Networking and Publishing

CE881: Mobile and Social Application Programming

Spyros Samothrakis

March 07, 2016

Theme: "Networking"

- "Cyberspace"-themed novels and movies
  - Tron
  - The Matrix
- Vannevar Bush
  - "Science, The Endless Frontier"
  - "As we may think"
- "Sciences of the Artificial"

Overview

- Motivation
  - Multi-user apps (one user per device)
  - Social apps
  - Better connected apps
- Technology
  - Simple sharing with Intents
  - Network Connectivity and WiFi Direct
  - Overview of Facebook, Twitter API

For some Apps, essential (e.g. email, Facebook)

- Can boost functionality of others
- Shopping list -> Shared Shopping List
- Game -> Multi-User Game
- Marketing and socialisation
- New game high scores / achievements auto-tweeted or posted to Facebook

Object Networking APIs: a Brief history

- Naturally want OO programs to run in distributed ways over networks
- There have been some failed approaches to this in the past:
- CORBA
  - Failed due to being far too heavyweight and incredibly complex to set up and run
- SOAP
  - Failed because it was poorly designed and did not deliver on expectation (Simple Object Access Protocol: it was not simple and did not give access to Objects !!!)

REST?

- Ideally, software evolves to strike a fine balance between functionality and simplicity: what is the simplest way to deliver a useful set of functions
- REST: stands for REpresentational State Transfer
- It uses URIs to represent the state of a system
- Requests can be made by passing parameters to an HTTP GET Request
  - Or can use any other HTTP Verb
- These are usually packed into the arguments of the URL

REST Example

- Google have been a major adopter of this approach
- Example: the Google Maps API
- Can construct query strings for things like Reverse GeoCoding
- And Navigation / Route Planning
- Can easily use these APIs within Android Apps

Reverse Geo Coding Example (JSON/XML)

- `https://maps.googleapis.com/maps/api/geocode/json?latlng=51.77634752,0.58275396&sensor=true`
  - Returns a JSON object
- `http://maps.googleapis.com/maps/api/directions/xml?origin=Colchester&destination=Chelmsford&sensor=false`
  - Returns an XML Object

REST Approach Summary

- Encode parameters within URL string
- This specifies the:
  - Server
  - Path and hence Web Application within it
  - Parameters to pass to the receiving Web App
- Client normally makes an HTTP GET request
- The server reads the parameters and responds
- Can respond with any format including:
  - Plain text
  - XML
  - JSON
- XML and JSON are very useful
- Client (your Android app) must then parse the response and do something useful with it

Basic Networking

- Basic Network Connectivity
  - We've already seen automatic use of Network Connectivity
  - For example, when setting a URL in a WebView, the WebView automatically connects to the the WebServer identified by the URL
- It then loads the HTML and displays it in the view
- We're now going to work through a similar example, except that
  - We'll manage the process step by step

Interacting with a Web Server (HTTP)

- Main steps are, given a URL e.g. http://google.com
  - Enable Internet access in the manifest file
  - Create a URL object given the URL string
  - Get a URLConnection from the URL object
  - Get an InputStream from the URLConnection object
  - Read the bytes from the InputStream
- In this case we want to read the bytes into a String object
- Then display the String object in the WebView

Android Developer Example (1)

- NetworkUsage.zip (more complex than my example)
- `http://developer.android.com/training/basics/network-ops/connecting.html`
- Set permissions
- Choose HTTP Client
  - Commonly used for Internet
  - Could use raw sockets for WiFi direct

Android Developer Example (2)

- Check Network Status
  - No point proceeding if no network
- Perform network operation on separate thread: use an AsyncTask
  - Connect and download (or upload) data
- Do something with the data
  - In this case: parse XML and reformat as HTML in WebView

Sample AsyncTask

```
// Sample implementation of AsyncTask.
    private class DownloadXmlTask extends AsyncTask<String, Void, String> {
        protected String doInBackground(String... urls) {
            try {
                return loadXmlFromNetwork(urls[0]);
            } catch (IOException e) {
                return getResources().getString(R.string.connection_error);
            } catch (XmlPullParserException e) {
                return getResources().getString(R.string.xml_error);
            }
        }

        @Override
        protected void onPostExecute(String result) {
            setContentView(R.layout.main);
            // Displays the HTML string in the UI via a WebView
            WebView myWebView = (WebView) findViewById(R.id.webview);
            myWebView.loadData(result, "text/html", null);
        }
}
```

