Android Application Development

Daniel Switkin
Senior Software Engineer, Google Inc.

Goal

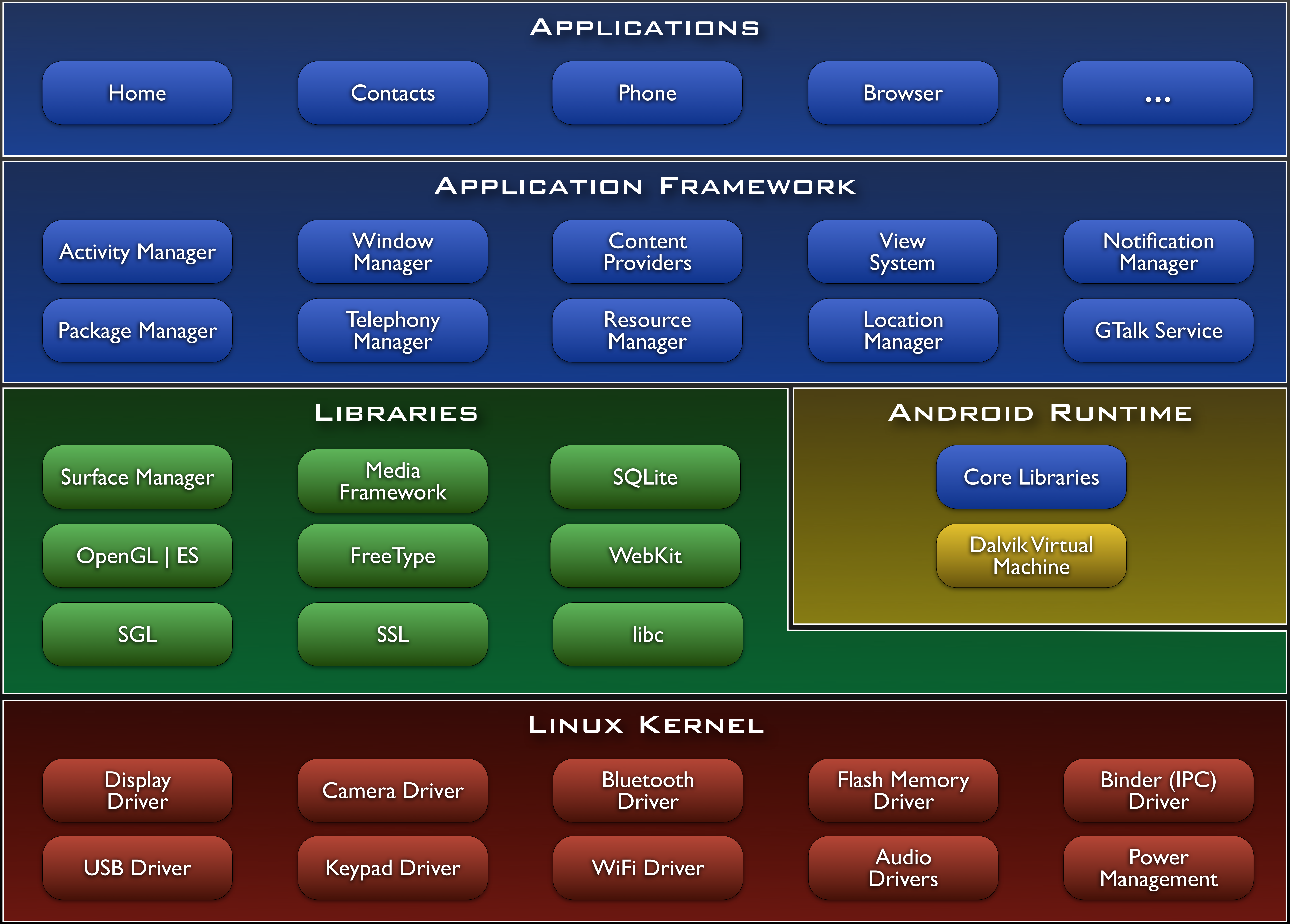
- Get you an idea of how to start developing Android applications
- Introduce major Android application concepts
- Walk you through a sample application in the development environment

