—to show the locations of apartments for rent in a given area. We use WeatherBug web services in Chapter 16's **Weather AppWidget**.

API	Description
Apps on external storage	Apps can be stored on an external memory device rather than just the Android device's internal memory.
Camera and camcorder	New features include the Camera Preview API which doubles the frame rate (now 20 frames per-second), portrait orientation, zoom controls, exposure data and a thumbnail utility. The new **CamcorderProfile** classes can be used in apps to determine the camcorder hardware capabilities of the user's device.
Data backup	Back-up data to the cloud and restore data after a user resets the device to the original factory settings or switches devices.

Fig. 1.8 | Android 2.2 APIs (`developer.android.com/sdk/android-2.2-highlights.html`). (Part 1 of 2.)

5. `code.google.com/android/c2dm/`.

API	Description
Device policy management	Create administrator apps to control device security features (e.g., password strength).
Graphics	Access to the OpenGL ES 2.0 graphics APIs which were previously available only through the Android NDK—a toolset that allows you to use native code for performance-critical app components (`developer.android.com/sdk/ndk/overview.html`).
Media framework	APIs for audio focus, auto-scanning files to the media database (e.g., audio and video files), detecting sound loading completion, auto-pause and auto-resume of audio playback, and more.
UI framework	The `UiModeManager` car mode, desk mode and night mode controls enable you to adjust an app's user interface, the scale gesture detector API improves multi-touch events, and the bottom strip of a **TabWidget** is now customizable.

Fig. 1.8 | Android 2.2 APIs (`developer.android.com/sdk/android-2.2-highlights.html`). (Part 2 of 2.)

Web services source	How they're used
Google Maps	Mapping services
Facebook	Social networking
Foursquare	Mobile check-in
LinkedIn	Social networking for business
YouTube	Video search
Twitter	Microblogging
Groupon	Social commerce
Netflix	Movie rentals
eBay	Internet auctions
Wikipedia	Collaborative encyclopedia
PayPal	Payments
Last.fm	Internet radio
Amazon eCommerce	Shopping for books and more
Salesforce.com	Customer Relationship Management (CRM)
Skype	Internet telephony
Microsoft Bing	Search
Flickr	Photo sharing
Zillow	Real-estate pricing
Yahoo Search	Search
WeatherBug	Weather

Fig. 1.9 | Some popular web services (`www.programmableweb.com/apis/directory/1?sort=mashups`).

Figure 1.10 lists directories where you'll find information about many of the most popular web services.

Directory	URL
ProgrammableWeb	`www.programmableweb.com`
Webmashup.com	`www.webmashup.com/`
Webapi.org	`www.webapi.org/webapi-directory/`
Google Code API Directory	`code.google.com/apis/gdata/docs/directory.html`
APIfinder	`www.apifinder.com/`

Fig. 1.10 | Web-services directories.

1.4 Android 2.3 (Gingerbread)

Android 2.3 (Gingerbread), released in December 2010 (with Android 2.3.3—a minor update—released in February 2011), added more user refinements, such as a redesigned keyboard, improved navigation capabilities, increased power efficiency and more. Figure 1.11 describes some of the key new user features and updates.

Feature	Description
Power management	Apps that consume processor power while running in the background, or are awake longer than normal, can be closed by Android (if appropriate) to save battery power and improve performance. Users can also view the apps and system components consuming battery power.
Manage Applications shortcut	The **Manage Applications** shortcut in the **Options** menu on the Home screen allows users to view all apps that are running. For each app, you can view the amount of storage and memory it's using, permissions the app has been granted (whether it can read the user's contact data, create Bluetooth connections, etc.) and more. Users can also "force-stop" the app.
Near-field communications	***Near-field communication (NFC)*** is a short-range wireless connectivity standard that enables communication between two devices, or a device and a tag (which stores data that can be read by NFC-enabled devices), within a few centimeters. NFC-enabled devices can operate in three modes—reader/writer (e.g., reading data from a tag), peer to peer (e.g., exchanging data between two devices) and card emulation (e.g., acting like a smart card for contactless payments). NFC-enabled Android devices can be used in reader/writer and peer-to-peer modes. NFC support and features vary by Android device.

Fig. 1.11 | Android 2.3 user features (`developer.android.com/sdk/android-2.3-highlights.html`). (Part 1 of 2.)

Feature	Description
Improved **Copy** and **Paste** functionality	You can touch a word to select it, drag the markers to adjust the selection, copy the text by touching the highlighted area, then paste the text. You can also move the cursor by dragging the cursor arrow.
Camera	Apps can access both rear-facing and front-facing cameras.
Internet calling	Android includes Session Initiation Protocol (SIP) support—an Internet Engineering Task Force (IETF) standard protocol for initiating and terminating voice calls over the Internet. Users with SIP accounts (available through third parties) can make Internet voice calls to other contacts with SIP accounts. Not all Android devices or carriers support SIP and Internet calling. For a list of SIP providers, see `www.cs.columbia.edu/sip/service-providers.html`.
Downloads app	Users can access files downloaded from e-mail, the browser, etc. through the **Downloads** app.

Fig. 1.11 | Android 2.3 user features (`developer.android.com/sdk/android-2.3-highlights.html`). (Part 2 of 2.)

The platform also added several new developer features for enhanced communications, game development and multimedia (Figure 1.12). For further details about each of these features, go to `developer.android.com/sdk/android-2.3-highlights.html`.

Feature	Description
Internet telephony	The new SIP support allows you to build Internet telephony functionality into your apps—namely, making and receiving voice calls.
Near-field communications API	Build apps that read and respond to data from NFC tags or devices. Android 2.3.3 apps can also write to tags and work in peer-to-peer mode with other devices. Note that NFC support varies by Android device.
Audio effects API	Add equalization (for adjusting bass or treble), bass boost (increasing the volume of bass sounds), headphone virtualization (simulated surround sound), and reverb (echo effects) to an audio track or across multiple tracks.
New audio formats	Built-in support for Advanced Audio Coding (AAC—a successor to MP3) and Adaptive Multi-Rate Wideband encoding (AMR-WB) for capturing high-quality audio.
New video formats	Built-in support for VP8 open video compression with the WebM open-container format.

Fig. 1.12 | Android 2.3 developer features (`developer.android.com/sdk/android-2.3-highlights.html`). (Part 1 of 2.)

Feature	Description
Camera API	Use the enhanced Camera API to access rear- and front-facing cameras on a device, determine their features and open the appropriate camera.

Fig. 1.12 | Android 2.3 developer features (`developer.android.com/sdk/android-2.3-highlights.html`). (Part 2 of 2.)

1.5 Android 3.0 (Honeycomb)

Tablet sales will account for over 20% of all personal-computer sales by 2015.[6] Interest in Android tablets is increasing rapidly. At the 2011 Consumer Electronic Show, 85 new Android tablets were announced.[7] ***Android 3.0 (Honeycomb)*** includes user-interface improvements specifically for large-screen devices (e.g., tablets), such as a redesigned keyboard for more efficient typing, a visually appealing 3D user interface, System and Action Bars for easier navigation and more (Fig. 1.13). It also gives developers new tools to optimize apps for larger-screen devices (Fig. 1.14).

Feature	Description
Holographic UI	Attractive 3D-looking user interface.
Customizable home screens	Organize widgets, app shortcuts and more.
Redesigned keyboard	Enables improved typing accuracy and efficiency.
Improved editing	New user interface makes it easier to select, copy and paste text.
System Bar	Quickly access navigation buttons, notifications and system status from the System Bar at the bottom of the screen.
Action Bar	Provides app-specific controls (such as navigation) from the Action Bar at the top of each app's screen.
Improved multitasking	The **Recent Apps** list in the System Bar allows you to see the tasks that are running simultaneously and switch between apps.
Connectivity options	Connect your Android device to a keyboard using either USB or Bluetooth.
Photo Transfer Protocol (PTP) and Media Transfer Protocol (MTP) support	Developed by Microsoft, these protocols enable you to transfer photos, videos and music files to your computer. You can create apps that allow users to create and manage media files and share them on multiple devices.

Fig. 1.13 | New Android 3 features (`developer.android.com/sdk/android-3.0-highlights.html`). (Part 1 of 2.)

