

Android 3 platform SDK techniques for
developing smartphone and tablet apps



Pro Android 3

Satya Komatineni | Dave MacLean | Sayed Hashimi

Apress®

Pro Android 3



Satya Komatineni
Dave MacLean
Sayed Y. Hashimi

Apress®

Pro Android 3

Copyright © 2011 by Satya Komatineni, Dave MacLean, and Sayed Y. Hashimi

All rights reserved. No part of this work may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording, or by any information storage or retrieval system, without the prior written permission of the copyright owner and the publisher.

ISBN-13 (pbk): 978-1-4302-3222-3

ISBN-13 (electronic): 978-1-4302-3223-0

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark. NFC Forum and the NFC Forum logo are trademarks of the Near Field Communication Forum.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

President and Publisher: Paul Manning

Lead Editor: Matthew Moodie

Technical Reviewer: Dylan Phillips

Editorial Board: Steve Anglin, Mark Beckner, Ewan Buckingham, Gary Cornell, Jonathan Gennick, Jonathan Hassell, Michelle Lowman, Matthew Moodie, Jeffrey Pepper, Frank Pohlmann, Douglas Pundick, Ben Renow-Clarke, Dominic Shakeshaft, Matt Wade, Tom Welsh

Coordinating Editor: Corbin Collins

Copy Editors: Heather Lang, Tracy Brown, Mary Behr

Compositor: MacPS, LLC

Indexer: BIM Indexing & Proofreading Services

Artist: April Milne

Cover Designer: Anna Ishchenko

Distributed to the book trade worldwide by Springer Science+Business Media, LLC., 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/info/bulksales.

The information in this book is distributed on an “as is” basis, without warranty. Although every precaution has been taken in the preparation of this work, neither the author(s) nor Apress shall have any liability to any person or entity with respect to any loss or damage caused or alleged to be caused directly or indirectly by the information contained in this work.

The source code for this book is available to readers at www.apress.com.

To my brother Hari, to whom life yielded few favors.

—Satya Komatineni

To my wife, Rosie, and my son, Mike, for their support; I couldn't have done this without you. And to Max, for spending so much time at my feet keeping me company.

—Dave MacLean

To my son, Sayed-Adieb.

—Sayed Y. Hashimi

Contents at a Glance

Contents	vi
Foreword	xviii
About the Authors	xix
About the Technical Reviewer	xx
Acknowledgments	xxi
Preface	xxii
■ Chapter 1: Introducing the Android Computing Platform	1
■ Chapter 2: Setting Up Your Development Environment	21
■ Chapter 3: Understanding Android Resources	63
■ Chapter 4: Understanding Content Providers	89
■ Chapter 5: Understanding Intents	125
■ Chapter 6: Building User Interfaces and Using Controls	145
■ Chapter 7: Working with Menus	217
■ Chapter 8: Working with Dialogs	243
■ Chapter 9: Working with Preferences and Saving State	265
■ Chapter 10: Exploring Security and Permissions	287
■ Chapter 11: Building and Consuming Services	307
■ Chapter 12: Exploring Packages	377
■ Chapter 13: Exploring Handlers	399
■ Chapter 14: Broadcast Receivers and Long-Running Services	425
■ Chapter 15: Exploring the Alarm Manager	465
■ Chapter 16: Exploring 2D Animation	491
■ Chapter 17: Exploring Maps and Location-based Services	519
■ Chapter 18: Using the Telephony APIs	559
■ Chapter 19: Understanding the Media Frameworks	575
■ Chapter 20: Programming 3D Graphics with OpenGL	623
■ Chapter 21: Exploring Live Folders	693
■ Chapter 22: Home Screen Widgets	711
■ Chapter 23: Android Search	745
■ Chapter 24: Exploring Text to Speech	825
■ Chapter 25: Touch Screens	845
■ Chapter 26: Using Sensors	891
■ Chapter 27: Exploring the Contacts API	937
■ Chapter 28: Deploying Your Application: Android Market and Beyond	993
■ Chapter 29: Fragments for Tablets and More	1015
■ Chapter 30: Exploring ActionBar	1069
■ Chapter 31: Additional Topics in 3.0	1097
Index	1141

Contents

Contents at a Glance	iv
Foreword	xviii
About the Authors	xix
About the Technical Reviewer	xx
Acknowledgments	xxi
Preface	xxii
■ Chapter 1: Introducing the Android Computing Platform	1
A New Platform for a New Personal Computer	1
Early History of Android	3
Delving Into the Dalvik VM	6
Understanding the Android Software Stack.....	6
Developing an End-User Application with the Android SDK.....	8
Android Emulator	8
The Android UI	9
The Android Foundational Components	10
Advanced UI Concepts	11
Android Service Components.....	13
Android Media and Telephony Components	13
Android Java Packages.....	14
Taking Advantage of Android Source Code.....	18
The Sample Projects in this Book	19
Summary	20
■ Chapter 2: Setting Up Your Development Environment	21
Setting Up Your Environment.....	22
Downloading JDK 6.....	22
Downloading Eclipse 3.6.....	23
Downloading the Android SDK.....	23
The Tools Window.....	26
Installing Android Development Tools (ADT).....	26

Learning the Fundamental Components	29
View	29
Activity	29
Intent	29
Content Provider	30
Service	30
AndroidManifest.xml	30
Android Virtual Devices	30
Hello World!	31
Android Virtual Devices	37
Exploring the Structure of an Android Application	39
Analyzing the Notepad Application	42
Loading and Running the Notepad Application	42
Dissecting the Application	44
Examining the Application Lifecycle	51
Debugging Your App	54
Launching the Emulator	56
StrictMode	57
References	61
Summary	62
Chapter 3: Understanding Android Resources	63
Understanding Resources	63
String Resources	64
Layout Resources	66
Resource Reference Syntax	67
Defining Your Own Resource IDs for Later Use	69
Compiled and Uncompiled Android Resources	70
Enumerating Key Android Resources	71
Working with Arbitrary XML Resource Files	80
Working with Raw Resources	82
Working with Assets	82
Reviewing the Resources Directory Structure	83
Resources and Configuration Changes	83
Reference URLs	87
Summary	88
Chapter 4: Understanding Content Providers	89
Exploring Android's Built-in Providers	90
Architecture of Content Providers	96
Implementing Content Providers	108
Exercising the Book Provider	120
Adding A Book	120
Removing a Book	120
Getting a Count of the Books	121
Displaying the List of Books	121
Resources	122
Summary	123
Chapter 5: Understanding Intents	125
Basics of Android Intents	125
Available Intents in Android	127
Exploring Intent Composition	129
Intents and Data URLs	129

