



Android Application Development

Daniel Switkin

Senior Software Engineer, Google Inc.

Goal



- Get you an idea of how to start developing Android applications
- Introduce major Android application concepts
- Walk you through a sample application in the development environment

Agenda

- System architecture
- Hello World!
- Application components
- Practical matters
- Toolchain



What is Android?



- A free, open source mobile platform
- A Linux-based, multiprocess, multithreaded OS
- Android is not a device or a product
- It's not even limited to phones - you could build a DVR, a handheld GPS, an MP3 player, etc.

APPLICATIONS

Home

Contacts

Phone

Browser

...

APPLICATION FRAMEWORK

Activity Manager

Window Manager

Content Providers

View System

Notification Manager

Package Manager

Telephony Manager

Resource Manager

Location Manager

GTalk Service

LIBRARIES

Surface Manager

Media Framework

SQLite

OpenGL | ES

FreeType

WebKit

SGL

SSL

libc

ANDROID RUNTIME

Core Libraries

Dalvik Virtual Machine

LINUX KERNEL

Display Driver

Camera Driver

Bluetooth Driver

Flash Memory Driver

Binder (IPC) Driver

USB Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management

Hello World!



The History of GUIs



- Hardcoded to the screen
- Hardcoded to the window
- Hardcoded within a view hierarchy
- Dynamic layout within a view hierarchy

Generating GUIs



- Two ways to create GUIs: in XML or in code
 - Declarative route via XML has advantages
- A lot of your GUI-related work will take place in:
 - `res/layout`
 - `res/values`
- `@id/name_for_component` gives you handle for referencing XML declarations in code

Views



- Views are building blocks
- Examples:
 - Can be as basic as: `TextView`, `EditText`, or `ListView`
 - Fancier views: `ImageView`, `MapView`, `WebView`

Layouts



- Controls how Views are laid out
 - `FrameLayout` : each child a layer
 - `LinearLayout` : single row or column
 - `RelativeLayout` : relative to other Views
 - `TableLayout` : rows and columns
 - `AbsoluteLayout` : $\langle x,y \rangle$ coordinates

Layouts are resizable



480x320

Time	Sun 8/19	Mon 8/20	Tue 8/21	Wed 8/22	Thu 8/23	Fri 8/24	Sat 8/25
5 PM							
6 PM							
7 PM							
8 PM							
9 PM							
10 PM							
11 PM							

9 - 11 PM Android Tech Talk, Tunis (B43, 2nd Floor)

240x320

Time	Sun 8/19	Mon 8/20	Tue 8/21	Wed 8/22	Thu 8/23	Fri 8/24	Sat 8/25
5 PM							
6 PM							
7 PM							
8 PM							
9 PM							
10 PM							
11 PM							