6. `www.forrester.com/ER/Press/Release/0,1769,1340,00.html`.
7. `www.computerworld.com/s/article/9206219/Google_Android_tablets_gain_traction_with_developers?source=CTWNLE_nlt_dailyam_2011-01-25`.

Feature	Description
Bluetooth tethering	Connect to a Wi-Fi or 3G network on your computer or other devices using your Android device as a modem.
Browser	Features tabs instead of multiple windows, easier browsing of non-mobile sites (using improved zoom, scrolling, etc.), "incognito" mode for browsing sites anonymously, multitouch support for JavaScript and plugins and more. You can also automatically sign into Google sites and sync your bookmarks with Google Chrome.
Camera	Redesigned for larger-screen devices, you can easily access camera features such as the front-facing camera, flash, auto-focus and more. The time-lapse video recording capabilities allow you to capture "frames" at a slower-than-normal rate, then play the video back at normal speed, making it appear as though time is moving faster.
Contacts	The two-pane user interface makes it easier to read, edit and organize contacts. Fast scroll helps you find contacts quickly.
Email	Use the Action Bar to organize e-mail in folders and sync attachments. You can also use the e-mail widget on your home screen to easily monitor your messages.
Gallery	View albums in full-screen mode, with thumbnail images to view other photos in the album.

Fig. 1.13 | New Android 3 features (`developer.android.com/sdk/android-3.0-highlights.html`). (Part 2 of 2.)

Feature	Description
Backward compatibility	Android 3.x is compatible with apps developed using previous versions of Android.
Holographic UI	Give your new and existing apps the new Android 3 holographic look and feel by adding an attribute in the app's manifest file.
Add layouts for large-screen devices to existing apps	Add new layouts and assets for large-screen devices to your existing apps designed for small-screen devices.
Activity fragments	Divide an app's Activities into modularized Fragments, which can be used in a variety of combinations. Google is enhancing this API so it can be used on Android 1.6 and later.
New and updated UI and Home-screen widgets	Include a search box, calendar, 3D stack, a date/time picker, number picker and more. Home-screen widgets can now be controlled with touch gestures to scroll and flip through the content.
Action Bar	Each app now has its own persistent Action Bar, providing users with options for navigation, etc.

Fig. 1.14 | New developer features in Android 3 (`developer.android.com/sdk/android-3.0-highlights.html`). (Part 1 of 2.)

Feature	Description
Enhancements for gaming	Enhancements for gaming include: • Performance enhancements such as a concurrent garbage collector, faster event distribution and updated video drivers. • Native input and sensor events • New sensors—gyroscope, barometer, gravity sensor and more—for better 3D motion processing. • Khronos OpenSL ES API for native audio. • Khronos EGL library for native graphics management. • Native access to the Activity lifecycle, and APIs for managing windows. • Native Asset Manager API and Storage Manager API.
Additional notifications capabilities	Add large and small icons, titles and priority flags to your apps' notifications using the builder class.
Clipboard	Allows users to copy and paste data across multiple apps.
Drag and drop	Use the `DragEvent` framework to add drag-and-drop capabilities in an app.
Multiselect	Allow users to select *multiple* items from a list or grid.
Media/Picture Transfer Protocol (MTP/PTP)	Allows users to easily transfer any type of media files between devices and to a host computer.
Multicore processor architecture support	Run Android 3.x on single-core or multicore processor architectures for enhanced performance.
HTTP Live Streaming (HLS)	Apps can provide a URL for a multimedia playlist to the media framework to launch an HTTP Live Streaming session. This provides higher quality support for adaptive video.
Renderscript 3D graphics engine	Create high-performance 3D graphics for apps, widgets, etc. and offloading calculations to the Graphics Processing Unit (GPU).
Hardware-accelerated 2D graphics	The new OpenGL renderer improves performance of common graphics operations.
New animation framework	Easily animate user-interface elements or objects.
Bluetooth A2DP and HSP	APIs for Bluetooth Advanced Audio Distribution Profile (A2DP) and Headset Profile (HSP) allow your apps to check for connected Bluetooth devices, battery level and more.
Digital Rights Management (DRM) framework	API that enables you to manage protected content in your apps.
New policies for device administration apps	Enterprise device-administration apps can now support policies such as password expiration and more.

Fig. 1.14 | New developer features in Android 3 (`developer.android.com/sdk/android-3.0-highlights.html`). (Part 2 of 2.)

1.6 Android Ice Cream Sandwich

Android Ice Cream Sandwich, scheduled to be released in late 2011, will merge Android 2.3 (Gingerbread) and Android 3.0 (Honeycomb) into one operating system for use on all Android devices. This will allow you to incorporate Honeycomb's features such as the holographic user interface, new launcher and more (previously available only on tablets) into your smartphone apps, and easily scale your apps to work on different devices. Ice Cream Sandwich will also add new functionality (Fig. 1.15).

Feature	Description
0-click NFC Peer-to-Peer Sharing	Users with compatible Android devices will be able to share content (e.g., contacts, videos) just by placing the devices near each other.
Head tracking	Using the camera, compatible devices will determine the positioning of the user's eyes, nose and mouth. The camera will also be able to track where the user is looking, allowing you to create apps that change perspective based on where the user is looking (e.g., 3D game landscapes).
Virtual camera operator	When taking video, the camera will automatically focus on the person speaking. For example, if two people are participating in a video chat, the camera will determine which of the two is speaking and focus the camera on that person.
Android@Home framework	Will enable you to create Android apps to control appliances in the user's home, such as turning lights on and off (with special light bulbs from Lighting Science), adjusting the thermostat, controlling the irrigation system and more.

Fig. 1.15 | Some Android Ice Cream Sandwich features.

1.7 Downloading Apps from the Android Market

At the time of this printing, there were hundreds of thousands of apps in Android Market, and the number continues to grow quickly. Figure 1.16 lists some popular Android apps. You can download additional apps directly onto your Android device through Google's ***Android Market***. Android Market notifies you when updates to your downloaded apps are available.

Android Market Category	Sample apps
Comics	Marvel Superheroes, Dilbert Calendar, Jerry Seinfeld Jokes
Communication	Google Voice, Skype mobile™, Wi-Fi Locator, Easy
Entertainment	Face Melter, Fingerprint Scanner, Fandango® Movies

Fig. 1.16 | Some popular Android apps in Android Market. (Part 1 of 2.)

Android Market Category	Sample apps
Finance	Mint.com Personal Finance, PayPal, Debt Payoff Planner
Games: Arcade & Action	NESoid, Droid Breakout, Raging Thunder 2 Lite, Whac 'em!
Games: Brain & Puzzle	Enjoy Sudoku, Spin Cube Lite, Ultimate Simpson Puzzle
Games: Cards & Casino	Texas Hold'em Poker, Tarot Cards, Chessmaster™
Games: Casual	City Mayor, LOL Libs, Paper Toss, SuperYatzy Free Edition
Health	Fast Food Calorie Counter, CardioTrainer, StopSmoking
Lifestyle	Zillow Real Estate, Epicurious Recipe App, Family Locator
Multimedia	Pandora Radio, Shazam, Last.fm, iSyncr, Camera Illusion
News & Weather	The Weather Channel, CNN, NYTimes, FeedR News Reader
Productivity	Adobe® Reader®, Documents To Go 2.0 Main App
Reference	Google Sky Map, Dictionary.com, Wikidroid for Wikipedia
Shopping	Gluten Free, Amazon.com, Barcode Scanner, Pkt Auctions eBay
Social	Facebook®, Twitter for Android, MySpace, Bump, AIM
Sports	NFL Mobile, Nascar Mobile, Google Scoreboard
Themes	Pixel Zombies Live Wallpaper, Aquarium Live Wallpaper
Tools	Compass, Droidlight LED Flashlight, AppAlarm Pro
Travel	Google Earth, Yelp®, Urbanspoon, WHERE, XE Currency
Demo	Screen Crack, Bubbles, CouponMap, SnowGlobe
Software libraries	Translate Tool, Security Guarder, Car Locator Bluetooth Plugin

Fig. 1.16 | Some popular Android apps in Android Market. (Part 2 of 2.)

Visit `market.android.com` to check out the featured apps, or check out some of the other Android app review and recommendation sites (Fig. 1.17). Some are free and some are fee based. Developers set the prices for their apps sold through Android Market and receive 70% of the revenue. As a marketing strategy, many app developers offer basic versions of their apps for free so users can determine whether they like them, then purchase more feature-rich versions. We discuss this so-called "lite" strategy in more detail in Section 2.8.