Generic Actions.....	130
Using Extra Information	131
Using Components to Directly Invoke an Activity	133
Understanding Intent Categories	134
Rules for Resolving Intents to Their Components	137
Exercising the ACTION_PICK.....	139
Exercising the GET_CONTENT Action.....	141
Introducing Pending Intents.....	142
Resources	144
Summary	144
■ Chapter 6: Building User Interfaces and Using Controls	145
UI Development in Android	145
Building a UI Completely in Code	147
Building a UI Completely in XML	149
Building a UI in XML With Code.....	150
Understanding Android's Common Controls	152
Text Controls.....	152
Button Controls	157
The ImageView Control.....	165
Date and Time Controls	167
The MapView Control.....	169
Understanding Adapters	170
Getting to Know SimpleCursorAdapter	171
Getting to Know ArrayAdapter	172
Using Adapters With AdapterViews	174
The Basic List Control: ListView.....	175
The GridView Control	183
The Spinner Control	185
The Gallery Control.....	187
Creating Custom Adapters	188
Other Controls in Android.....	194
Styles and Themes.....	194
Using Styles	194
Using Themes	197
Understanding Layout Managers	198
The LinearLayout Layout Manager.....	199
The TableLayout Layout Manager	202
The RelativeLayout Layout Manager	206
The FrameLayout Layout Manager	208
Customizing Layout for Various Device Configurations	210
Debugging and Optimizing Layouts with the Hierarchy Viewer	213
References.....	216
Summary	216
■ Chapter 7: Working with Menus	217
Understanding Android Menus	217
Creating a Menu.....	219
Working with Menu Groups	220
Responding to Menu Items	221
Creating a Test Harness for Testing Menus.....	222
Working with Other Menu Types.....	229
Expanded Menus	229
Working with Icon Menus	229

Working with Submenus	230
Provisioning for System Menus	231
Working with Context Menus	231
Working with Alternative Menus	234
Working with Menus in Response to Changing Data	238
Loading Menus Through XML Files	238
Structure of an XML Menu Resource File	239
Inflating XML Menu Resource Files	239
Responding to XML-Based Menu Items	240
A Brief Introduction to Additional XML Menu Tags	241
Resource	242
Summary	242
Chapter 8: Working with Dialogs	243
Using Dialogs in Android	243
Designing an Alert Dialog	244
Designing a Prompt Dialog	246
Nature of Dialogs in Android	251
Rearchitecting the Prompt Dialog	252
Working with Managed Dialogs	253
Understanding the Managed-Dialog Protocol	253
Recasting the Nonmanaged Dialog as a Managed Dialog	253
Simplifying the Managed-Dialog Protocol	255
Working with Toast	263
Resources	264
Summary	264
Chapter 9: Working with Preferences and Saving State	265
Exploring the Preferences Framework	265
Understanding ListPreference	266
Understanding CheckBoxPreference	275
Understanding EditTextPreference	277
Understanding RingtonePreference	278
Organizing Preferences	280
Manipulating Preferences Programmatically	283
Saving State with Preferences	284
Reference	285
Summary	286
Chapter 10: Exploring Security and Permissions	287
Understanding the Android Security Model	287
Overview of Security Concepts	287
Signing Applications for Deployment	288
Performing Runtime Security Checks	295
Understanding Security at the Process Boundary	295
Declaring and Using Permissions	295
Understanding and Using Custom Permissions	297
Understanding and Using URI Permissions	303
References	305
Summary	305
Chapter 11: Building and Consuming Services	307
Consuming HTTP Services	307
Using the HttpClient for HTTP GET Requests	308
Using the HttpClient for HTTP POST Requests (a Multipart Example)	310

SOAP, JSON, and XML Parsers	312
Dealing with Exceptions	313
Addressing Multithreading Issues	315
Fun With Timeouts	318
Using the HttpURLConnection	319
Using the AndroidHttpClient	319
Using Background Threads (AsyncTask)	320
Handling Configuration Changes with AsyncTasks	327
Getting Files Using DownloadManager	331
Using Android Services	337
Understanding Services in Android	338
Understanding Local Services	339
Understanding AIDL Services	346
Defining a Service Interface in AIDL	347
Implementing an AIDL Interface	349
Calling the Service from a Client Application	351
Passing Complex Types to Services	355
Real-World Example Using Services	366
Google Translate API	366
Using the Google Translate API	367
References	375
Summary	376
■ Chapter 12: Exploring Packages	377
Packages and Processes	377
Details of a Package Specification	377
Translating Package Name to a Process Name	378
Listing Installed Packages	378
Deleting a Package through the Package Browser	379
Revisiting the Package Signing Process	379
Understanding Digital Signatures: Scenario 1	380
Understanding Digital Signatures: Scenario 2	380
A Pattern for Understanding Digital Signatures	380
So How Do You Digitally Sign?	381
Implications of the Signing Process	381
Sharing Data Among Packages	382
The Nature of Shared User IDs	382
A Code Pattern for Sharing Data	383
Library Projects	384
What Is a Library Project?	384
Library Project Predicates	385
Creating a Library Project	387
Creating an Android Project That Uses a library	390
References	397
Summary	398
■ Chapter 13: Exploring Handlers	399
Android Components and Threading	399
Activities Run on the Main Thread	400
Broadcast Receivers run on the Main Thread	401
Services Run on the Main Thread	401
Content Provider Runs on the Main Thread	401
Implications of a Singular Main Thread	401
Thread Pools, Content Providers, External Service Components	401