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

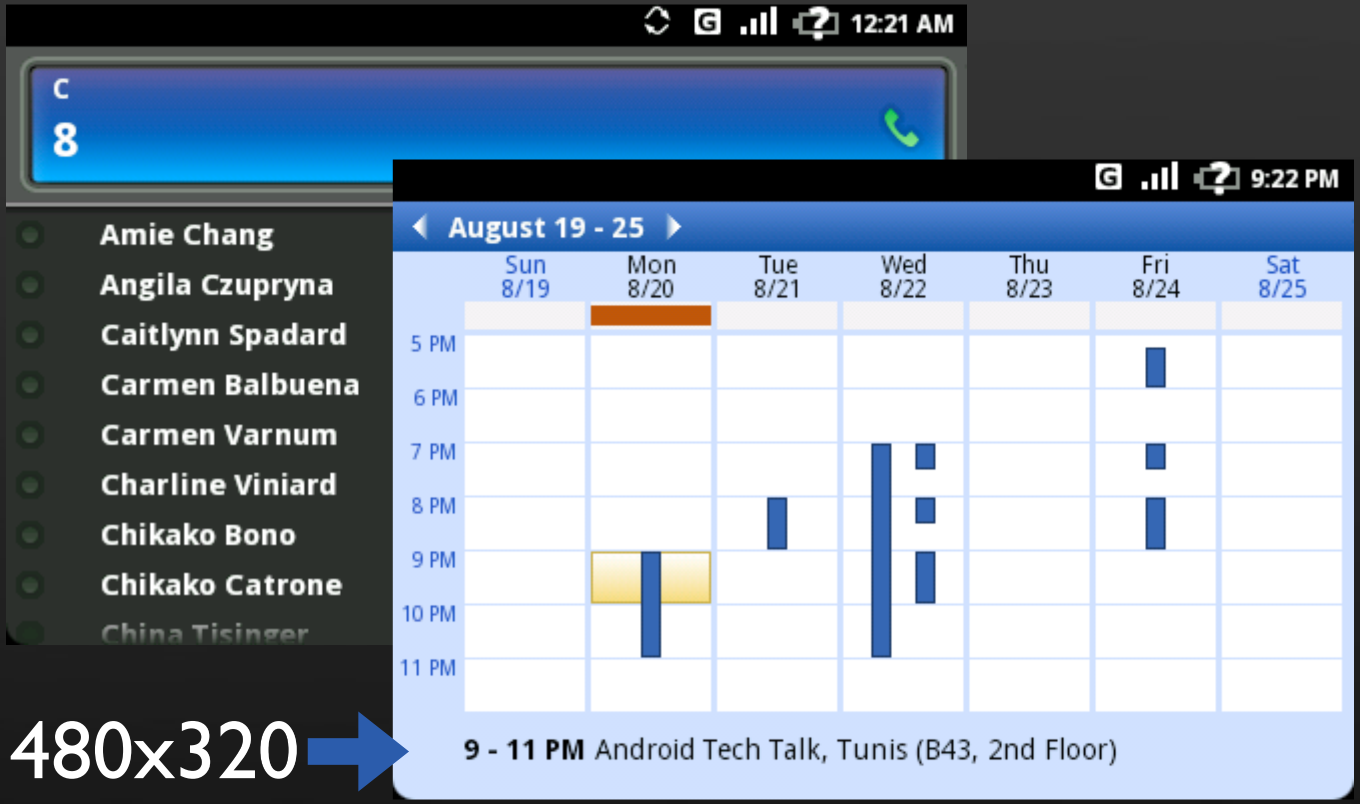
Views

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

Layouts

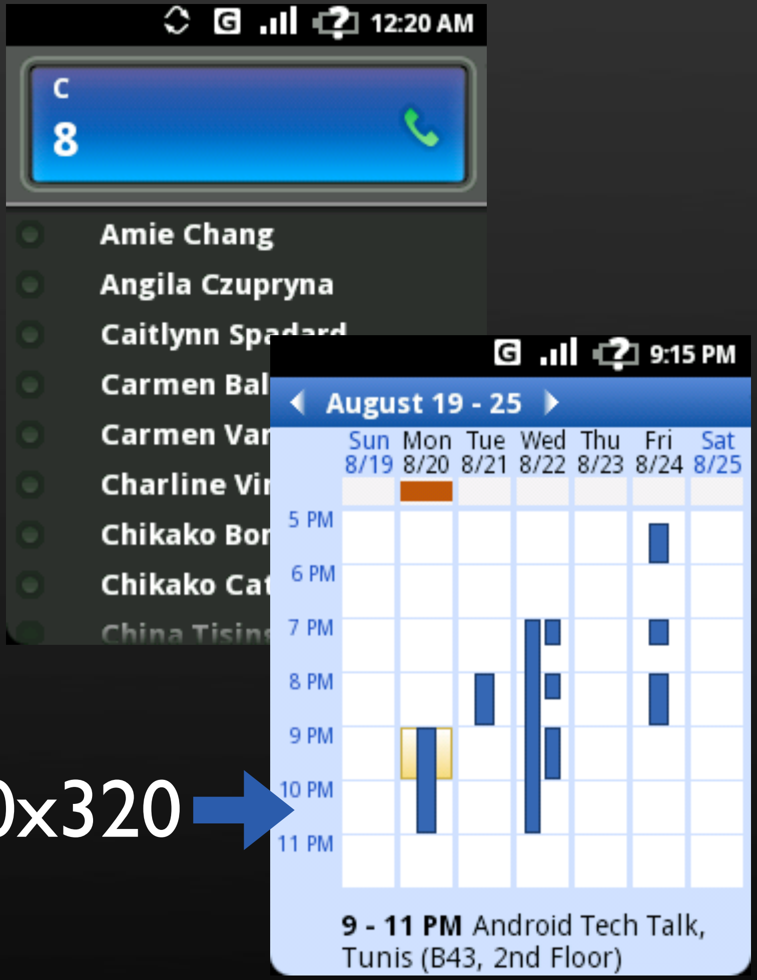
- Controls how Views are laid out
  - FrameLayout : each child a layer