Name	URL
AppBrain	`www.appbrain.com/`
AndroidLib	`www.androlib.com/`
Android Tapp™	`www.androidtapp.com/`
Appolicious™	`www.androidapps.com/`
AndroidZoom	`www.androidzoom.com/`
doubleTwist®	`www.doubletwist.com/apps/`
mplayit™	`mplayit.com/#homepage`

Fig. 1.17 | Android app review and recommendation sites.

1.8 Packages

Android uses a collection of packages, which are named groups of related, predefined classes. Some of the packages are Android specific, while others are Java and Google packages. These packages allow you to conveniently access Android OS features and incorporate them into your apps. They're written mainly in Java and are accessible to Java programs. The Android packages help you create apps that adhere to Android's unique look-and-feel conventions. Figure 1.18 lists the packages we discuss in this book. For a complete list of Android packages, see `developer.android.com/reference/packages.html`.

Packages	Description
`android.app`	Includes high-level classes in the Android app model. (Chapter 4's **Tip Calculator** app.)
`android.os`	Operating-systems services. (Chapter 4's **Tip Calculator** app.)
`android.text`	Rendering and tracking text on the device. (Chapter 4's **Tip Calculator** app.)
`android.widget`	User-interface classes for widgets. (Chapter 4's **Tip Calculator** app.)
`android.net`	Network access classes. (Chapter 5's **Favorite Twitter® Searches** app.)
`android.view`	User interface classes for layout and user interactions. (Chapter 5's **Favorite Twitter® Searches** app.)
`java.io`	Streaming, serialization and file-system access of input and output facilities. (Chapter 5's **Favorite Twitter® Searches** app.)
`java.util`	Utility classes. (Chapter 5's **Favorite Twitter® Searches** app.)
`android.content.res`	Classes for accessing app resources (e.g., media, colors, drawables, etc.), and device-configuration information affecting app behavior. (Chapter 6's **Flag Quiz Game** app.)
`android.graphics.drawable`	Classes for display-only elements (e.g., gradients, etc.). (Chapter 6's **Flag Quiz Game** app.)
`android.media`	Classes for handling audio and video media interfaces. (Chapter 7's **Spotz Game** app.)
`android.util`	Utility methods and XML utilities. (Chapter 8's **Cannon Game** app.)
`android.content`	Access and publish data on an Android device. (Chapter 9's **Doodlz** app.)
`android.hardware`	Device hardware support. (Chapter 9's **Doodlz** App and Chapter 13's **Enhanced Slideshow** app.)
`android.provider`	Access to Android content providers. (Chapter 9's **Doodlz** app.)
`android.database`	Handling data returned by the content provider. (Chapter 10's **Address Book** app.)

Fig. 1.18 | Android, Java and Google packages used in this book, listed with the chapter in which they *first* appear. (Part 1 of 2.)

Packages	Description
`android.database.sqlite`	SQLite database management for private databases. (Chapter 10's **Address Book** app.)
`android.graphics`	Graphics tools used for drawing to the screen. (Chapter 11's **Route Tracker** app.)
`android.location`	Location-based services. (Chapter 11's **Route Tracker** app.)
`com.google.android.maps`	Used in Chapter 11's **Route Tracker** app.
`android.appwidget`	Used in Chapter 16's **Weather AppWidget**.
`java.net`	Networking classes (e.g., handling Internet addresses and HTTP requests). (Chapter 16's **Weather App Widget**.)
`javax.xml.parsers`	Processing XML documents. (Chapter 16's **Weather App Widget**.)
`org.xml.sax`	Simple API for XML (SAX API) for reading data from XML documents. (Chapter 16's **Weather App Widget**.)
`android.speech`	Speech recognition classes. (Chapter 17's **Papa's Pizza** app.)
`android.speech.tts`	Text-to-speech classes. (Chapter 17's **Papa's Pizza** app.)
`android.telephony`	Phone APIs for monitoring network information, connection state and more. We'll use these APIs to send SMS messages. (Chapter 17's **Papa's Pizza** app.)
`android.opengl`	OpenGL graphics tools. (Chapter 18's **3D Art** app.)
`java.nio`	Buffers for handling data. (Chapter 18's **3D Art** app.)
`javax.microedition.khronos.egl`	Khronos EGL APIs for 3D graphics. (Chapter 18's **3D Art** app.)
`javax.microedition.khronos.opengles`	Khronos OpenGL® ES interfaces. (Chapter 18's **3D Art** app.)

Fig. 1.18 | Android, Java and Google packages used in this book, listed with the chapter in which they *first* appear. (Part 2 of 2.)

1.9 Android Software Development Kit (SDK)

The Android SDK provides the tools you'll need to build Android apps. It's available at no charge through the Android Developers site. See the Before You Begin section after the Preface for complete details on downloading the tools you need to develop Android apps, including the Java SE, the Eclipse IDE, the Android SDK 3.x and the ADT Plugin for Eclipse.

Eclipse Integrated Development Environment (IDE)

Eclipse is the recommended integrated development environment for Android development, though developers may also use a text editor and command-line tools to create Android apps. Eclipse supports many programming languages, including Java, C++, C, Python, Perl, Ruby on Rails and more. The vast majority of Android development is done in Java. The Eclipse IDE includes:

- Code editor with support for syntax coloring and line numbering

- Auto-indenting and auto-complete (i.e., type hinting)
- Debugger
- Version control system
- Refactoring support

You'll use Eclipse in Section 1.11 to test-drive the **Doodlz** app. Starting in Chapter 3, **Welcome** App, you'll use Eclipse to build apps.

Android Development Tools (ADT) Plugin for Eclipse

The ***Android Development Tools (ADT) Plugin for Eclipse***—an extension to the Eclipse IDE—allows you to create, run and debug Android apps, export them for distribution (e.g., upload them to Android Market), and more. ADT also includes a visual GUI design tool. GUI components can be dragged and dropped into place to form GUIs without any coding. You'll learn more about ADT in Chapter 3, **Welcome** App.

The Android Emulator

The Android emulator, included in the Android SDK, allows you to run Android apps in a simulated environment within Windows, Mac OS X or Linux. The emulator displays a realistic Android user-interface window. Before running an app in the emulator, you'll need to create an ***Android Virtual Device (AVD)***, which defines the characteristics of the device on which you want to test, including the hardware, system image, screen size, data storage and more. If you want to test your apps for multiple Android devices, you'll need to create separate AVDs to emulate each unique device.

We used the emulator (not an actual Android device) to take most of the Android screen shots for this book. You can reproduce on the emulator most of the Android gestures (Fig. 1.19) and controls (Fig. 1.20) using your computer's keyboard and mouse. The gestures on the emulator are a bit limited, since your computer probably cannot simulate all the Android hardware features. For example, to test GPS apps in the emulator, you'll need to create files that simulate GPS readings. Also, although you can simulate orientation changes (to portrait or landscape mode), there's no way to simulate particular ***accelerometer*** readings (the accelerometer measures the orientation and tilting of the device). You can, however, upload your app to an Android device to test these features. You'll see how to do this in Chapter 11, **Route Tracker** app. You'll start creating AVDs and using the emulator to develop Android apps in Chapter 3's **Welcome** app.

Gesture	Emulator action
Tap	Click the mouse once. Introduced in Chapter 4's **Tip Calculator** app.
Double tap	Double-click the mouse. Introduced in Chapter 8's **Cannon Game** app.
Long press	Click and hold the mouse.
Drag	Click, hold and drag the mouse. Introduced in Chapter 8's **Cannon Game** app.

Fig. 1.19 | Android gestures on the emulator (`developer.android.com/guide/developing/tools/emulator.html`). (Part 1 of 2.)

Gesture	Emulator action
Swipe	Click and hold the mouse, move the pointer in the swipe direction and release the mouse. Introduced in Chapter 10's **Address Book** app.
Fling	Click and hold the mouse, move the pointer in the flick direction and quickly release. Introduced in Chapter 10's **Address Book** app.
Pinch	Press and hold the *Ctrl* (*Control*) key. Two circles that simulate the two touches will appear. Move the circles to the start position, click and hold the mouse and drag the circles to the end position. Introduced in Chapter 11's **Route Tracker** app.