Thread Utilities: Discover Your Threads.....	401
Handlers.....	403
Implications of Holding the Main Thread	404
Using a Handler to Defer Work on the Main Thread.....	405
A Sample Handler Source Code That Defers Work	405
Constructing a Suitable Message Object.....	407
Sending Message Objects to the Queue	407
Responding to the handleMessage Callback	408
Using Worker Threads.....	408
Invoking a Worker Thread from a Menu	409
Communicating Between the Worker and the Main Threads	410
A Quick Overview of Thread Behavior.....	412
Handler Example Driver classes	413
Driver Activity File.....	414
Layout File	417
Menu File	417
Manifest File	417
Component and Process Lifetimes	418
Activity Life Cycle.....	418
Service Life Cycle	420
Receiver Life Cycle	420
Provider Life Cycle	421
Instructions for Compiling the Code.....	421
Creating the Project from the ZIP File.....	421
Creating the Project from the Listings	422
References.....	422
Summary	423
Chapter 14: Broadcast Receivers and Long-Running Services	425
Broadcast Receivers	425
Sending a Broadcast.....	426
Coding a Simple Receiver: Sample Code	426
Registering a Receiver in the Manifest File	427
Sending a Test Broadcast.....	428
Accommodating Multiple Receivers	431
A Project for Out-of-Process Receivers	433
Using Notifications from a Receiver.....	434
Monitoring Notifications Through the Notification Manager	435
Sending a Notification.....	437
Long-Running Receivers and Services	440
Long-Running Broadcast Receiver Protocol	441
IntentService.....	442
IntentService Source Code.....	443
Extending IntentService for a Broadcast Receiver	445
Long-Running Broadcast Service Abstraction	445
A Long-Running Receiver	447
Abstracting a Wake Lock with LightedGreenRoom	449
Long-Running Service Implementation.....	455
Details of a Nonsticky Service	456
Details of a Sticky Service	457
A Variation of Nonsticky: Redeliver Intents.....	457
Specifying Service Flags in onStartCommand	457
Picking Suitable Stickiness.....	457

Controlling the Wake Lock from Two Places	458
Long-Running Service Implementation.....	458
Testing Long Running Services	460
Instructions for Compiling the Code.....	461
Creating the Projects from the ZIP File	461
Creating the Project from the Listings	461
References.....	464
Summary	464
■ Chapter 15: Exploring the Alarm Manager	465
Alarm Manager Basics: Setting Up a Simple Alarm	465
Obtaining the Alarm Manager	466
Setting Up the Time for the Alarm	466
Setting Up a Receiver for the Alarm.....	467
Creating a PendingIntent Suitable for an Alarm.....	467
Setting the Alarm	468
Test Project.....	468
Exploring Alarm Manager Alternate Scenarios	476
Setting Off an Alarm Repeatedly.....	476
Cancelling an Alarm.....	479
Working with Multiple Alarms.....	480
Intent Primacy in Setting Off Alarms.....	484
Persistence of Alarms	487
Alarm Manager Predicates	487
References.....	488
Summary	489
■ Chapter 16: Exploring 2D Animation	491
Frame-by-Frame Animation.....	492
Planning for Frame-by-Frame Animation	492
Creating the Activity.....	493
Adding Animation to the Activity.....	494
Layout Animation	498
Basic Tweening Animation Types	498
Planning the Layout Animation Test Harness	499
Creating the Activity and the ListView	500
Animating the ListView	502
Using Interpolators.....	506
View Animation	507
Understanding View Animation.....	507
Adding Animation.....	511
Using Camera to Provide Depth Perception in 2D.....	514
Exploring the AnimationListener Class	515
Some Notes on Transformation Matrices	516
Resources	517
Summary	517
■ Chapter 17: Exploring Maps and Location-based Services	519
Understanding the Mapping Package	520
Obtaining a map-api Key from Google	520
Understanding MapView and MapActivity	522
Adding Markers Using Overlays.....	528
Understanding the Location Package	533
Geocoding with Android.....	534

Geocoding with Background Threads	538
Understanding the LocationManager Service	541
Showing Your Location Using MyLocationOverlay	549
Using Proximity Alerts	554
References	558
Summary	558
■ Chapter 18: Using the Telephony APIs	559
Working with SMS	559
Sending SMS Messages	559
Monitoring Incoming SMS Messages	563
Working with SMS Folders	565
Sending E-mail	567
Working with the Telephony Manager	568
Session Initiation Protocol (SIP)	571
References	574
Summary	574
■ Chapter 19: Understanding the Media Frameworks	575
Using the Media APIs	575
Using SD Cards	576
Playing Media	581
Playing Audio Content	581
Playing Video Content	593
Recording Media	595
Exploring Audio Recording with MediaRecorder	596
Recording Audio with AudioRecord	600
Exploring Video Recording	605
Exploring the MediaStore Class	614
Recording Audio Using an Intent	615
Adding Media Content to the Media Store	618
Triggering MediaScanner for the Entire SD Card	621
References	621
Summary	621
■ Chapter 20: Programming 3D Graphics with OpenGL	623
Understanding the History and Background of OpenGL	624
OpenGL ES	625
OpenGL ES and Java ME	626
M3G: Another Java ME 3D Graphics Standard	626
Fundamentals of OpenGL	627
Essential Drawing with OpenGL ES	628
Understanding OpenGL Camera and Coordinates	633
Interfacing OpenGL ES with Android	637
Using GLSurfaceView and Related Classes	638
Implementing the Renderer	638
Using GLSurfaceView from an Activity	641
Changing Camera Settings	647
Using Indices to Add Another Triangle	649
Animating the Simple OpenGL Triangle	651
Braving OpenGL: Shapes and Textures	653
Drawing a Rectangle	653
Working with Shapes	656
Working with Textures	668

Drawing Multiple Figures.....	674
OpenGL ES 2.0	678
Java Bindings for OpenGL ES 2.0.....	678
Rendering Steps	682
Understanding Shaders	682
Compiling Shaders into a Program	684
Getting Access to the Shader Program Variables	685
A Simple ES 2.0 Triangle.....	685
Further Reading on OpenGL ES 2.0.....	689
Instructions for Compiling the Code.....	689
References.....	690
Summary	691
■ Chapter 21: Exploring Live Folders.....	693
Exploring Live Folders.....	693
How a User Experiences Live Folders.....	694
Building a Live Folder	700
Instructions for Compiling the Code.....	709
References.....	710
Summary	710
■ Chapter 22: Home Screen Widgets	711
Architecture of Home Screen Widgets.....	712
What Are Home Screen Widgets?	712
User Experience with Home Screen Widgets.....	713
Life Cycle of a Widget	716
A Sample Widget Application.....	722
Defining the Widget Provider	724
Defining Widget Size.....	725
Widget Layout-Related Files	726
Implementing a Widget Provider.....	728
Implementing Widget Models	730
Implementing Widget Configuration Activity.....	738
Widget Limitations and Extensions	742
Resources	742
Summary	743
■ Chapter 23: Android Search	745
Android Search Experience.....	746
Exploring Android Global Search	746
Enabling Suggestion Providers for Global Search.....	753
Activities and Search Key Interaction	757
Behavior of Search Key on a Regular Activity.....	758
Behavior of an Activity that Disables Search.....	766
Explicitly Invoking Search Through a Menu.....	767
Understanding Local Search and Related Activities	771
Enabling Type-to-Search	777
Implementing a Simple Suggestion Provider.....	778
Planning the Simple Suggestions Provider	779
Simple Suggestions Provider Implementation Files	779
Implementing the SimpleSuggestionProvider class	780
Understanding Simple Suggestions Provider Search Activity.....	784
Search Invoker Activity	789
Simple Suggestion Provider User Experience.....	791