  - LinearLayout : single row or column
  - RelativeLayout : relative to other Views
  - TableLayout : rows and columns
  - AbsoluteLayout : <x,y> coordinates

Layouts are resizable



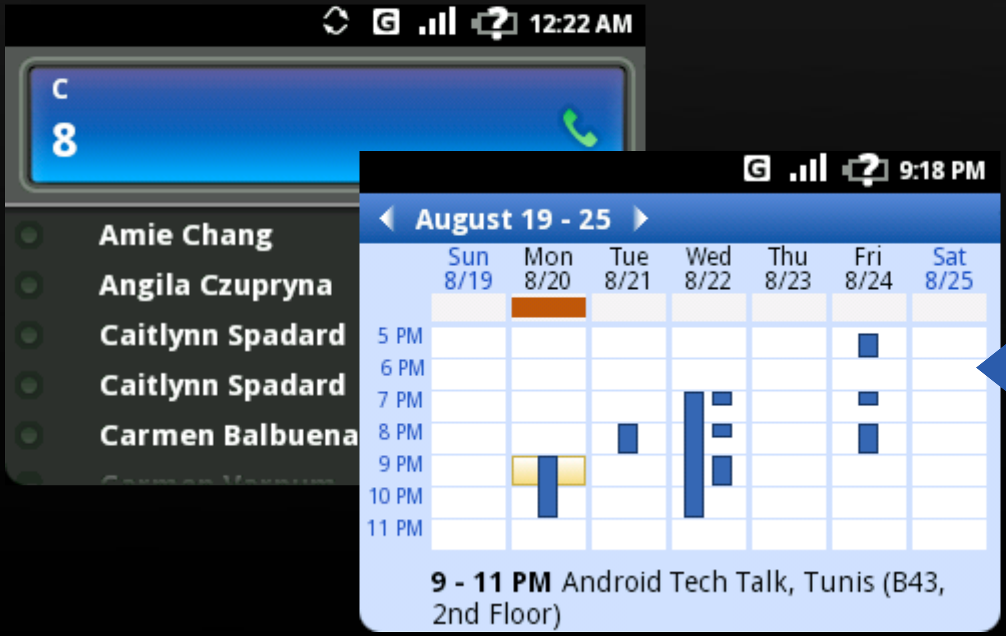
9 - 11 PM Android Tech Talk, Tunis (B43, 2nd Floor)