Fig. 1.19 | Android gestures on the emulator (`developer.android.com/guide/developing/tools/emulator.html`). (Part 2 of 2.)

Control	Emulator action
Back	*Esc*
Call/dial button	*F3*
Camera	*Ctrl-KEYPAD_5*, *Ctrl-F3*
End call button	*F4*
Home	*Home* button
Menu (left softkey)	*F2* or *Page Up* button
Power button	*F7*
Search	*F5*
* (right softkey)	*Shift-F2* or *Page Down* button
Switch to previous orientation (e.g., landscape or portrait)	*KEYPAD_7*, *Ctrl-F11*
Switch to next orientation (e.g., landscape or portrait)	*KEYPAD_9*, *Ctrl-F12*
Toggle cell networking on/off	*F8*
Volume up button	*KEYPAD_PLUS*, *Ctrl-F5*
Volume down button	*KEYPAD_MINUS*, *Ctrl-F6*

Fig. 1.20 | Android hardware controls on the emulator (for additional controls, go to `developer.android.com/guide/developing/tools/emulator.html`).

1.10 Object Technology: A Quick Refresher

Building software quickly, correctly and economically remains an elusive goal at a time when demands for new and more powerful software are soaring. *Objects*, or more precisely—as we'll see in Chapter 3—the *classes* objects come from, are essentially *reusable* software components. There are date objects, time objects, audio objects, video objects, automobile objects, people objects, etc. Almost any *noun* can be reasonably represented as a software object in terms of *attributes* (e.g., name, color and size) and *behaviors* (e.g., calculating, moving and communicating). Software developers are discovering that using a modular, object-oriented design and implementation approach can make software development groups much more productive than was possible with earlier popular techniques like "structured programming"—object-oriented programs are often easier to understand, correct and modify.

The Automobile as an Object

To help you understand objects and their contents, let's begin with a simple analogy. Suppose you want to *drive a car and make it go faster by pressing its accelerator pedal.* What must happen before you can do this? Well, before you can drive a car, someone has to *design* it. A car typically begins as engineering drawings, similar to the *blueprints* that describe the design of a house. These drawings include the design for an accelerator pedal. The pedal *hides* from the driver the complex mechanisms that actually make the car go faster, just as the brake pedal hides the mechanisms that slow the car, and the steering wheel "hides" the mechanisms that turn the car. This enables people with little or no knowledge of how engines, braking and steering mechanisms work to drive a car easily.

Just as you cannot cook meals in the kitchen of a blueprint, you cannot drive a car's engineering drawings. Before you can drive a car, it must be *built* from the engineering drawings that describe it. A completed car has an *actual* accelerator pedal to make the car go faster, but even that's not enough—the car won't accelerate on its own (hopefully!), so the driver must *press* the pedal to accelerate the car.

Methods and Classes

Let's use our car example to introduce some key object-oriented programming concepts. Performing a task in a program requires a ***method***. The method houses the program statements that actually perform its tasks. The method hides these statements from its user, just as the accelerator pedal of a car hides from the driver the mechanisms of making the car go faster. In Java, we create a program unit called a ***class*** to house the methods that perform the class's tasks. For example, a class that represents a bank account might contain one method to *deposit* money to an account, another to *withdraw* money from an account and a third to *inquire* what the account's current balance is. A class is similar in concept to a car's engineering drawings, which house the design of an accelerator pedal, steering wheel, and so on.

Instantiation

Just as someone has to *build a car* from its engineering drawings before you can actually drive a car, you must *build an object* of a class before a program can perform the tasks that the class's methods define. The process of doing this is called *instantiation*. An object is then referred to as an ***instance*** of its class.

Reuse

Just as a car's engineering drawings can be *reused* many times to build many cars, you can *reuse* a class many times to build many objects. Reuse of existing classes when building new classes and programs saves time and effort. Reuse also helps you build more reliable and effective systems, because existing classes and components often have gone through extensive *testing*, *debugging* and *performance* tuning. Just as the notion of *interchangeable parts* was crucial to the Industrial Revolution, reusable classes are crucial to the software revolution that has been spurred by object technology.

Messages and Methods Calls

When you drive a car, pressing its gas pedal sends a *message* to the car to perform a task—that is, to go faster. Similarly, you *send messages to an object*. Each message is a ***method call*** that tells a method of the object to perform its task. For example, a program might call a particular bank-account object's *deposit* method to increase the account's balance.

Attributes and Instance Variables

A car, besides having capabilities to accomplish tasks, also has *attributes*, such as its color, its number of doors, the amount of gas in its tank, its current speed and its record of total miles driven (i.e., its odometer reading). Like its capabilities, the car's attributes are represented as part of its design in its engineering diagrams (which, for example, include an odometer and a fuel gauge). As you drive an actual car, these attributes are carried along with the car. Every car maintains its *own* attributes. For example, each car knows how much gas is in its own gas tank, but *not* how much is in the tanks of *other* cars.

An object, similarly, has attributes that it carries along as it's used in a program. These attributes are specified as part of the object's class. For example, a bank-account object has a *balance attribute* that represents the amount of money in the account. Each bank-account object knows the balance in the account it represents, but *not* the balances of the *other* accounts in the bank. Attributes are specified by the class's ***instance variables***.

Encapsulation

Classes ***encapsulate*** (i.e., wrap) attributes and methods into objects—an object's attributes and methods are intimately related. Objects may communicate with one another, but they're normally not allowed to know how other objects are implemented—implementation details are *hidden* within the objects themselves. This ***information hiding***, as we'll see, is crucial to good software engineering.

Inheritance

A new class of objects can be created quickly and conveniently by ***inheritance***—the new class absorbs the characteristics of an existing class, possibly customizing them and adding unique characteristics of its own. In our car analogy, an object of class "convertible" certainly *is an* object of the more *general* class "automobile," but more *specifically*, the roof can be raised or lowered.

Object-Oriented Analysis and Design (OOAD)

How will you create the code for your programs? Perhaps, like many programmers, you'll simply turn on your computer and start typing. This approach may work for small programs, but what if you were asked to create a software system to control thousands of automated teller machines for a major bank? Or suppose you were asked to work on a team of 1,000 software developers building the next U.S. air traffic control system? For projects so large and complex, you should not simply sit down and start writing programs.

To create the best solutions, you should follow a detailed ***analysis*** process for determining your project's ***requirements*** (i.e., defining *what* the system is supposed to do) and developing a ***design*** that satisfies them (i.e., deciding *how* the system should do it). Ideally, you'd go through this process and carefully review the design (and have your design reviewed by other software professionals) before writing any code. If this process involves analyzing and designing your system from an object-oriented point of view, it's called an

object-oriented analysis and design (OOAD) process. Languages like Java are object oriented. Programming in such a language, called ***object-oriented programming (OOP),*** allows you to implement an object-oriented design as a working system.

1.11 Test-Driving the Doodlz App in an Android Virtual Device (AVD)

In this section, you'll run and interact with your first Android app. The **Doodlz** app allows the user to "paint" on the screen using different brush sizes and colors. You'll build this app in Chapter 9. The following steps show how to import the app's project into Eclipse and how to test-drive the app in the Android Virtual Device (AVD) that you set up in the Before You Begin section following the Preface. Later in this section, we'll also discuss how to run the app on an actual Android device.

The screen captures in the following steps (and throughout this book) were taken on a computer running Windows 7, Java SE 6, Eclipse 3.6.1, Android 2.2/2.3/3.0 and the ADT Plugin for Eclipse.

1. *Checking your setup.* Confirm that you've set up your computer properly to develop Android apps by reading the Before You Begin section located after the Preface.
2. *Opening Eclipse.* To start Eclipse, open the folder containing Eclipse on your system and double-click the Eclipse (●) icon. If this is your first time opening Eclipse, the ***Welcome*** tab (Fig. 1.21) will open. Click the **Workbench** button to close this tab and switch to the program development view—this is formally called the *Java perspective* in Eclipse.

