BEGIN ANDROID JOURNEY IN HOURS
CS425 / CSE 424 / ECE 428 [Fall 2009]

Sept. 14, 2009
Ying Huang
REFERENCE

- **Online development guide**

- **Book resource**
MOBILE OS

- Symbian
- iPhone
- RIM's BlackBerry
- Window mobile
- Linux
- Palm webOS
- Android
- ....
**What is Android?**

- **Google → OHA (Open Handset Alliance)**
  - *The first truly open and comprehensive platform for mobile devices, all of the software to run a mobile phone but without the proprietary obstacles that have hindered mobile innovation.*

- Linux OS kernel
- Java programming
- Open source libraries: SQLite, WebKit, OpenGL
WHY ANDROID

- A simple and powerful SDK
- No licensing, distribution, or development fees
- Development over many platform
  - Linux, Mac OS, windows
- Excellent documentation
- Thriving developer community

For us
- Java-based, easy to import 3rd party Java library
- Funding (40+ G1 phones)
- Prize (amazon’s kindle)
- Job opportunity
Android SDK Feature

- GSM, EDGE, and 3G networks, WiFi, Bluetooth
  - API Support for Bluetooth, WiFi Ad-hoc mode
- Libraries
  - Media, SQLite, WebKit, SSL
- Hardware control: MP3
  - Accelerometer, compass, microphone, camera, GPS
  - Touch screen, power
- Location-based service, map (Google API) MP3
TOOLS

- The Android Emulator
  - Implementation of the Android virtual machine
  - Test and debug your android applications.
- Dalvik Debug Monitoring Service (DDMS)
  - Monitor and Control the Dalvik virtual machines
  - Logcat (see logged msgs)
- Android Debug Bridge (ADB)
  - Manage the state of an emulator instance or Android-powered device
  - Copy files, install compiled application packages, and run shell commands.
- Traceview
  - Graphical analysis tool for viewing the trace logs from your Android application
  - Debug your application and profile its performance
- MkSDCard
  - Creates an SDCard disk image
RUN TWO NETWORKING EMULATORS IN A COMPUTER (UPDATED) - ADB

Q: run two networking emulators in a computer A using the public IP address of A, during debugging and demo?

A1: telnet + redir (MP1 doc)

A2: adb forward
  1) Port forward to connect Android from localhost
     a. abd –s emulator-5554 forward tcp:15216 tcp:15216
  2) Use a proxy server which can listen on my_public_ip:15216 and forward the data to localhost:15216
     stcppipe localhost 15216 15216
Underlying Infrastructure-based WiFi /TCP/IP Network

Peer Registration Server

G1 Phone (Peer)

Peer Peer
MP1 Project Structure

Registration
PeerList
Messaging

ANDY:
- There, Julie?
- We have a party tomorrow. Do you wanna join us?
ANDROID APPLICATION ARCHITECTURE

- **Views:**
  - Building block for user interface components.

- **Activities**
  - A single, focused thing that the user can do.
  - Interaction with users: creating a window to place UI
  - full-screen windows, floating windows, embedded inside of another activity
  - **Ex: Registration, Peerlist, Messaging GUI**
ANDROID APPLICATION ARCHITECTURE

- **Services (Background)**
  - Ex: Network Operation

- **Intent**
  - Inter-communication among activities or services

- **Resource**
  - Externalization of strings and graphics

- **Notification**
  - Signaling users: Light, sound, icon, dialog, notification
    - Ex: new message arrives

- **Content Providers**
  - Share data between applications
**VIEW**

- **Layout of visual interface**

  ![Image of layout](image)

- **Java Code**
  
  - Initialize
    ```java
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);
        setContentView(R.layout.screen);
    }
    ```
  
  - Access
    ```java
    TextView myTextView = (TextView)findViewById(R.id.myTextView);
    ```

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:id="@+id/myTextView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Hello World, HelloWorld"/>
</LinearLayout>
```
**VIEW COMPONENT**

- **Widget Toolbox**
  - `TextView`, `EditText`, `Button`, `Form`, `TimePicker`...
  - `ListView` (`PeerList`)
    - Update list by arrays
      - `ArrayAdapter`
      - `myListView.setAdapter`
  - **Layout**
    - Positions of controls
    - `LinearLayout`, `RelativeLayout`

- **Menu**
  - Exit app
ACTIVITY

- Foreground Activity: suspended when invisible
  - Visual, interactive
  - Ex: Game, Map

- Background Service: Little interaction
  - Ex: Hardware, power management

- Intermittent Activity
  - Notification, service
  - Expects some interactivity but does most of its work in the background.
USER INTERACTION EVENT

- onKeyDown, onKeyUp
- onTrackBallEvent
- onTouchEvent