480x320



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

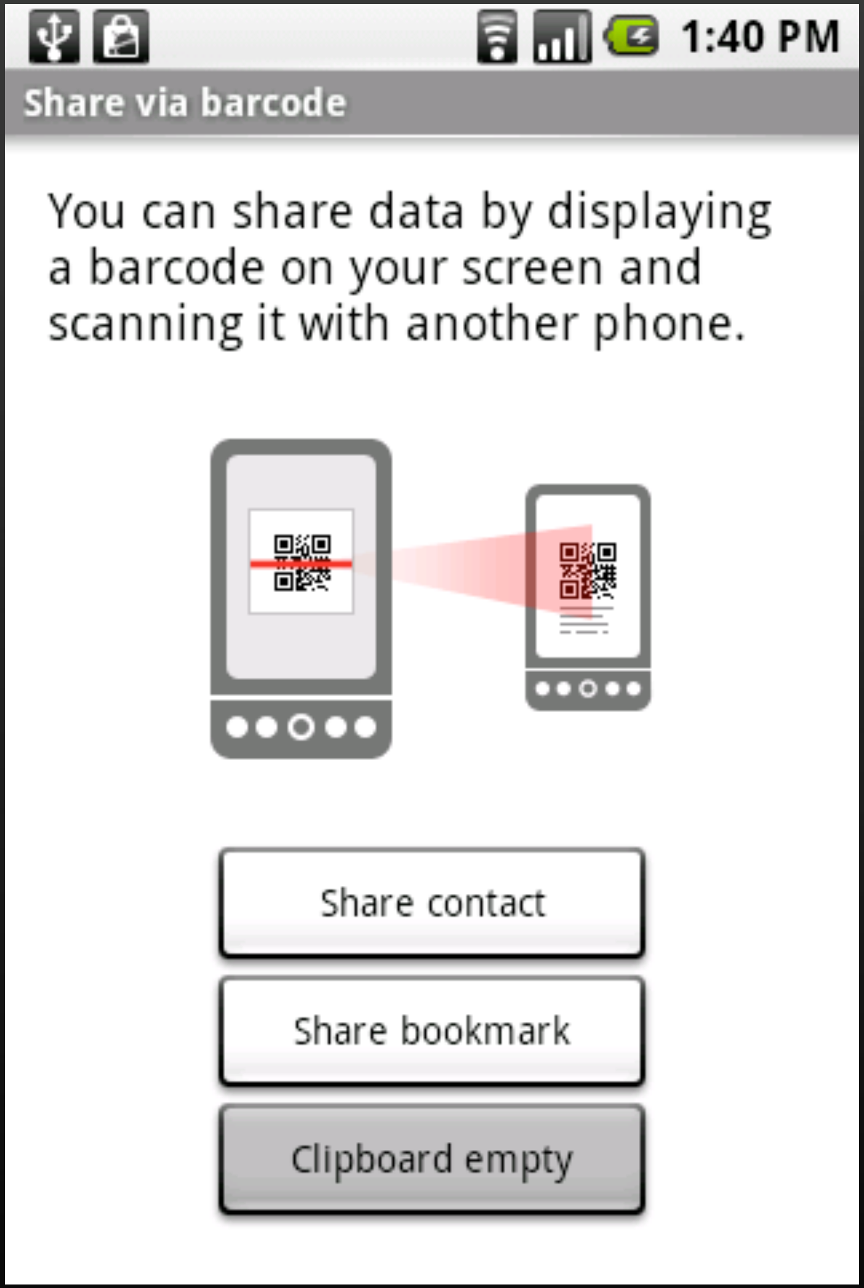
240x320



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

320x240

Layouts are customizable

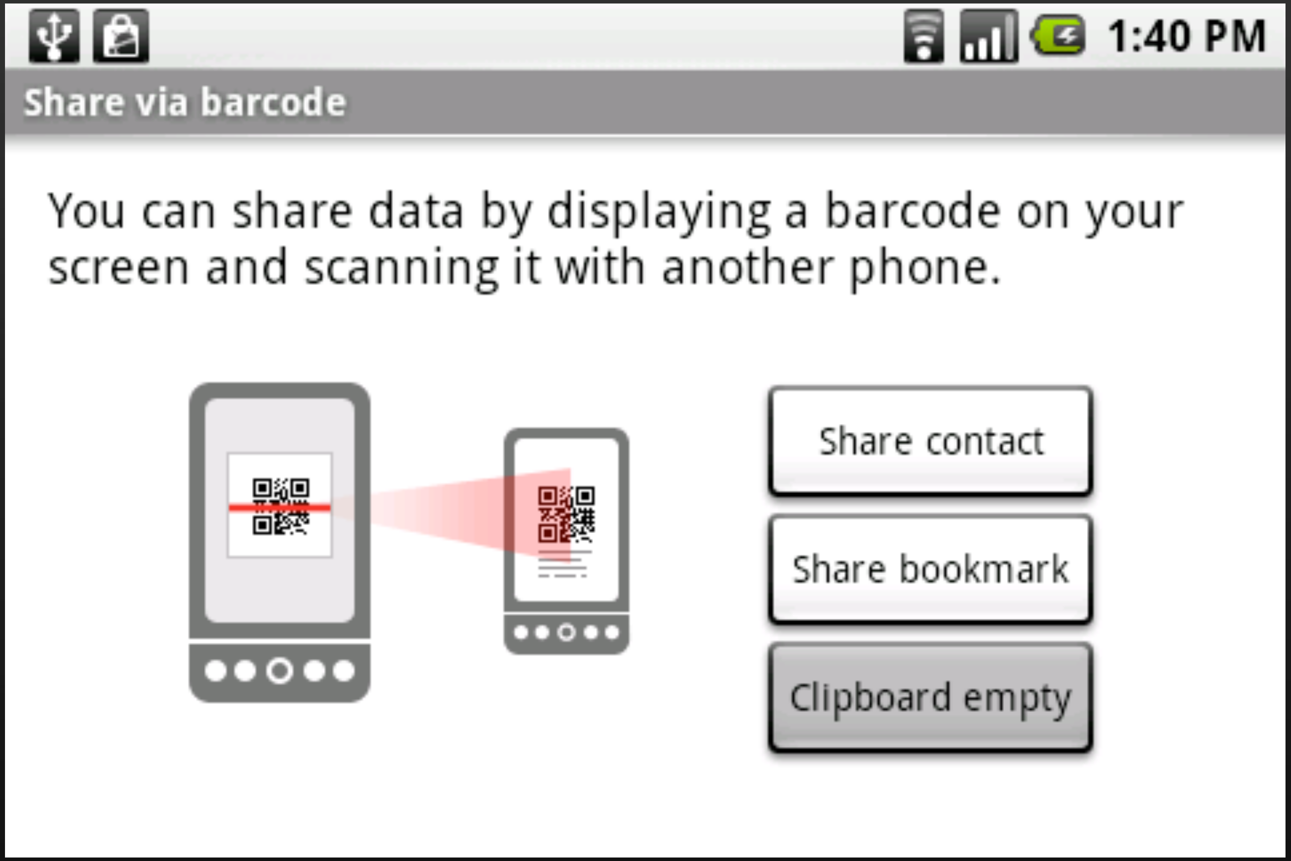


1:40 PM

Share via barcode

You can share data by displaying a barcode on your screen and scanning it with another phone.

`res/layout/share.xml`



1:40 PM

Share via barcode

You can share data by displaying a barcode on your screen and scanning it with another phone.