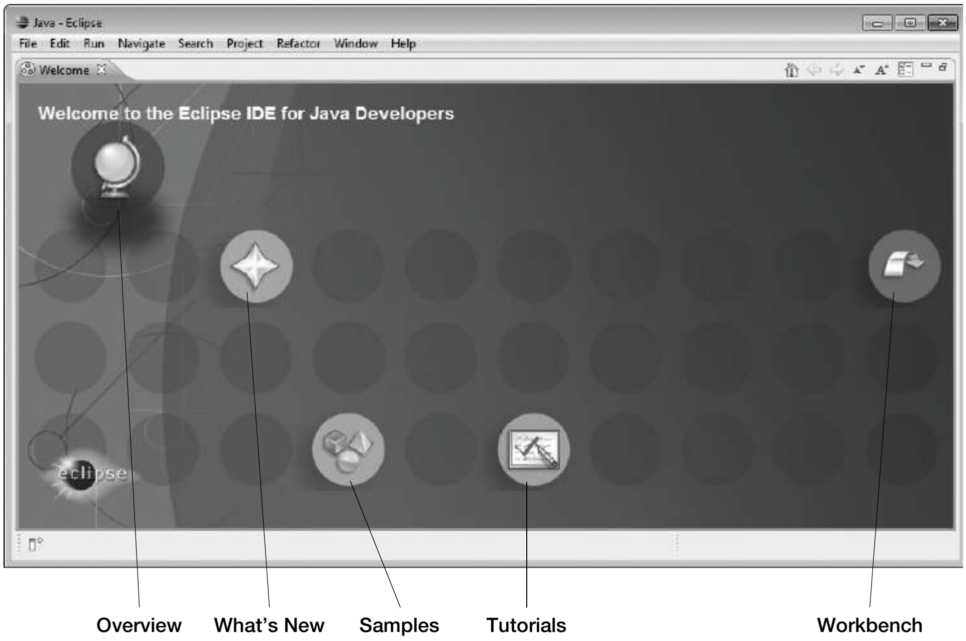

Java - Eclipse
File Edit Run Navigate Search Project Refactor Window Help
Welcome
Welcome to the Eclipse IDE for Java Developers
eclipse
Overview
What's New
Samples
Tutorials
Workbench

Fig. 1.21 | **Welcome to Eclipse** tab in Eclipse.

3. *Opening the* **Import** *Dialog.* Select **File > Import...** to open the **Import** dialog (Fig. 1.22).

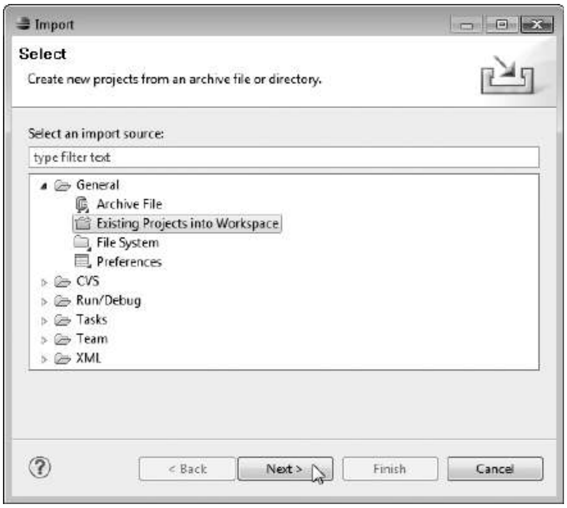


Fig. 1.22 | **Import** dialog.

4. *Importing the* **Doodlz** *app's project.* In the **Import** dialog, expand the **General** node and select **Existing Projects into Workspace**, then click **Next >** to proceed to the **Import Projects** step (Fig. 1.23). Ensure that **Select root directory** is selected, then

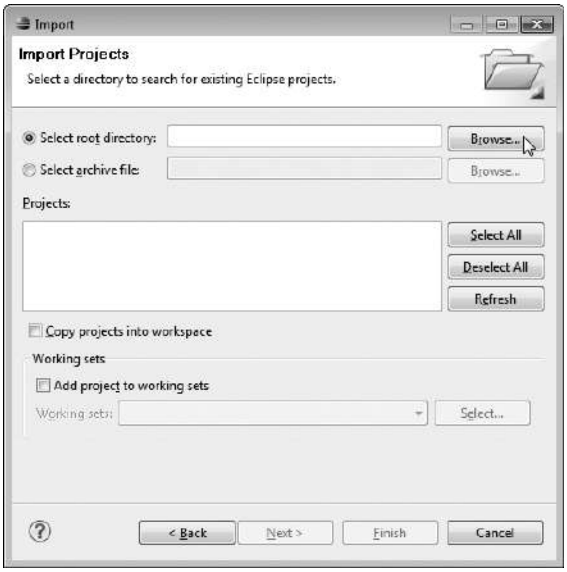


Fig. 1.23 | **Import** dialog's **Import Projects** step.

click the **Browse...** button. In the **Browse For Folder** dialog (Fig. 1.24), locate the **Doodlz** folder in the book's examples folder, select it and click **OK**. Click **Finish** to import the project into Eclipse. The project now appears in the **Package Explorer** window (Fig. 1.25) at the left side of the Eclipse window.

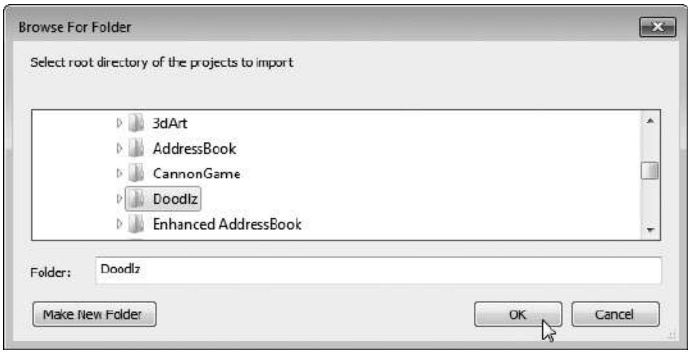


Fig. 1.24 | **Browser For Folder** dialog.

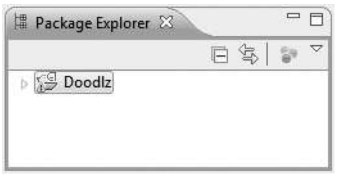


Fig. 1.25 | **Package Explorer** window in Eclipse.

5. ***Launching the* Doodlz *app.*** In Eclipse, select the **Doodlz** project in the **Package Explorer** window (Fig. 1.25), then select **Run As > Android Application** from the **Run As** button (▶ ▾) drop-down menu on the IDE's toolbar (Fig. 1.26). This will execute **Doodlz** in the Droid Android Virtual Device (AVD) (Fig. 1.27) that you created in the Before You Begin section. If you prefer to test the app in a different AVD, you select **Window > Android SDK and AVD Manager**, then select the AVD you wish to use and click **Start...**. If multiple AVDs are running when you launch an app, the **Android Device Chooser** dialog will appear to allow you to choose the AVD on which to execute the app. We'll discuss the **Android Device Chooser** dialog later in this section.
6. ***Exploring the AVD.*** The left side of the AVD displays the running app. The right side (Fig. 1.28) contains various buttons that simulate the hard and soft buttons on an actual Android device and a keyboard that simulates the device's hard or soft keyboard. ***Hard buttons*** are actual buttons on a device. ***Soft buttons*** are buttons that appear on the device's touch screen. You use the AVD's buttons to interact with apps and the Android OS in the AVD. When the app is installed on an Android device, you can create a new painting by dragging your finger anywhere on the canvas. In the AVD, you "touch" the screen by using the mouse.

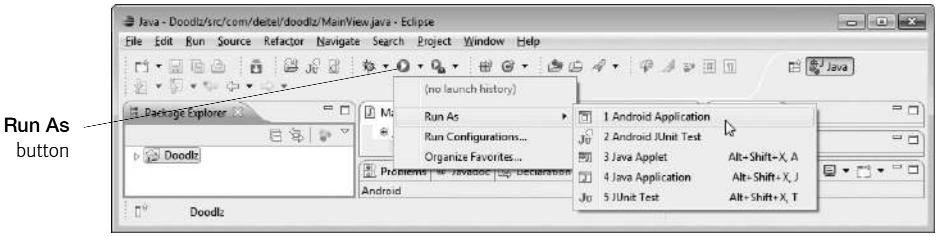

Run As button
Java - Doodlz/src/com/deitel/doodlz/MainView.java - Eclipse
File Edit Run Source Refactor Navigate Search Project Window Help
(no launch history)
Run As
Run Configurations...
Organize Favorites...
1 Android Application
2 Android JUnit Test
3 Java Applet Alt+Shift+X, A
4 Java Application Alt+Shift+X, J
5 JUnit Test Alt+Shift+X, T
Package Explorer
Doodlz

Fig. 1.26 | Launching the **Doodlz** app.