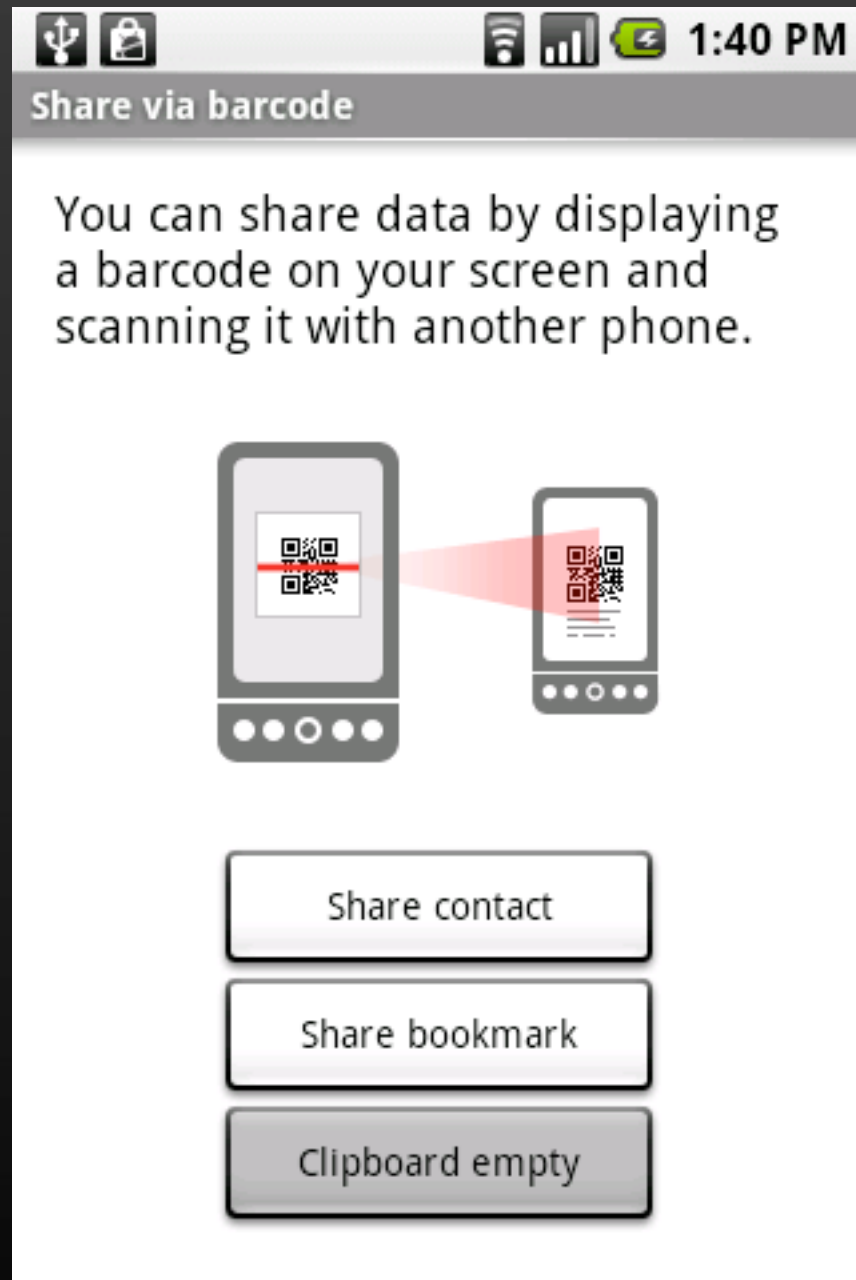
9 - 11 PM Android Tech Talk, Tunis (B43, 2nd Floor)