Implementing a Custom Suggestion Provider.....	796
Planning the Custom Suggestion Provider.....	796
SuggestURLProvider Project Implementation Files.....	796
Implementing the SuggestUrlProvider Class	797
Implementing a Search Activity for a Custom Suggestion Provider	807
Custom Suggestions Provider Manifest File	813
Custom Suggestion User Experience	814
Using Action Keys and Application-Specific Search Data.....	818
Using Action Keys in Android Search.....	818
Working with Application-Specific Search Context	821
Resources	822
Implications for Tablets	823
Summary	823
■ Chapter 24: Exploring Text to Speech	825
The Basics of Text-to-Speech Capabilities in Android.....	825
Using Utterances to Keep Track of Our Speech	830
Using Audio Files for Your Voice	832
Advanced Features of the TTS Engine	838
Setting Audio Streams	839
Using Earcons	839
Playing Silence	840
Choosing a Different Text-to-Speech Engine.....	840
Using Language Methods	840
References.....	842
Summary	843
■ Chapter 25: Touch Screens.....	845
Understanding MotionEvent.....	845
The MotionEvent Object.....	845
Recycling MotionEvent.....	857
Using VelocityTracker	857
Exploring Drag and Drop.....	859
Multitouch.....	862
Multitouch Before Android 2.2.....	863
Multitouch Since Android 2.2.....	871
Touches with Maps.....	871
Gestures.....	874
The Pinch Gesture.....	875
GestureDetector and OnGestureListeners.....	878
Custom Gestures.....	881
The Gestures Builder Application.....	882
References.....	889
Summary	889
■ Chapter 26: Using Sensors	891
What Is a Sensor?.....	891
Detecting Sensors.....	892
What Can We Know About a Sensor?	892
Getting Sensor Events.....	895
Issues with Getting Sensor Data.....	898
Interpreting Sensor Data.....	905
Light Sensors	905
Proximity Sensors.....	906

Temperature Sensors	907
Pressure Sensors	907
Gyroscope Sensors	907
Accelerometers	908
Magnetic Field Sensors	914
Using Accelerometers and Magnetic Field Sensors Together	915
Orientation Sensors	915
Magnetic Declination and GeomagneticField	922
Gravity Sensors	923
Linear Acceleration Sensors	923
Rotation Vector Sensors	923
Near Field Communication Sensors	923
References	934
Summary	935
Chapter 27: Exploring the Contacts API	937
Understanding Accounts	938
A Quick Tour of Account Screens	938
Relevance of Accounts to Contacts	942
Enumerating Accounts	943
Understanding Contacts Application	944
Show Contacts	944
Show Contact Detail	945
Edit Contact Details	946
Setting a Contact's Photo	948
Exporting Contacts	949
Various Contact Data Types	951
Understanding Contacts	952
Examining the Contents SQLite Database	952
Raw Contacts	953
Data Table	955
Aggregated Contacts	956
view_contacts	958
contact_entities_view	959
Working with the Contacts API	960
Exploring Accounts	960
Exploring Aggregated Contacts	968
Exploring Raw Contacts	977
Exploring Raw Contact Data	982
Adding a Contact and Its Details	985
Controlling Aggregation	988
Impacts of Syncing	989
References	990
Summary	991
Chapter 28: Deploying Your Application: Android Market and Beyond	993
Becoming a Publisher	994
Following the Rules	994
Developer Console	997
Preparing Your Application for Sale	1001
Testing for Different Devices	1001
Supporting Different Screen Sizes	1001
Preparing AndroidManifest.xml for Uploading	1002
Localizing Your Application	1003

Preparing Your Application Icon.....	1004
Considerations for Making Money From Apps	1004
Directing Users Back to the Market	1005
The Android Licensing Service	1006
Preparing Your .apk File for Uploading	1007
Uploading Your Application	1007
User Experience on Android Market	1010
Beyond Android Market	1012
References.....	1013
Summary	1013
■ Chapter 29: Fragments for Tablets and More.....	1015
What is a Fragment?.....	1015
When to Use Fragments.....	1016
The Structure of a Fragment.....	1017
A Fragment’s Lifecycle	1018
Sample Fragment App Showing the Lifecycle	1024
FragmentManagerTransactions and the Fragment Back Stack.....	1032
FragmentManagerTransaction Transitions and Animations	1034
The FragmentManager.....	1035
Caution When Referencing Fragments	1037
FragmentManagerListFragments and <fragment>	1037
Invoking a Separate Activity When Needed	1041
FragmentManagerPersistence of Fragments	1044
Understanding Dialog Fragments	1044
FragmentManagerDialogFragment Basics.....	1045
FragmentManagerDialogFragment Sample Application.....	1050
More Communications with Fragments.....	1063
FragmentManagerUsing startActivity() and setTargetFragment()	1064
Custom Animations with ObjectAnimator	1064
References.....	1067
Summary	1068
■ Chapter 30: Exploring ActionBar	1069
ActionBarAnatomy of an ActionBar	1070
ActionBarTabbed Navigation ActionBar Activity	1071
ActionBarImplementing Base Activity Classes	1073
ActionBarAssigning Uniform Behavior for the ActionBar.....	1075
ActionBarImplementing the Tabbed Listener	1077
ActionBarImplementing the Tabbed ActionBar Activity.....	1078
ActionBarScrollable Debug Text View Layout.....	1080
ActionBarActionBar and Menu Interaction	1081
ActionBarAndroid Manifest File.....	1083
ActionBarExamining the Tabbed ActionBar Activity	1084
ActionBarList Navigation ActionBar Activity	1084
ActionBarCreating a SpinnerAdapter	1085
ActionBarCreating a List Listener.....	1086
ActionBarSetting Up a List ActionBar	1086
ActionBarMaking Changes to BaseActionBarActivity	1087
ActionBarMaking Changes to AndroidManifest.xml	1087
ActionBarExamining the List ActionBar Activity	1088
ActionBarStandard Navigation ActionBar Activity	1090
ActionBarStandard Navigation ActionBar Activity	1090
ActionBarMaking Changes to BaseActionBarActivity.....	1091