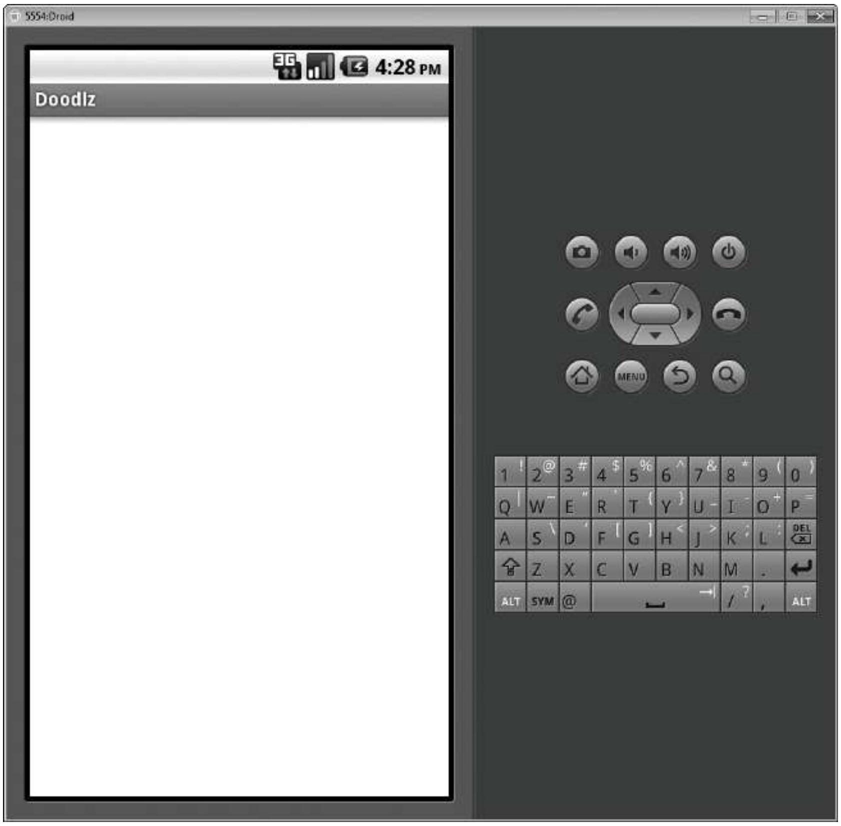


Fig. 1.27 | Android Virtual Device (AVD) with the running **Doodlz** app.

7. ***Displaying the app's options.*** To display the app's options, touch the **Menu** (MENU) button—on some actual devices this button appears as parallel horizontal bars (☰). The app now appears as shown in Fig. 1.29. The options include **Color**, **Line Width**, **Erase**, **Clear** and **Save Image**. Touching **Color** displays a GUI for changing the line color. Touching **Line Width** displays a GUI for changing the thickness of the line that will be drawn. Touching **Erase** sets the drawing color to

white so that as you draw over colored areas, the color is erased. Touching **Clear** clears the entire drawing. Touching **Save Image** saves the image into the device's **Gallery** of images. You'll explore each of these options momentarily.

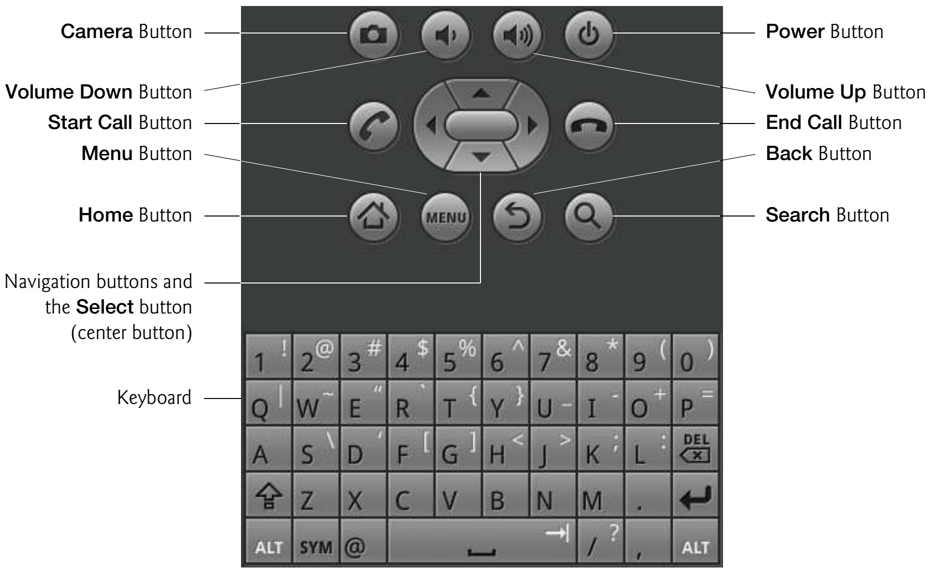


Fig. 1.28 | Android Virtual Device (AVD) with the running **Doodlz** app.

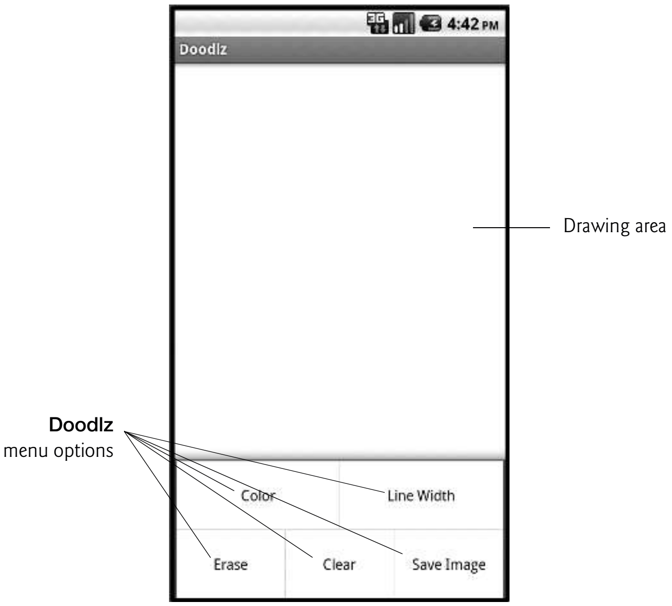


Fig. 1.29 | **Doodlz** menu options.

8. ***Changing the brush color to red.*** To change the brush color, first touch the **Color** menu item to display the GUI for changing the color (Fig. 1.30(a)). Colors are defined using the RGB color scheme in which the red, green and blue components are specified by integers in the range 0–255. The GUI consists of **Red**, **Green** and **Blue** `SeekBars` that allow you to select the amount of red, green and blue in the drawing color. Drag any of the three `SeekBars` to change the color. As you do so, a rectangle below the `SeekBars` displays the new color. Android also supports alpha transparency (partial transparency), which we discuss in Chapter 7's **Spot-On Game** app. Select a red color now by dragging the **Red** `SeekBar` to the right as in Fig. 1.30(a). Touch the **Done** button to return to the drawing area. Drag your "finger" (that is, the mouse) on the screen to draw flower petals (Fig. 1.30(b)).