```java
registerButton.setOnClickListener(new OnClickListener() {
    public void onClick(View arg0) { .... }
});

myEditText.setOnKeyListener(new OnKeyListener() {
    public boolean onKey(View v, int keyCode, KeyEvent event) {
        if (event.getAction() == KeyEvent.ACTION_DOWN)
            if (keyCode == KeyEvent.KEYCODE_DPAD_CENTER)
                { .... }
        return false;
    }
});
```
APPLICATION AND COMPONENT GLUES

- An intent is an abstract description of an operation to be performed.

  - Launch an activity
    - Explicit
    - Implicit: Android selects the best
    - `startActivity();`

  - Subactivity: feedback
    - Child: use intent as feedback, `setResult`
    - Parent: `onActivityResult`
    - `startActivityForResult`

- Action, data, extra parameter
  - `intent.putExtra(name, property);`

  **Example: Intent intent = new Intent(MyActivity.this, MyOtherActivity.class);**
  **Implicit: Intent intent = new Intent(Intent.ACTION_DIAL, Uri.parse("tel:555-2368");**
**INTENT (CNTD.)**

- **Broadcast**
  - announce application events system-wide
  - sendBroadcast
  - MyBroadcastReceiver extends BroadcastReceiver
  - registerReceiver (in java / in xml)

- **Intent Filter**
  - Register Activities, Services, and Broadcast Receivers as being capable of performing an action on a particular kind of data.

Manifest.xml

```xml
<activity ...>
  <intent-filter>
    <action
      android:name="com.paad.earthquake.intent.action.SHOW_DAMAGE">
      </action>
    <category
      android:name="android.intent.category.DEFAULT"/>
    <category
      android:name="android.intent.category.ATTRIBUTE_SELECTED"/>
    <data
      android:mimeType="vnd.earthquake.cursor.item/*"/>
  </intent-filter>
</activity>
```
INTENT FROM PEERLIST TO MESSAGING

Select a peer
Send Intent

PeerList

Messaging

ANDY:
- There, Julie?
- We have a party tomorrow. Do u wanna join us?
**WORKING IN BACKGROUND**

- **Services**
  - NO GUI, *higher* priority than inactive Activities
  - Usage:
    - responding to events, polling for data, updating Content Providers.
  - However, all in the main thread

- Background threads
**SERVICE**

- **Service class**
  - public class MyService extends Service
  - public void onStart() {...}

- **Manifest.xml**
  - <service android:enabled="true"
    android:name=".MyService"></service>

- **Control**
  - startService
  - stopService

- **Communication**
  - Bind service with activity: use public method and properties
  - Intent
THREADING

- **Being Responsive (1sec)**
  - Respond to any user action within 5 seconds.
  - A Broadcast Receiver must return within 10 seconds.

- What to thread?
  - Network, file IO, Complex processing

- How?
  - New Thread
  - Synchronize threads
    - Handler.post()
MP1 Threading Structure

- Is it enough?
PERIODICAL REGISTER WITH SERVER

- Every 15 seconds

How to update PeerList on PeerList GUI?
  - Intent
ACTIVITY LIFETIME

- Android apps do not control their lifetime.

- Active (Foreground) - Paused (FG, lose focus) - Stopped (invisible) – Inactive (kill, exit)

create Services and threads

threads, processes, or Broadcast Receivers

Broadcast Receivers exclusively used to update UI
Declaring an App – Manifest.xml

- Service

- Activity (intent-filter)

- Permission
  - Don’t forget. Otherwise, your socket programming won’t run.
EXTERNAL RESOURCES

- values/
  - String, color, array, dimension, style theme
- drawables/
  - Image
- layout/
  - screen.xml
DEBUG

- `System.err.println()`
- Package - `android.util.Log`

View results
- Logcat
- Eclipse IDE
**DEBUG ON DEVICE**

- **On device**
  - Debug mode

- **On desktop**
  - Connect your G1 with your PC
  - When it asks for driver location choose
    - For windows, android-sdk-windows-1.5_r3\usb_driver\x86\
  - You'll see sth like "HTC Dream Composite ADB Interface" on success
  - (Re)Start Eclipse
  - Your G1 should now be listed in the DDMS-Perspective under Device

- Reference: [http://www.anddev.org/debugging-installing_apps_on_the_g1_windows_driver-t3236.html]
INSTALL PACKAGE TO ANDROID PHONES

- Compile the apk packages in Eclipse
  - Export signed application package

- adb install ...apk
  - Error: uninstall
OTHER TIPS

- **Automatic imports of packages**
  In Eclipse, cmd-shift-o or ctrl-shift-o

- Start as early as possible
  - Possible task partition for a group
    - P1: GUI, activity, intent
    - P2: network service
    - P3: integration
DEMO
DALVIK DEBUG MONITORING SERVICE
ANDROID DEBUG BRIDGE (ADB)

C:\Windows\system32\cmd.exe

C:\Users\Ying>adb help
Android Debug Bridge version 1.0.20

-d               - directs command to the only connected USB device
                 returns an error if more than one USB device is present.
-e               - directs command to the only running emulator.
                 returns an error if more than one emulator is running.
-s <serial number> - directs command to the USB device or emulator with
                    the given serial number
-p <product name or path> - simple product name like ‘sooner’ or
                            a relative/absolute path to a product
                            out directory like ‘out/target/product/sooner’.
                            If -p is not specified, the ANDROID_PRODUCT_OUT
                            environment variable is used, which must
                            be an absolute path.

devices
     - list all connected devices

device commands:
adb push <local> <remote> - copy file/dir to device
adb pull <remote> <local> - copy file/dir from device
adb sync [ <directory> ] - copy host->device only if changed
                            <see ‘adb help all’>
adb shell      - run remote shell interactively
adb shell <command> - run remote shell command
adb emu <command> - run emulator console command
adb logcat [ <filter-spec> ] - View device log
adb forward <local> <remote> - forward socket connections