Making Changes to AndroidManifest.xml	1092
Examining the Standard Action Bar activity.....	1092
References.....	1093
Summary	1094
■ Chapter 31: Additional Topics in 3.0	1097
List-Based Home Screen Widgets	1097
New Remote Views in 3.0.....	1098
Working with Lists in Remote Views.....	1099
Working Sample: Test Home Screen List Widget.....	1114
Testing the Test List Widget	1122
Drag and Drop.....	1124
Basics of Drag and Drop in 3.0	1124
Drag and Drop Sample Application	1125
Testing the Sample Drag-and-Drop Application	1137
References.....	1138
Summary	1139
Index.....	1141

Foreword

All this has happened before, and all this will happen again. Emergence Theory is the way complex systems and patterns arise out of a set of environmental interactions.

And we have been here before.

When I started programming in 1985, there were a variety of personal computers available. While I cut my teeth on an Apple II C, my friends either had Commodore 128s, Tandy CoCo 3s, or Atari computers. Each of us grew within the constraints of our own environment, but we were rarely able to share our work. When affordable IBM clones running Microsoft's DOS began to emerge, developers started to see value of the marketplace created, and rapid evolution began to occur within the DOS ecosystem. Eventually Microsoft created the dominate position in the PC market that it still enjoys today.

In 2003, when I started mobile programming, the ecosystem looked much the same as it did back in 1985. You could implement your vision in everything from Microsoft .NET CF to Java Micro Edition to BREW. But like the games I coded with my friends, our applications were isolated within our chosen ecosystem.

As 2011 dawns, by spreading the Android OS across hardware vendors, Google looks to be the Microsoft of the Mobile Space. That is likely why you have picked up this book and are reading this foreword. Either you are a student of history or, like me, you were lucky enough to live it.

Well, good news! We have worked very hard in this edition of the book to ensure you have the tools to implement the ideas rattling around in your imagination. We take you from the basics of setting up your environment through deploying to the marketplace. Of course this is a vast journey, so we mainly take you down the road most travelled. But we will provide you plenty of resources to explore on your own.

Good luck, and happy trails.

—Dylan Phillips

About the Authors



Satya Komatineni (www.satyakomatineni.com) has over 20 years of programming experience working with small and large corporations. Satya has published over 30 articles on web development using Java, .NET, and database technologies. He is a frequent speaker at industry conferences on innovative technologies and a regular contributor to the weblogs on Java.net. He is the author of AspireWeb (www.activeintellect.com/aspire), a simplified open source tool for Java web development, and the creator of Aspire Knowledge Central (www.knowledgefolders.com), an open source personal web operating system with a focus on individual productivity and publishing. Satya is also a contributing member to a number of Small Business Innovation Research Programs (SBIR). He received a bachelor's degree in electrical engineering from Andhra University, Visakhapatnam, and a master's degree in electrical engineering from the Indian Institute of Technology, New Delhi. You can contact him at satya.komatineni@gmail.com.



Dave MacLean is a software engineer and architect currently living and working in Jacksonville, Florida. Since 1980, he has programmed in many languages, developing solutions ranging from robot automation systems to data warehousing, from web self-service applications to EDI transaction processors. Dave has worked for Sun Microsystems, IBM, Trimble Navigation, General Motors, and several small companies. He graduated from the University of Waterloo in Canada with a degree in systems design engineering. Visit his blog at <http://davemac327.blogspot.com> or contact him at davemac327@gmail.com.



Sayed Y. Hashimi was born in Afghanistan and now resides in Jacksonville, Florida. His expertise spans the fields of health care, financials, logistics, and service-oriented architecture. In his professional career, Sayed has developed large-scale distributed applications with a variety of programming languages and platforms, including C/C++, MFC, J2EE, and .NET. He has published articles in major software journals and has written several other popular Apress titles. Sayed holds a master's degree in engineering from the University of Florida. You can reach him by visiting www.sayedhashimi.com.

Please visit the authors at their web site: www.androidbook.com.

About the Technical Reviewer



Dylan Phillips is a software engineer and architect who has been working in the mobile space for the last 10 years. With a broad range of experience ranging from J2ME to .NET Compact Framework to Android, he is incredibly excited about the opportunity presented by the broad consumer adoption of an array of Android devices. He can be reached at mykoan@hotmail.com, [@mykoan](https://twitter.com/mykoan) on Twitter, or at lunch, in various Pho Houses around the country.

Acknowledgments

Writing this book took effort not only on the part of the authors, but also from some of the very talented staff at Apress, as well as the technical reviewer. Therefore, we would like to thank Steve Anglin, Matthew Moodie, Corbin Collins, Heather Lang, Tracy Brown, Mary Behr, and Brigid Duffy from Apress. We would also like to extend our appreciation to the technical reviewer, Dylan Phillips, for the work he did on this book. His commentary and corrections were invaluable. When searching for answers on the android developers forum, we were often helped by Dianne Hackborn, Nick Pelly, Brad Fitzpatrick, and other members of the Android Team, at all hours of the day and weekends, and to them we would like to say, "Thank you." They truly are the hardest working team in mobile. The Android community is very much alive and well and was also very helpful in answering questions and offering advice. We hope this book in some way is able to give back to the community. Finally, the authors are deeply grateful to their families for accommodating prolonged irresponsibility.

Preface

Have you ever wanted to be a Rodin? Sitting with a chisel and eroding away a block of rock to mold it to your vision? Well, mainstream programmers have kept away from the severely constrained mobile devices for fear of being unable to chisel out at a workable application. Those times have passed.