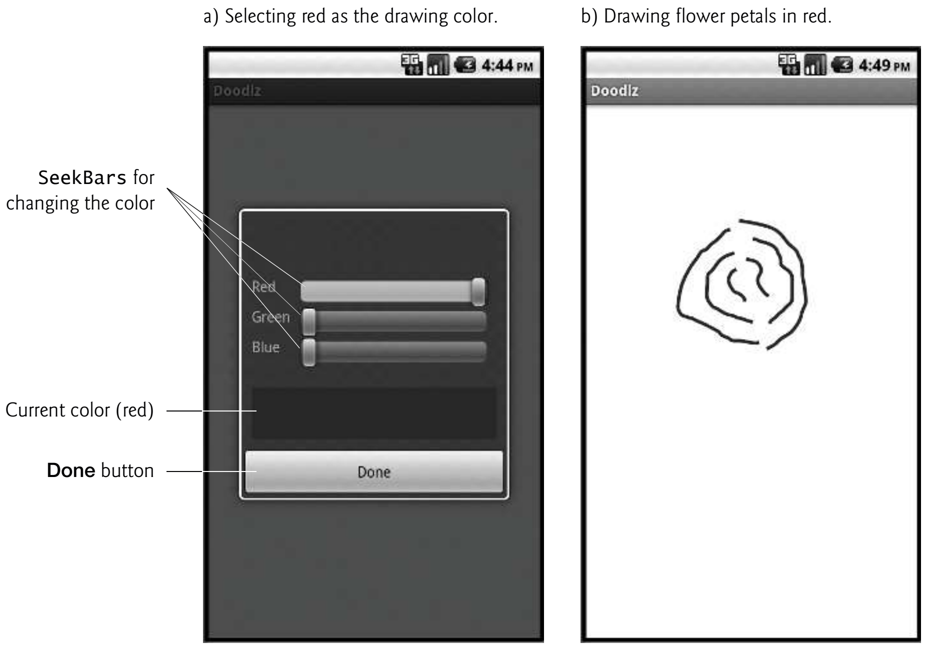

a) Selecting red as the drawing color.
b) Drawing flower petals in red.
SeekBars for changing the color
Current color (red)
Done button
4:44 PM
Doodlz
Red
Green
Blue
Done
4:49 PM
Doodlz

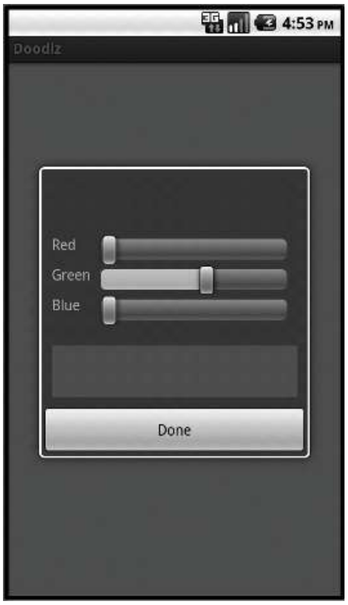
Fig. 1.30 | Changing the drawing color to red and drawing flower petals.

9. ***Changing the brush color to dark green.*** Change the drawing color again by touching the AVD's **Menu** (MENU) button, then touching **Color**. Select a dark green color by dragging the **Green** `SeekBar` to the right and ensuring that the **Red** and **Blue** `SeekBars` are at the far left (Fig. 1.31(a)).

10. ***Changing the line width.*** To change the line width, touch the **Menu** (MENU) button, then touch **Line Width**. Drag the `SeekBar` for the line width to the right to thicken the line (Fig. 1.31(b)). Touch the **Done** button to return to the drawing area. Draw the flower stem, leaves and grass (Fig. 1.32).

11. ***Finishing the drawing.*** Use the instructions in Steps 9–10 to change the drawing color to blue (Fig. 1.33(a)) and select a narrower line (Fig. 1.33(b)). Switch back to the drawing area and draw the raindrops (Fig. 1.34).

a) Selecting dark green as the drawing color.



b) Selecting a thicker line.

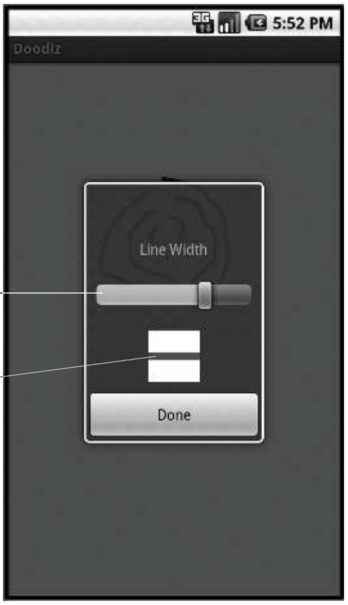

5:52 PM
Doodlz
Line Width
Done
`SeekBar` to control the line width
Displays the line in the current drawing color (green) and line width

Fig. 1.31 | Changing the line color and line width.

Drawing the stem, leaves and grass.

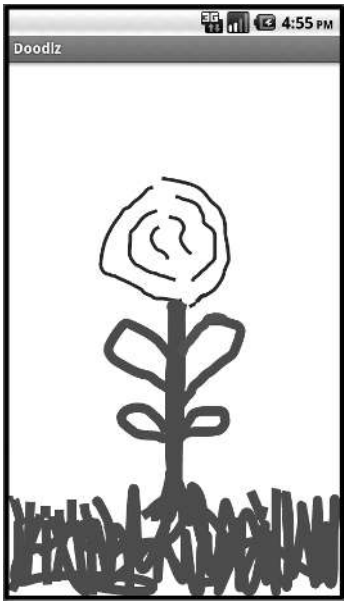


Fig. 1.32 | Drawing the stem and grass in the new line color and line width.

a) Selecting blue as the drawing color.

b) Selecting a thinner line.

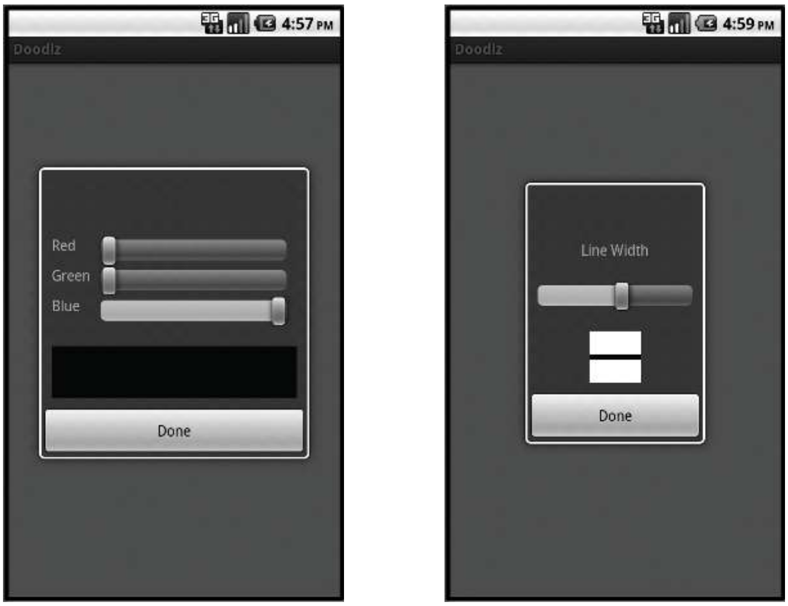


Fig. 1.33 | Changing the line color and width.

c)

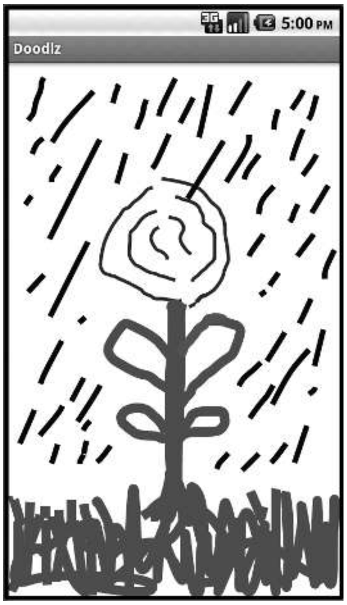


Fig. 1.34 | Drawing the rain in the new line color and line width.

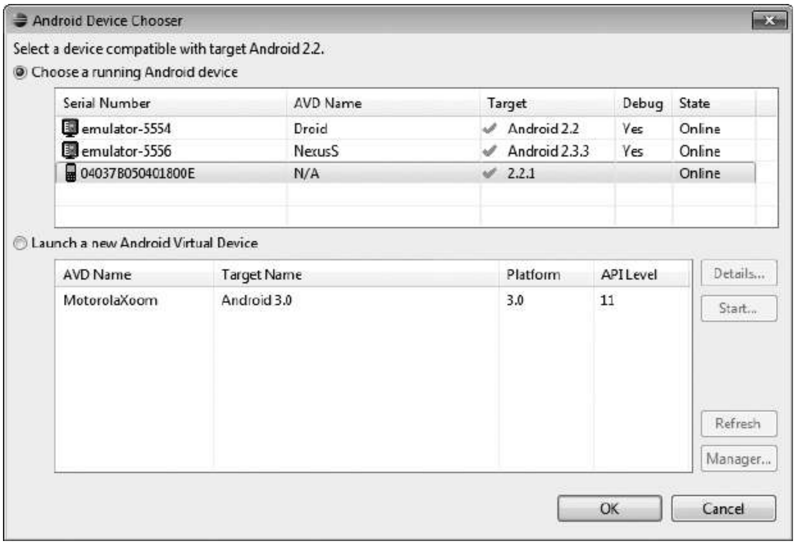
12. ***Saving the image.*** If you'd like, you can save the image to the **Gallery** by touching the **Menu** (MENU) button, then touching **Save Image**. You can then view this image and others stored on the device by opening the **Gallery** app.
13. ***Returning to the home screen.*** You can return to the AVD's home screen by clicking the home ( ) button on the AVD.

