Release and Monetization

CS 2046
Mobile Application Development
Fall 2010
Announcements

• Assignment 3 due Friday, 11/19
• Office Hours next week (on course website):
  – Jeff: MF 11:15 - 12:15
  – Jae: W 12 - 1
• Final lecture today!
• Course evaluations
Prototype ➔ Release

• Where we stand:
  – Working prototype of application
    • Works on emulator, or personal device
    • Full-featured
  – Expects some API version for full functionality
    • Perhaps started with 3, incremented as new features were needed
Release Checklist

• Finalize Functionality

• Prepare for Business Model

• Release
Supporting Multiple Versions

Optional Features

• Which features are required?
• Which features are nice, but not necessary?

• Step 1: Declare with `<uses-feature>`
• Step 2: Dynamic loading of classes
<uses-feature>

- Informs Android (Market) of the hardware and software on which your application depends.

- Examples:
  - `<uses-feature android:name="android.hardware.bluetooth" />`
  - `<uses-feature android:name="android.hardware.camera" />`

- `android:required` attribute specifies need.

- List of features & Market filtering rules:
Targeting an SDK

• Set the build target to the lowest version supporting all *optional* components.

• Set `<uses-sdk android:minSdkVersion="..." />` in Manifest to lowest version supporting all *required* components.

• Simply declaring feature as optional is clearly not enough
  – Must make sure program functions in both cases
Reflection

• Reflection is a method of dynamically loading classes.
  – If class/method is present, it is used.
  – Otherwise, we catch the event (instead of crashing) and respond accordingly.
Example - ScaleGestureDetector

```java
public class ScaleGestureWrapper {
    private ScaleGestureDetector mInstance;

    static {
        try {
            Class.forName("ScaleGestureDetector");
        } catch (Exception ex) {
            throw new RuntimeException(ex);
        }
    }

    public static void checkAvailable() {}

    public ScaleGestureWrapper(Context c, OnScaleGestureListener osl) {
        mInstance = new ScaleGestureDetector(c, osl);
    }

    public boolean onTouchEvent(MotionEvent event) {
        return mInstance.onTouchEvent(event);
    }
}
```
public class ScaleGestureActivity extends Activity {
    private static boolean mAvailable = false;
    private ScaleGestureWrapper mWrapper;

    static {
        try {
            ScaleGestureWrapper.checkAvailable();
            mAvailable = true;
        } catch (Throwable t) {} 
    }

    public void onCreate(Bundle savedInstanceState) {
        if (mAvailable) {
            mWrapper = new ScaleGestureWrapper(this, null);
        }
    }

    public boolean onTouchEvent(MotionEvent e) {
        if (mAvailable) {
            return mWrapper.onTouchEvent(e);
        }
        return super.onTouchEvent(e);
    }
}
Avoiding Reflection

• With too many new features, reflection can get messy and cumbersome to use.

• Alternative approach: singletons and lazy class loading.

• Example – gestures on Cupcake-Froyo
Finishing Application

• Recall: UI guidelines

• Other pages listed under “Best Practices”:
  – Compatibility
  – Supporting Multiple Screens
  – Designing for:
    • Performance
    • Responsiveness
    • Seamlessness

• Run through each guideline and make sure you meet them where appropriate.
Finishing Application

- Design icon, specify label

- Remove android:debuggable="true" from <application> tag in Manifest.

- Remove unnecessary files from project.

- Remove any calls to Log (or System.out)
Release Checklist

- ✔ Finalize Functionality
- Prepare for Business Model
- Release
Three Models

- Free, Ad-Supported, Paid
  - Can also mix between some subset
- Tradeoff:
The “Freemium” Model

• Akin to “shareware” that was big in the 1990s

• Two versions:
  – Free, but hampered
    • Time-limited or use-limited trial
    • Missing extra functionality
    • Advertising
    • Or, some combination of above.
  – Full featured, but for a price

• Aim: Same expected earnings for both versions
  – Reality – very tough to calculate
Implementing Freemium on Android

• Android Market keys based on package.
  – Copy program to new package name (e.g. *.free)

• Modify free version as desired
  – Integrate mobile ad framework
  – Remove features
    • Can leave menu item in, but replace functionality with link to premium app.
Mobile Ads

• Advertisers sign up with ad networks, give ads to display.

• Ad networks show these ads in applications developed by others

• Equivalent to Google Adsense

• Examples of networks:
  – Admob (Google Mobile Ads), Mobclix, Smaato, iAd
Example – Integrating AdMob

• Register at http://www.admob.com
• Add new Android App
• Download Android SDK
  – Includes admob-sdk-android.jar
  – Import into Eclipse workspace, right click, add to build path.
    • Side note: works for other 3rd party Java libraries
• Add entries to AndroidManifest
• Add com.admob.android.ads.AdView to your application.
Ad Dashboard

- Advanced tracking of revenue
  - Measure if ad placement is successful
  - Help ensure ad revenues match paid product revenue.

### Today's Revenue

<table>
<thead>
<tr>
<th></th>
<th>Revenue</th>
<th>eCPM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yesterday</td>
<td>$0.00 (0%)</td>
<td>$0.00 (0%)</td>
</tr>
<tr>
<td>Last 7 Days</td>
<td>$0.00 (0%)</td>
<td>$0.00 (0%)</td>
</tr>
</tbody>
</table>

Last updated on 2010-11-11 22:00:00 GMT
Licensing

• Ads insure earnings on free version.
• Market sales insure earnings on paid version.
  – But how to stop piracy?
    • Impossible battle to win 100% of the time
    • But, possible to win 90% with far less effort.

• Solution: Android Market Licensing
  – Downloadable component of Android SDK
Android Market Licensing Overview

• Network-based service

• Query licensing server to determine whether current device is licensed.
  – Application responsible for reaction

• Prevents basic copying from device to device
  – With additional obfuscation, can make it even more difficult to copy.

• Main interface: License Verification Library (LVL)
Integrating LVL

• Choose a Policy
  – What to do for a given user with a given license
  – Two provided implementations:
    • ServerManagedPolicy – flexible, cache responses if network is down
    • StrictPolicy – only runs application if server says licensed
  – Can also implement custom policy

• Check license from main Activity

• For full guide, see: http://developer.android.com/guide/publishing/licensing.html
Additional Steps

• LVL prevents casual privacy – these make it even more difficult.
  – From:

• Obfuscate application
  – Prevents looking at strings in disassembled code to figure out what program is doing.

• Modify license library
  – (Library itself is actually open source)
  – Interface is fixed, but can change behavior so that no two apps work the same way.

• Prevent tampering
  – Checksum application code and verify at runtime

• Offload validation to a trusted server
Release Checklist

- ✓ Finalize Functionality
- ✓ Prepare for Business Model
- Release
Testing

• **JUnit instrumentation tests**
  – Important for avoiding regressions

• **UI/Application Exerciser Monkey**
  – Test atypical user flows

• Test on different emulators
  – Different versions of Android SDK
  – Different configurations - -dpi, -device, -scale, -netspeed, -netdelay, -cpu-delay...

• Test on as many hardware devices as possible.
  – Beta test groups
Android Market

• Centrally hosted service for nearly all users of Android phone to purchase and/or download your app.
  – Android is open – other app stores can and do exist.

• Register at http://market.android.com/publish/
  – $25 to become a developer
    • vs. $99/year for the iOS App Store
  – Central dashboard for posting updates, viewing statistics, reviews, crashes, etc.
Release Checklist

- Finalize Functionality

- Prepare for Business Model

- Release

- And we’re done!
  - Of course, for updates, process repeats.