Android OS places the incredible reach of a programmable device at your door step. In this book we want to positively confirm your suspicion that Android is a great OS to program with. If you are a Java programmer, you have a great opportunity to profit from this exciting, capable, general-purpose computing platform. We are excited about Android because it is an advanced platform that introduces a number of new paradigms in framework design (even with the limitations of a mobile platform).

This is our third edition on the subject of Android, and it's our best edition yet. *Pro Android 3* is an extensive programming guide. In this edition we've refined, rewritten, and enhanced everything from *Pro Android 2* to create a thoroughly updated guide for both beginners and professionals—the result of our three years of research. We cover over 100 topics in 31 chapters. This edition covers versions 2.3 and 3.0 of Android, the optimized versions of Android for phones and tablets, respectively.

In this edition we have beefed up Android internals by covering threads, processes, long running services, broadcast receivers, and alarm managers. We cover many more UI controls in this edition. We have over 150 pages of dedicated material on 3.0, covering fragments, fragment dialogs, ActionBar, and drag and drop. We have significantly enhanced the services and sensor chapters. OpenGL has been revised to include OpenGL ES 2.0.

Concepts, Code, and Tutorials are the essence of this book. Every chapter in the book reflects this philosophy. The self-contained tutorials in each chapter are annotated with expert advice. All projects in the book are available for download for easy importing into Eclipse. We have worked hard so that the code can also be compiled right out of the book. The list of files that goes into each project are explicitly catalogued and listed in each chapter for easy reference.

The areas we cover in the book include key concepts such as resources, intents, content providers, processes, threads, UI controls, broadcast receivers, services, and long running services. We have a lot of coverage on OpenGL ES 1.0 and 2.0 for OpenGL beginners. We have a lot of coverage on text to speech, sensors, and multi-touch. We are also able to incorporate a lot of coverage on 3.0 topics that include fragments, fragment dialogs, ActionBar, and drag and drop.

Finally, in this book we went beyond basics, asked tough questions on every topic, and documented the results (see the table of contents for the extensive list of what we cover in the book). We are also actively updating the supplemental website (www.androidbook.com) with current and future research material on the Android SDK. As you walk through the book, if you have any questions we are only an email away for a quick response.

Introducing the Android Computing Platform

Computing continues to become more and more personalized and accessible. Handheld devices have largely transformed into computing platforms. Mobile phones are no longer just for talking—they have been capable of carrying data and video for some time. Be it a phone or a tablet, the mobile device is now so capable of general-purpose computing that it's becoming more like a PC. A number of traditional PC manufacturers such as ASUS, HP, and Dell are producing devices of various form factors based on the Android OS. The battles between operating systems, computing platforms, programming languages, and development frameworks are being shifted and reapplied to mobile devices.

We are also seeing a surge in mobile programming as more and more IT applications start to offer mobile counterparts. In this book, we'll show you how to take advantage of your Java skills to write programs for devices that run on Google's Android platform (<http://developer.android.com/index.html>), an open-source platform for mobile and tablet development.

NOTE: We are excited about Android because it is an advanced platform that introduces a number of new paradigms in framework design (even with the limitations of a mobile platform).

In this chapter, we'll provide an overview of Android and its SDK, give a brief overview of key packages, introduce what we are going to cover in each chapter briefly, show you how to take advantage of Android source code, and highlight the benefits of programming for the Android platform.

A New Platform for a New Personal Computer

The fact that dedicated devices such as mobile phones can now count themselves among general-computing platforms is good news for programmers (see Figure 1-1).

Starting with Android 3.0, we can officially add tablets to this list. This trend makes programming for mobile devices possible with general-purpose computing languages, which increases the range and market share for mobile applications.

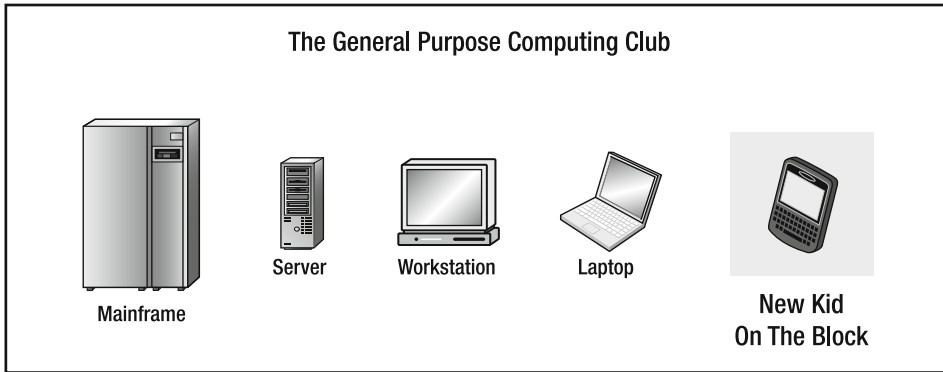


Figure 1–1. *Handheld is the new PC.*

The Android platform embraces this idea of general-purpose computing for handheld devices. It is a comprehensive platform that features a Linux-based operating system stack for managing devices, memory, and processes. Android’s Java libraries cover telephony, video, speech, graphics, connectivity, UI programming, and a number of other aspects of the device.

NOTE: Although built for mobile- and tablet-based devices, the Android platform exhibits the characteristics of a full-featured desktop framework. Google makes this framework available to Java programmers through a Software Development Kit (SDK) called the Android SDK. When you are working with the Android SDK, you rarely feel that you are writing to a mobile device because you have access to most of the class libraries that you use on a desktop or a server—including a relational database.

The Android SDK supports most of the Java Platform, Standard Edition (Java SE), except for the Abstract Window Toolkit (AWT) and Swing. In place of AWT and Swing, Android SDK has its own *extensive modern UI framework*. Because you’re programming your applications in Java, you could expect that you need a Java Virtual Machine (JVM) that is responsible for interpreting the runtime Java byte code. A JVM typically provides the necessary optimization to help Java reach performance levels comparable to compiled languages such as C and C++. Android offers its own optimized JVM to run the compiled Java class files in order to counter the handheld device limitations such as memory, processor speed, and power. This virtual machine is called the Dalvik VM, which we’ll explore in a later section “Delving into the Dalvik VM.”

NOTE: The familiarity and simplicity of the Java programming language, coupled with Android’s extensive class library, makes Android a compelling platform to write programs for.

Figure 1–2 provides an overview of the Android software stack. (We’ll provide further details in the section “Understanding the Android Software Stack.”)