320x240

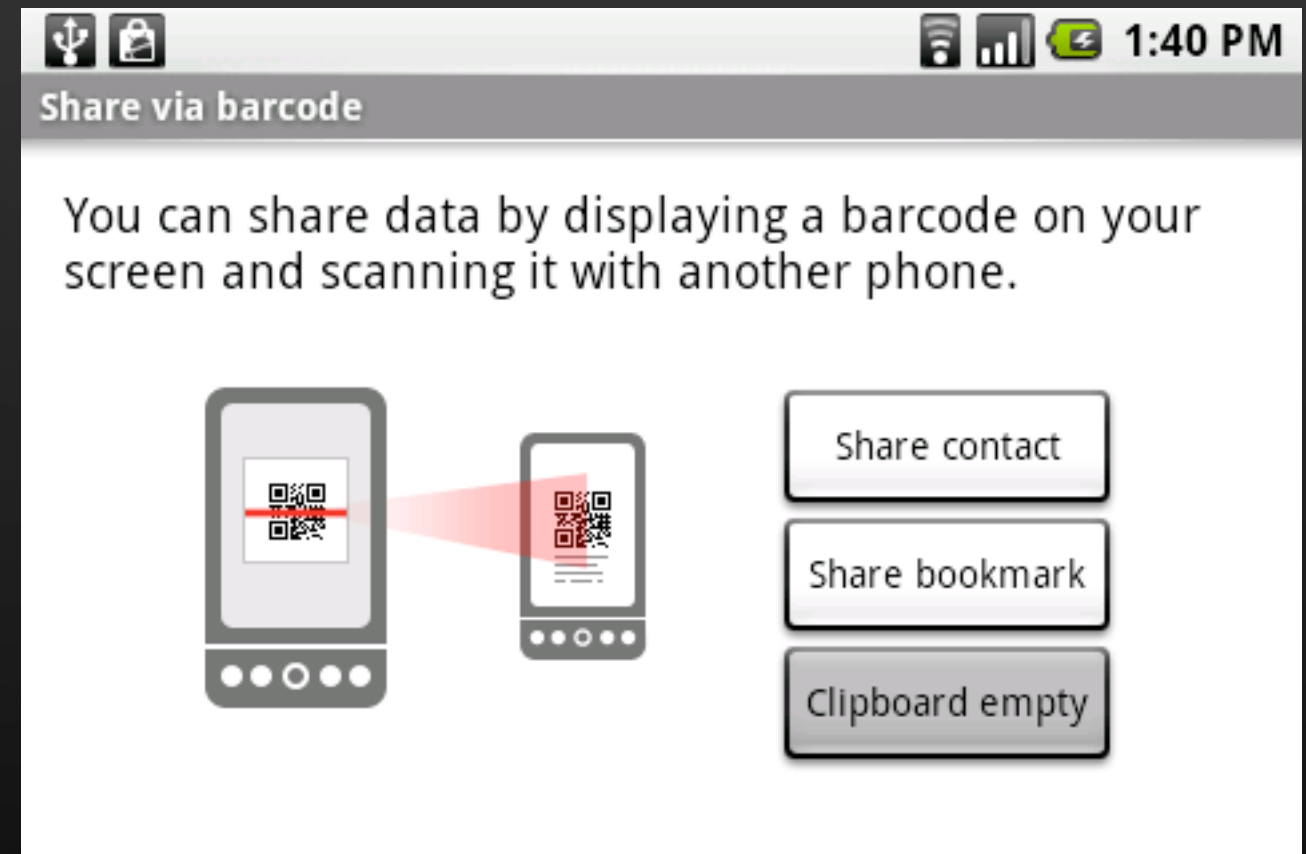
Time	Sun 8/19	Mon 8/20	Tue 8/21	Wed 8/22	Thu 8/23	Fri 8/24	Sat 8/25
5 PM							
6 PM							
7 PM							
8 PM							
9 PM							
10 PM							
11 PM							

9 - 11 PM Android Tech Talk, Tunis (B43, 2nd Floor)

Layouts are customizable



`res/layout/share.xml`



`res/layout-land/share.xml`

Layout Parameters



- Specify many aspects of what's being rendered
- Examples:
 - `android:layout_height`
 - `android:layout_width`
- Tip: start with documentation for a specific View or Layout and then look at what's inherited from parent class

Application Components



Basic components



Activities	UI component typically corresponding to one screen.
BroadcastReceivers	Respond to broadcast Intents.
Services	Faceless tasks that run in the background.
ContentProviders	Enable applications to share data.

Activities



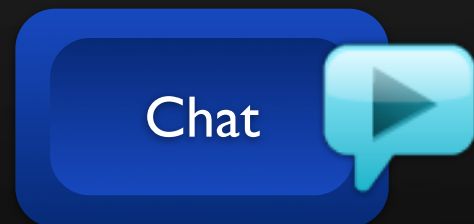
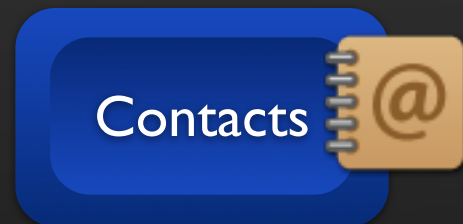
- Typically correspond to one screen in a UI
- But, they can:
 - be faceless
 - be in a floating window
 - return a value

Intents



- Think of Intents as a verb and object; a description of what you want done
- Examples: `VIEW`, `CALL`, `PLAY`, etc.
- System matches Intent with Activity that can best provide that service
- Activities and BroadcastReceivers describe what Intents they can service in their IntentFilters (via `AndroidManifest.xml`)

Intents



New components can use existing functionality

BroadcastReceivers



- Components designed to respond to broadcast Intents
- Think of them as a way to respond to external notifications or alarms
- Applications can invent and broadcast their own Intents as well

Services



- Faceless components that run in the background
 - Example: music player, network download, etc.
- Bind your code to a running service via a remote-able interface defined in an IDL
- Can run in your own process or separate process