`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

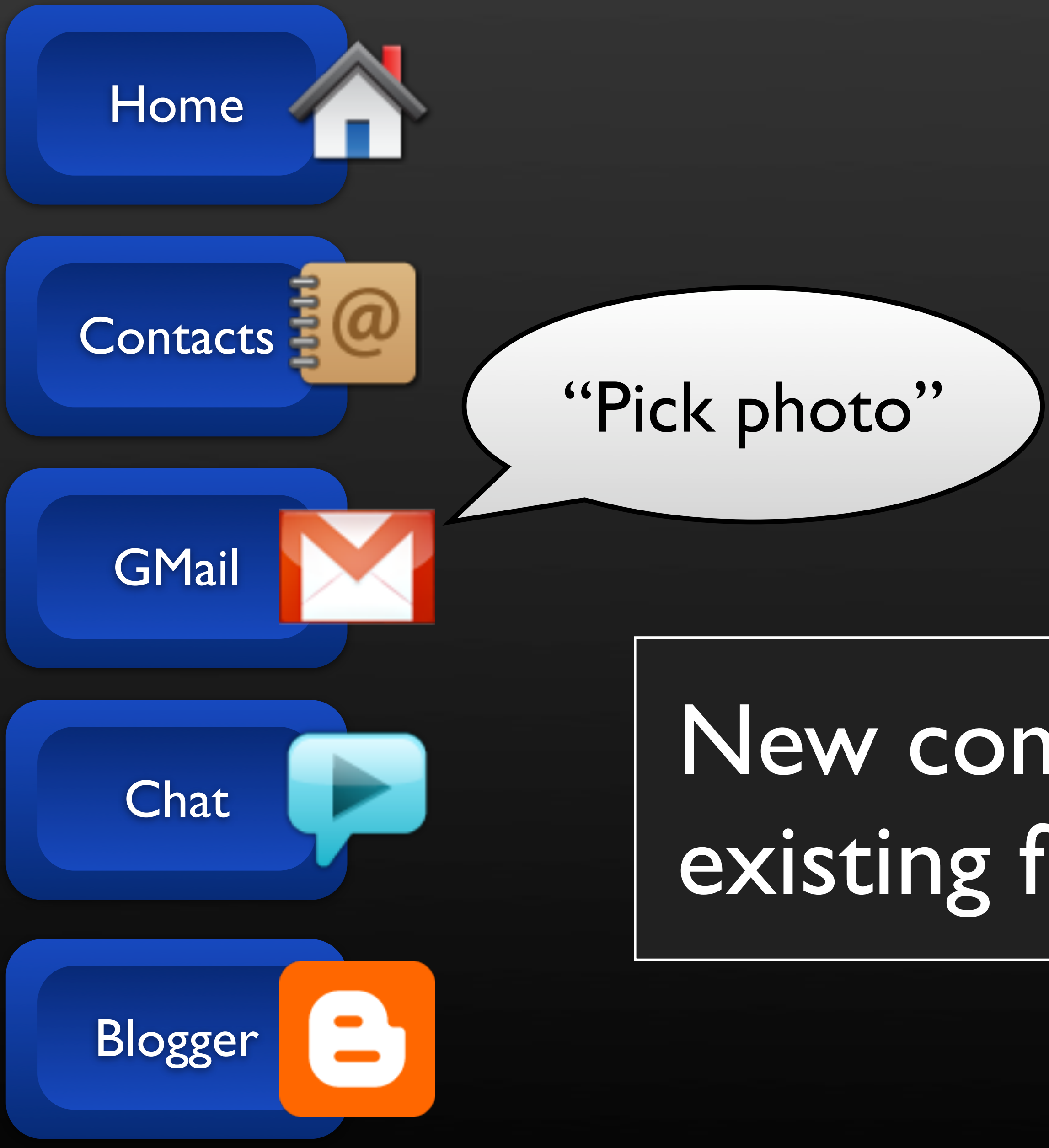


Photo Gallery

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 downlaod, 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`