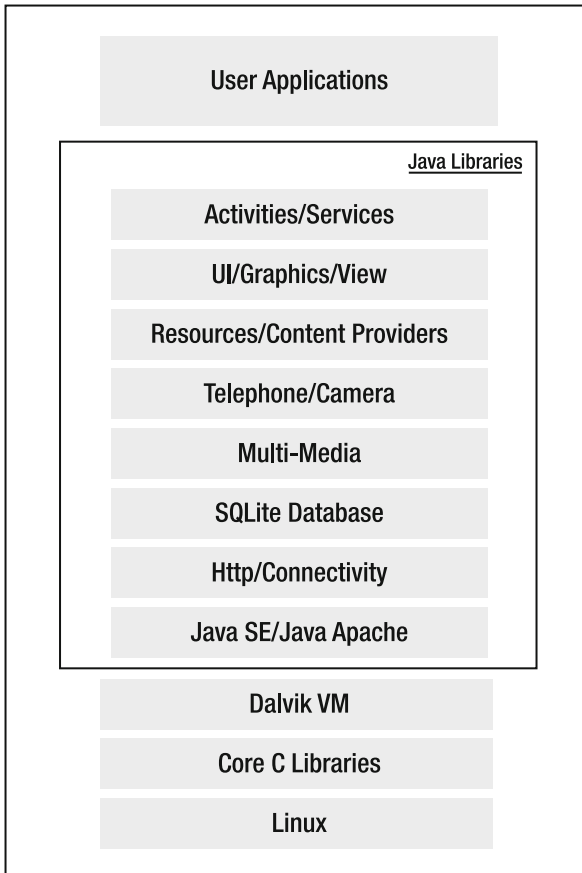


Figure 1–2. High-level view of the Android software stack

Early History of Android

Mobile phones use a variety of operating systems, such as Symbian OS, Microsoft’s Windows Mobile, Mobile Linux, iPhone OS (based on Mac OS X), Moblin (from Intel), and many other proprietary OSes. So far, no single OS has become the de facto standard. The available APIs and environments for developing mobile applications are too restrictive and seem to fall behind when compared to desktop frameworks. In contrast, the Android platform promised openness, affordability, open-source code, and, more important, a high-end, all-in-one-place, consistent development framework.

Google acquired the startup company Android Inc. in 2005 to start the development of the Android platform (see Figure 1–3). The key players at Android Inc. included Andy Rubin, Rich Miner, Nick Sears, and Chris White.

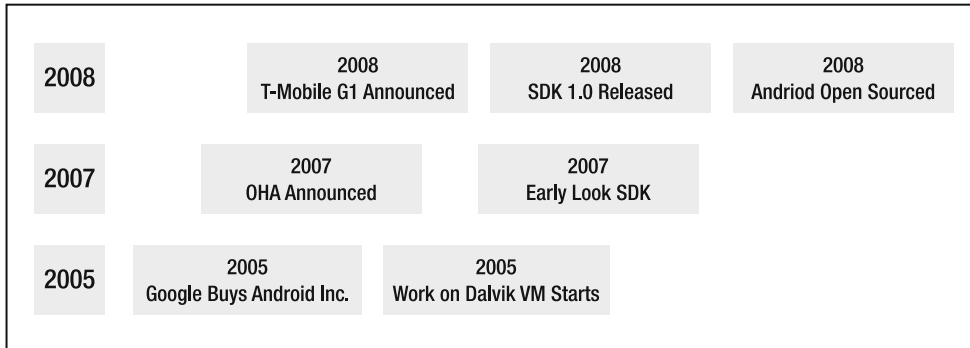


Figure 1–3. *Android early timeline*

In late 2007, a group of industry leaders came together around the Android platform to form the Open Handset Alliance (www.openhandsetalliance.com). Some of the alliance’s prominent members as of 2009 were as follows:

- Sprint Nextel
- T-Mobile
- Motorola
- Samsung
- Sony Ericsson
- Toshiba
- Vodafone
- Google
- Intel
- Texas Instruments

As of 2011, this list has grown by multifold (over 80 in number), as you can see at the Open Handset Alliance web site.

According to the site, part of the alliance’s goal is to innovate rapidly and respond better to consumer needs in the mobile space and its first key outcome was the Android platform. Android was designed to serve the needs of mobile operators, handset manufacturers, and application developers. The members have committed to release significant intellectual property through the open source Apache License, Version 2.0.

The Android SDK was first issued as an “early look” release in November 2007. In September 2008, T-Mobile announced the availability of T-Mobile G1, the first smartphone based on the Android platform. A few days after that, Google announced

the availability of Android SDK Release Candidate 1.0. In October 2008, Google made the source code of the Android platform available under Apache's open source license. In late 2010, Google released Android SDK 2.3 for smartphones, code named Gingerbread, which was upgraded to 2.3.3 by March 2011. In early 2011 an optimized version of Android for tablets, Android 3.0 code named Honeycomb, was released. Motorola XOOM is one of the early tablets to carry this OS release.

When Android was released, one of its key architectural goals was to allow applications to interact with one another and reuse components from one another. This reuse not only applies to services, but also to data and the user interface (UI). As a result, the Android platform has a number of architectural features that keep this openness a reality.

Android has attracted an early following and sustained the developer momentum because of its fully developed features to exploit the cloud-computing model offered by Web resources and to enhance that experience with local data stores on the handset itself. Android's support for a relational database on the handset also played a part in early adoption.

In releases 1.0 and 1.1 (2008) Android did not support soft keyboards, requiring the devices to carry physical keys. Android fixed this issue by releasing the 1.5 SDK in April 2009, along with a number of other features, such as advanced media-recording capabilities, widgets, and live folders.

In September 2009 came release 1.6 of the Android OS and, within a month, Android 2.0 followed, facilitating a flood of Android devices in time for the 2009 Christmas season. This release introduced advanced search capabilities and text to speech.

With support for HTML 5, Android 2.0 introduces interesting possibilities for using HTML. The contact API is significantly overhauled. Support for Flash is added. More and more Android-based applications are introduced every day, as well as new types of independent online application stores. Much anticipated tablet computers based on Android can now be purchased.