ContentProviders



- Enables sharing of data across applications
 - Examples: address book, photo gallery, etc.
- Provides uniform APIs for:
 - querying (returns a Cursor)
 - delete, update, and insert rows
- Content is represented by URI and MIME type

Practical matters



Storage and Persistence



- A couple of different options:
 - Preferences
 - Flat file
 - SQLite
 - ContentProvider

Packaging



- Think of .apk files as Android packages
- Everything needed for an application is bundled up therein
- Basically a glorified ZIP file

Resources



- `res/layout`: declarative layout files
- `res/drawable`: intended for drawing
- `res/anim`: bitmaps, animations for transitions
- `res/values`: externalized values for things like strings, colors, styles, etc.
- `res/xml`: general XML files used at runtime
- `res/raw`: binary files (e.g. sound)

Assets



- Similar to Resources
- Differences:
 - Read-only
 - InputStream access to assets
- Any kind of file
 - Be mindful of file sizes

Application Lifecycle



- Application lifecycle is managed by the system
- Application start/stop is transparent to the user
- End-user only sees that they are moving between screens
- Read documentation for `android.app.Activity`

Toolchain



Emulator



- QEMU-based ARM emulator runs same system image as a device
- Use same toolchain to work with devices or emulator

Eclipse Plugin



Project template

A screenshot of the 'New Android Project' dialog box in the Eclipse IDE. The dialog has a title bar with standard window controls and the text 'New Android Project'. Below the title bar, it says 'Creates a new Android Project resource.' and features a yellow starburst icon with the Eclipse logo. The main area contains four text input fields: 'Project name:', 'Package Name:', 'Activity Name:', and 'Application Name:'. Below these is a checked checkbox labeled 'Use default location' and a 'Location:' field containing the path '/Users/mcleron/Documents/workspace-x', with a 'Browse...' button to its right. At the bottom, there are four buttons: a help icon (?), '< Back', 'Next >', 'Cancel', and 'Finish'.

Debugging



The screenshot shows the Eclipse IDE interface during a debug session. The top toolbar contains various debugging icons. The left sidebar shows the project structure with 'TelephonyEditor [Remote Java Application]' and 'DalvikVM[localhost:8700]'. The main area is divided into three panes:

- Call stack:** Located in the top-left pane, it shows the current method call stack. A yellow arrow points to the top frame: 'PhoneWindow\$DecorView(View).draw(Canvas, Transformation)'. Below it, other frames include 'PhoneWindow\$DecorView.draw(Canvas, Transformation, long)', 'PhoneWindow\$DecorView(View).drawTraversal(Canvas, Trans...', 'PhoneWindow\$DecorView(FrameLayout).drawTraversal(Canva...', 'PhoneWindow\$DecorView.drawTraversal(Canvas, Transformat...', 'ViewRoot.draw(boolean) line: 373', 'ViewRoot.performTraversals() line: 288', 'ViewRoot.handleMessage(Message) line: 421', and 'ViewRoot(Handler).dispatchMessage(Message) line: 40'.
- Examine variables:** Located in the top-right pane, it displays the current state of variables. A yellow arrow points to the 'drawTime' variable. The table below shows the following data:

Name	Value
this	PhoneWindow\$DecorView (id=830011...)
canvas	Canvas (id=830011571256)
mBitmap	null
mNativeCanvas	1249144
mSurfaceFormat	4
currentTransformation	null
drawTime	1179363240090
- Breakpoints, single stepping:** Located in the bottom pane, it shows the source code of 'View.java'. A yellow arrow points to line 1827, which contains the code: 'if (bg == null) {'. The code snippet is as follows:

```
1823 * @param drawTime The time at which the current draw pass started
1824 */
1825 protected void draw(Canvas canvas, Transformation currentTransformation, long drawTime) {
1826     Drawable bg = mBGDrawable;
1827     if (bg == null) {
1828         mBackgroundSizeChanged = false;
1829         return;
1830     }
1831     ...
    - mLeft, mBottom - mTop);
```

The bottom status bar includes 'Console', 'Tasks', 'Display', and 'Search' tabs.

Eclipse demo



Q & A