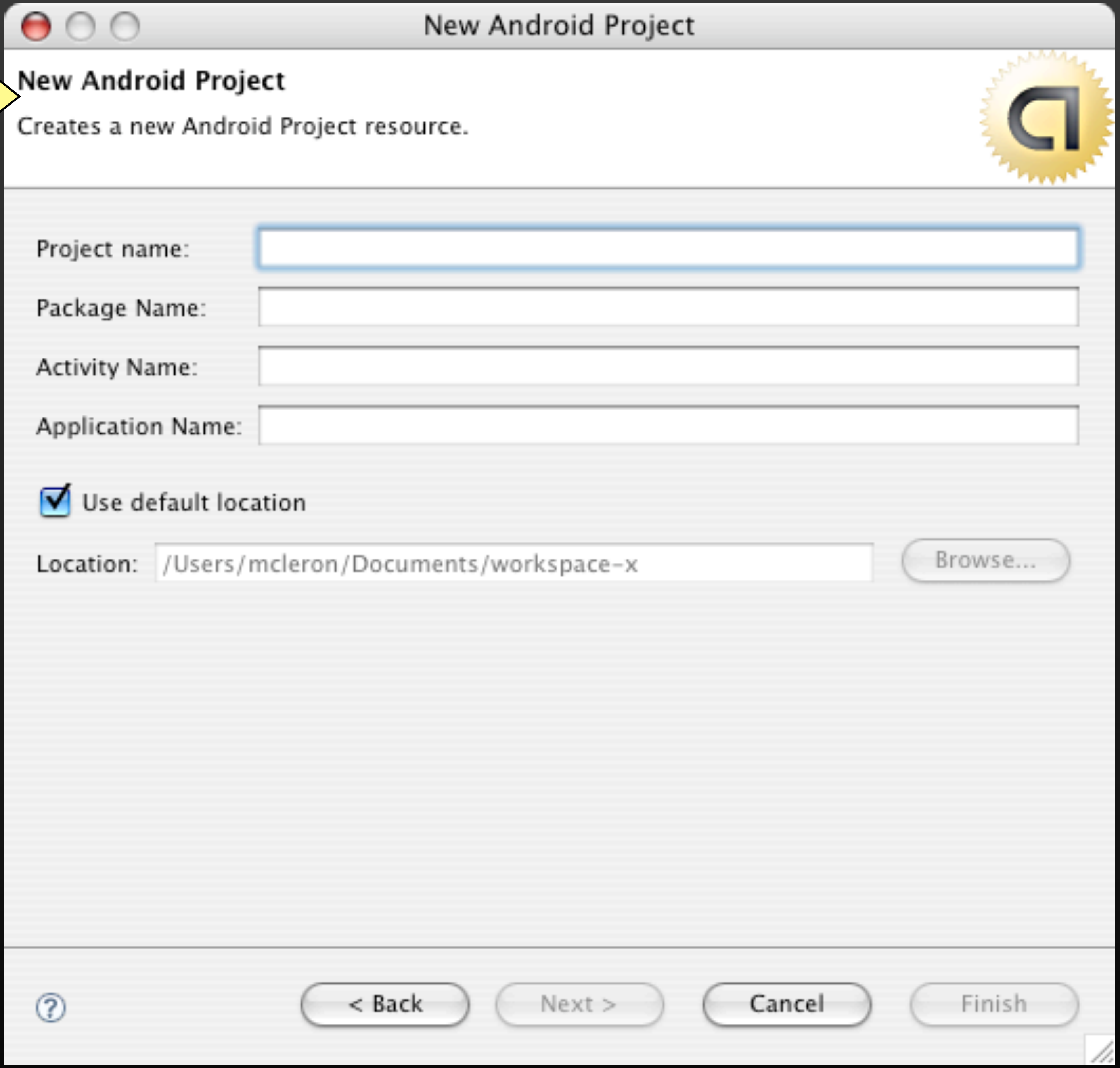
Toolchain

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


New Android Project

New Android Project

Creates a new Android Project resource.

Project name:

Package Name:

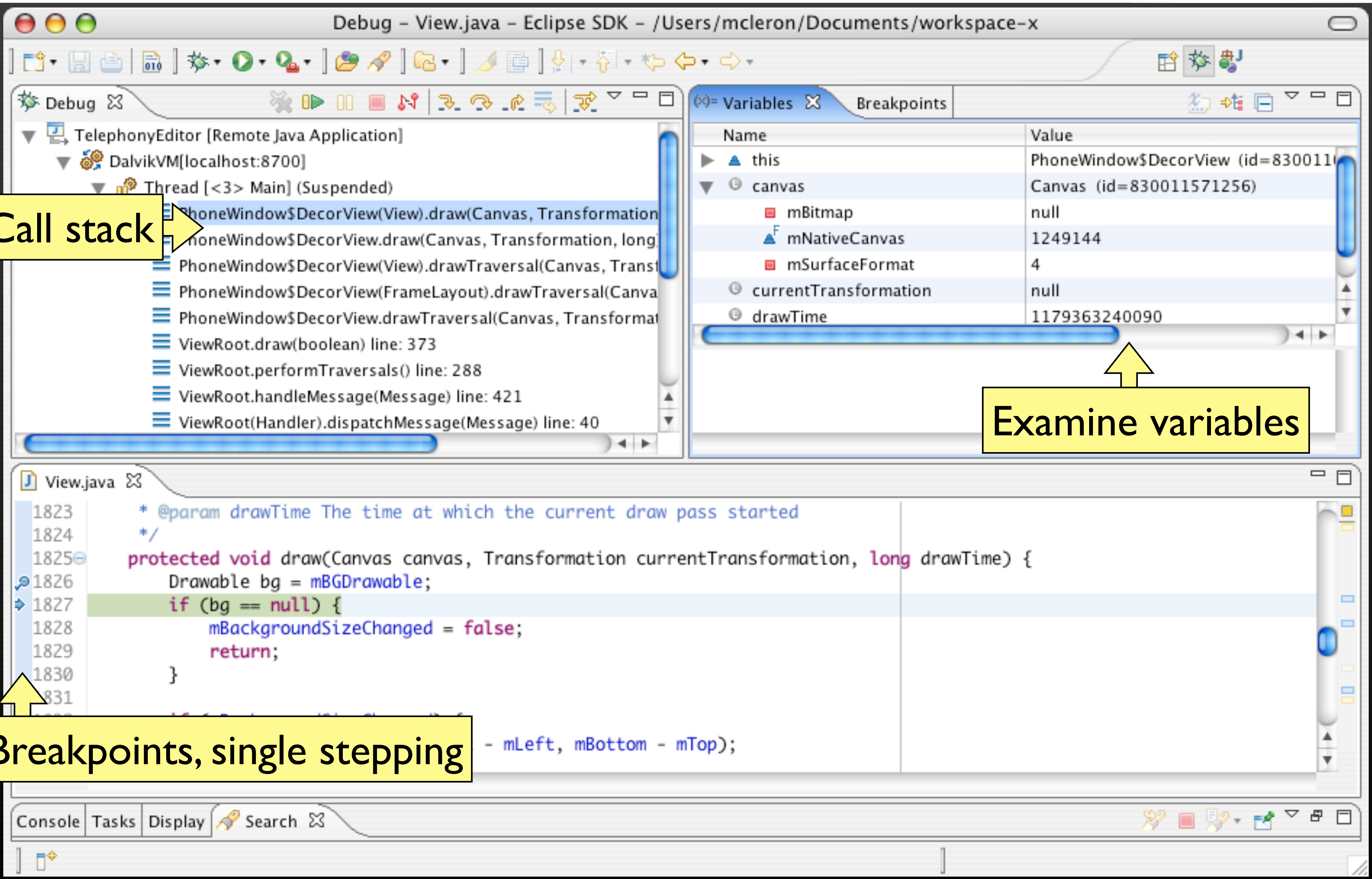
Activity Name:

Application Name:

✓ Use default location

Location: /Users/mcleron/Documents/workspace-x

Debugging



Debug – View.java – Eclipse SDK – /Users/mcleron/Documents/workspace-x

```
TelephonyEditor [Remote Java Application]
  DalvikVM[localhost:8700]
    Thread [<3> Main] (Suspended)
      PhoneWindow$DecorView(View).draw(Canvas, Transformation
      PhoneWindow$DecorView.draw(Canvas, Transformation, long
      PhoneWindow$DecorView(View).drawTraversal(Canvas, Transf
      PhoneWindow$DecorView(FrameLayout).drawTraversal(Canva
      PhoneWindow$DecorView.drawTraversal(Canvas, Transformat
      ViewRoot.draw(boolean) line: 373
      ViewRoot.performTraversals() line: 288
      ViewRoot.handleMessage(Message) line: 421
      ViewRoot(Handler).dispatchMessage(Message) line: 40
```

Name	Value
this	PhoneWindow$DecorView (id=830011
canvas	Canvas (id=830011571256)
mBitmap	null
mNativeCanvas	1249144
mSurfaceFormat	4
currentTransformation	null
drawTime	1179363240090

```java
     * @param drawTime The time at which the current draw pass started
     */
    protected void draw(Canvas canvas, Transformation currentTransformation, long drawTime) {
        Drawable bg = mBGDrawable;
        if (bg == null) {
            mBackgroundSizeChanged = false;
            return;
        }

- mLeft, mBottom - mTop);
```

Eclipse demo

Q & A