In Android 2.3 the significant features include remote wiping of secure data by administrators, the ability to use camera and video in low-light conditions, WiFi hotspot, significant performance improvements, improved Bluetooth functionality, installation of applications on the SD card optionally, OpenGL ES 2.0 support, improvements in backup, improvements in search usability, Near Field Communications support for credit card processing, much improved motion and sensor support (similar to Wii), video chat, and improved Market.

The latest incarnation of Android, 3.0 is focused on tablet-based devices and much more powerful dual core processors such as Nvidia Tegra2. The main features of this release include support to use larger screen. A significantly new concept called Fragments has been introduced. This permeates the 3.0 experience. More desktop-like capabilities, such as ActionBar and Drag and Drop, have been introduced. Home screen widgets have been significantly enhanced. More UI controls are now available. In the 3D space, OpenGL has been enhanced with Renderscript to further supplement ES 2.0. It is an exciting introduction for tablets.

Delving Into the Dalvik VM

As part of Android, Google has spent a lot of time thinking about optimizing designs for low-powered handheld devices. Handheld devices lag behind their desktop counterparts in memory and speed by eight to ten years. They also have limited power for computation. The performance requirements on handsets are severe as a result, requiring handset designers to optimize everything. If you look at the list of packages in Android, you'll see that they are fully featured and extensive.

These issues led Google to revisit the standard JVM implementation in many respects. The key figure in Google's implementation of this JVM is Dan Bornstein, who wrote the Dalvik VM—Dalvik is the name of a town in Iceland. Dalvik VM takes the generated Java class files and combines them into one or more Dalvik Executable (.dex) files. It reuses duplicate information from multiple class files, effectively reducing the space requirement (uncompressed) by half from a traditional .jar file

Google has also fine-tuned the garbage collection in the Dalvik VM, but it has chosen to omit a just-in-time (JIT) compiler, in early releases. Android 2.3 has added JIT. The reports are that this can give two to five times faster raw performance at places and 10 to 20% for general-purpose applications.

Dalvik VM uses a different kind of assembly-code generation, in which it uses registers as the primary units of data storage instead of the stack. Google is hoping to accomplish 30% fewer instructions as a result. We should point out that the final executable code in Android, as a result of the Dalvik VM, is based not on Java byte code but on .dex files instead. This means you cannot directly execute Java byte code; you have to start with Java class files and then convert them to linkable .dex files.

This performance paranoia extends into the rest of the Android SDK. For example, the Android SDK uses XML extensively to define UI layouts. However, all of this XML is compiled to binary files before these binary files become resident on the devices. Android provides special mechanisms to use this XML data.

Understanding the Android Software Stack

So far we've covered Android's history and its optimization features including the Dalvik VM, and we've hinted at the Java programming stack available. In this section, we will cover the development aspect of Android. Figure 1-4 is a good place to start this discussion.

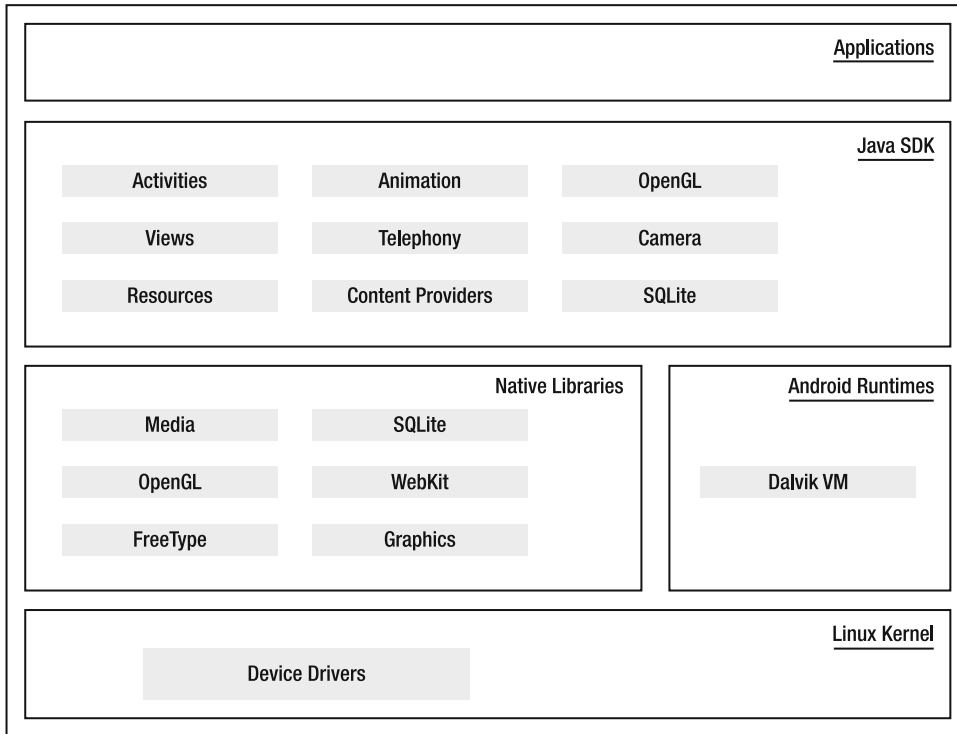


Figure 1–4. Detailed Android SDK software stack

At the core of the Android platform is a Linux kernel responsible for device drivers, resource access, power management, and other OS duties. The supplied device drivers include Display, Camera, Keypad, WiFi, Flash Memory, Audio, and IPC (inter-process communication). Although the core is Linux, the majority—if not all—of the applications on an Android device such as a Motorola Droid are developed in Java and run through the Dalvik VM.

Sitting at the next level, on top of the kernel, are a number of C/C++ libraries such as OpenGL, WebKit, FreeType, Secure Sockets Layer (SSL), the C runtime library (libc), SQLite, and Media. The system C library based on Berkeley Software Distribution (BSD) is tuned (to roughly half its original size) for embedded Linux-based devices. The media libraries are based on PacketVideo’s (www.packetvideo.com/) OpenCORE. These libraries are responsible for recording and playback of audio and video formats. A library called Surface Manager controls access to the display system and supports 2D and 3D. More of these native libraries are likely to be added with new releases.

The WebKit library is responsible for browser support; it is the same library that supports Google Chrome and Apple’s Safari. The FreeType library is responsible for font support. SQLite (www.sqlite.org/) is a relational database that is available on the device itself. SQLite is also an independent open-source effort for relational databases and not directly tied to Android. You can acquire and use tools meant for SQLite for Android databases as well.