*Running the **Doodlz** App on an Android Device*

If you have an Android device, you can easily execute an app on the device for testing purposes.

1. First, you must enable debugging on the device. To do so, go to the device's **Settings** app, then select **Applications > Development** and ensure that **USB debugging** is checked.
2. Next, connect the device to your computer via a USB cable—typically this comes with the device when you purchase it.
3. In Eclipse, select the **Doodlz** project in the **Package Explorer** window, then select **Run As > Android Application** from the **Run As** button ( ) drop-down menu on the IDE's toolbar (Fig. 1.26).

If you do not have any AVDs open, but do have an Android device connected, the IDE will automatically install the app on your device and execute it. If you have one or more AVDs open and/or devices connected, the **Android Device Chooser** dialog (Fig. 1.35) is displayed so that you can select the AVD or device on which to install and execute the app. In this case, we first started two AVDs and connected one actual device, so there are three "devices" on which we could possibly run the app. We set up several AVDs so that we could simulate real Android devices with different versions of the Android OS and different screen sizes.


Android Device Chooser

Select a device compatible with target Android 2.2.

● Choose a running Android device

Serial Number	AVD Name	Target	Debug	State
emulator-5554	Droid	✓ Android 2.2	Yes	Online
emulator-5556	NexusS	✓ Android 2.3.3	Yes	Online
04037B050401800E	N/A	✓ 2.2.1		Online

○ Launch a new Android Virtual Device

AVD Name	Target Name	Platform	API Level
MotorolaXoom	Android 3.0	3.0	11

Fig. 1.35 | **Android Device Chooser** dialog.

In the **Choose a running Android device** section of Fig. 1.35, the dialog shows that we have one actual device connected to the computer (represented by the third line in the device list) and three AVDs. Each AVD has an **AVD Name** that we chose (`Droid`, `NexusS` and `MotorolaXoom`). Select the device or AVD you wish to use, then click **OK** to install and execute the app on that device or AVD. If you have other AVDs that you've defined and they're not currently executing, you can use the bottom half of this dialog to select and launch one of those AVDs.

When you build apps for distribution via the Android Market, you should test the apps on as many actual devices as you can. Remember that some features can be tested *only* on real devices. If you don't have many actual devices available to you, consider creating AVDs that simulate the various devices on which you'd like your app to execute. When you configure each AVD to simulate a specific real device, look up the real device's specifications online and configure the AVD accordingly. In addition, you can modify the AVD's `config.ini` file as described in the section **Setting hardware emulation options** at

```
    developer.android.com/guide/developing/tools/avd.html
```

This file contains options that are not configurable via the ADT Plugin in Eclipse. Modifying these options allows you to more precisely match the hardware configuration of a real device.

1.12 Deitel Resources

Our website (`www.deitel.com`) provides more than 100 Resource Centers on various topics including programming languages, software development, Web 2.0, Internet business and open-source projects. The Resource Centers evolve out of the research we do to support our publications and business endeavors. We've found many exceptional resources online, including tutorials, documentation, software downloads, articles, blogs, podcasts, videos, code samples, books, e-books and more—most of them are free. We announce our latest Resource Centers in our newsletter, the *Deitel® Buzz Online*, and on Facebook and Twitter. Figure 1.36 provides a list of the Deitel resources to help you get started with Android app development.

Deitel Android resources	URL
Android for Programmers: An App-Driven Approach book page	`www.deitel.com/books/AndroidFP/`
Android Resource Center	`www.deitel.com/android/`
Android Best Practices Resource Center	`www.deitel.com/androidbestpractices/`
Java Resource Center	`www.deitel.com/java/`
Eclipse Resource Center	`www.deitel.com/Eclipse/`
SQLite 3 Resource Center	`www.deitel.com/SQLite3/`

Fig. 1.36 | Deitel Android resources. (Part 1 of 2.)

Deitel Android resources	URL
Deitel Resource Centers homepage	www.deitel.com/ResourceCenters.html
Deitel on Facebook	www.deitel.com/DeitelFan/
Deitel on Twitter	@deitel
Deitel Buzz Online e-mail newsletter	www.deitel.com/newsletter/subscribe.html

Fig. 1.36 | Deitel Android resources. (Part 2 of 2.)

1.13 Android Development Resources

Figure 1.37 is a list of Android development resources. Figure 1.38 lists several of the Android developer videos available on `developer.android.com`. For additional resources, visit our Android Resource Center at `www.deitel.com/android`.

Android development tips and resources	URL
Android Developers' Channel on YouTube	www.youtube.com/user/androiddevelopers
Sample Android apps from Google	code.google.com/p/apps-for-android/
O'Reilly article, "Ten Tips for Android Application Development"	answers.oreilly.com/topic/862-ten-tips-for-android-application-development/
Bright Hub™ website for Android programming tips and how-to guides	www.brighthub.com/mobile/google-android.aspx
The article, "10 User Experience Tips for Successful Android Apps"	www.androidtapp.com/10-user-experience-tips-for-successful-android-apps/
Rapid Android development tips	www.droidnova.com/
The tutorial, "Working with XML on Android: Build Java applications for mobile devices," by Michael Galpin, software architect at eBay	www.ibm.com/developerworks/opensource/library/x-android/index.html
The Android Developers blog	android-developers.blogspot.com/
The Sprint Application Developers Program	developer.sprint.com/site/global/develop/mobile_platforms/android/android.jsp
The T-Mobile Android developer website	developer.t-mobile.com/site/global/resources/partner_hubs/android/p_android.jsp
HTC's Developer Center for Android and Windows Mobile development	developer.htc.com/
The Motorola Android development site	developer.motorola.com/

Fig. 1.37 | Android development tips and resources.

Video	URL
Androidology, Part 1 of 3: Architecture Overview	`developer.android.com/videos/index.html#v=QBGfUs9mQYY`
Androidology, Part 2 of 3: Application Lifecycle	`developer.android.com/videos/index.html#v=fL6gSd4ugSI`
Androidology, Part 3 of 3: APIs	`developer.android.com/videos/index.html#v=MPukbH6D-lY`
Android Developer Soapbox: Easy for Java Developers, Build Desktop Widgets	`developer.android.com/videos/index.html#v=FTAxE6SIWeI`
A Beginner's Guide to Android	`developer.android.com/videos/index.html#v=yqCj83leYRE`
The World of List View	`developer.android.com/videos/index.html#v=wDBM6wVEO70`
Android UI Design Patterns	`developer.android.com/videos/index.html#v=M1ZBjlCRfz0`
Writing Zippy Android Apps	`developer.android.com/videos/index.html#v=c4znvD-7VDA`
Casting a Wide Net for All Android Devices	`developer.android.com/videos/index.html#v=zNmohaZYvPw`
Building Push Applications for Android	`developer.android.com/videos/index.html#v=PLM4LajwDVc`

Fig. 1.38 | Android developer videos.

1.14 Wrap-Up

This chapter presented a brief history of Android and discussed its functionality. We discussed features of the Android 2.2, 2.3 and 3.0 operating system. We provided links to some of the key online documentation and to the newgroups and forums you can use to connect with the developer community. We discussed Android Market and provided links to some popular app review and recommendation sites. You learned the Android gestures and how to perform each on an Android device and on the emulator. We introduced the Java, Android and Google packages that enable you to use the hardware and software functionality you'll need to build your Android apps. You'll use many of these packages in this book. We also discussed Java programming and the Android SDK. We provided a quick refresher on basic object-technology concepts, including classes, objects, attributes and behaviors. You test-drove the **Doodlz** app on the Android emulator.

In Chapter 2, we discuss the business side of Android app development. You'll see how to prepare your apps for submission to the Android Market. We provide tips for pricing and marketing your app. We also show how to use Android Market capabilities for tracking app sales, payments and more.