Simple Sharing with Intents

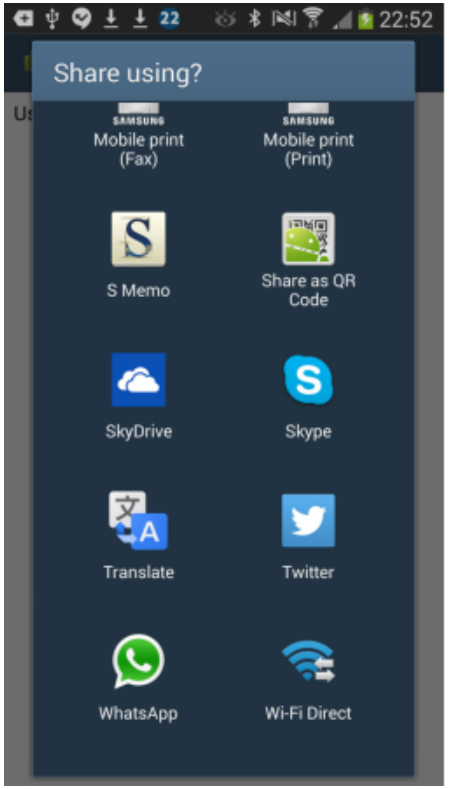
- Can use an Intent.ACTION_SEND to send to a recipient
- Recommended way is to use a Chooser dialog with this
- However, the effects are a bit random, and work better for some recipients than for others
- E.g. The code shown next work fine for email, texting and Bluetooth, but not so well for Facebook
  - In the case of Facebook, the message body was lost!
  - Still: a good method when you want to user in the loop

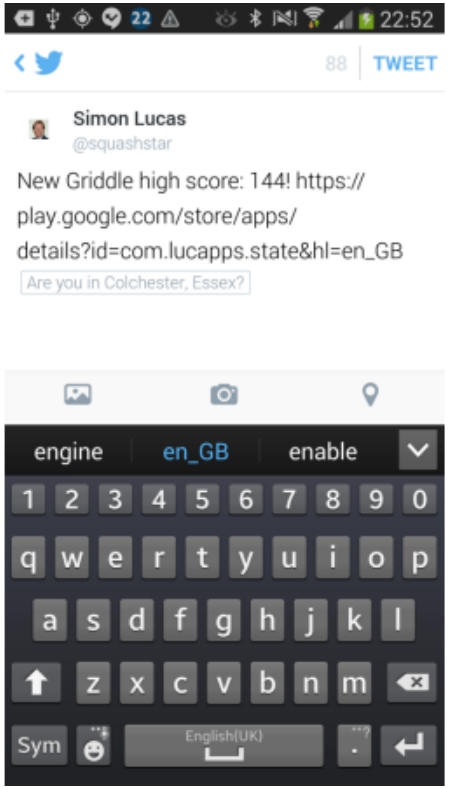
Intent sharing code: Intent.ACTION_SEND

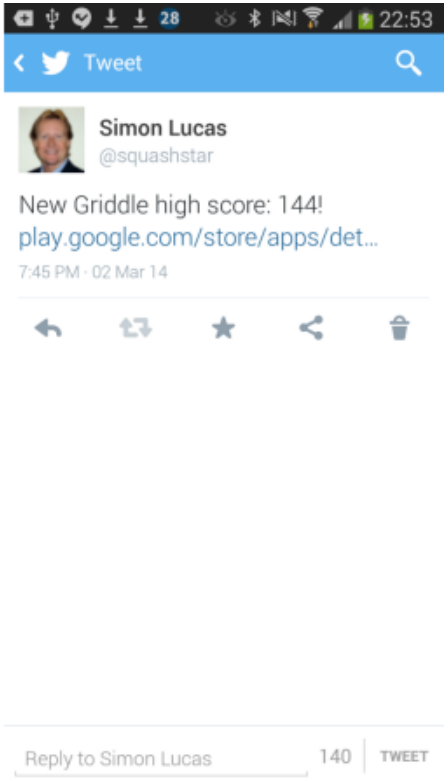
(example uses Score class from previous lecture)

```
public void shareHighScore(Score score) {
    Log.i(TAG, "Trying to share " + score);
    String message ="CE881, new high score: "
            + score.person + " : " + score.score;
    Intent sendIntent = new Intent();
    sendIntent.setAction(Intent.ACTION_SEND);
    sendIntent.putExtra(Intent.EXTRA_TEXT, message);
    sendIntent.setType("text/plain");
    startActivity(Intent.createChooser(
            sendIntent, "Share using?"));
}
```

This pops up a chooser dialog







Sharing with Intents Contd.

- Also possible to specify a particular App to process the Intent
- Intents provide a very easy way to interact with other apps
- Also possible in many cases to exploit a social networking site's API e.g. Facebook or Twitter ...
- This allows deeper access

WiFi Direct

- Can make an Ad Hoc network using Android and other WiFi enabled devices
  - No need for Internet connectivity or separate WiFi router
  - Available from API Level 16
  - Should be excellent for local multi-user apps
- Quizzes
  - Multi-player games
  - Unfortunately many devices or OS version seem plagued with difficulties: makes resulting usage difficult and unpredictable
  - Based on my experience and from Stackoverflow
- More details, see here + .zip file:
  - `http://developer.android.com/guide/topics/connectivity/wifip2p.html`
- Note: might be broken!

WiFi Direct Main Steps

- Specify permissions in manifest file
- Provide an implementation of a BroadcastReceiver
  - This will listen for events relating to the availability of peers
- Handle events for:
  - Discovering peers
  - Selecting and then connecting to peers
- One peer then normally acts as a Server
  - Sets up ServerSocket and listens for connections
- The other as a Client
  - Connects to the server socket
  - Exchanges information with server over the streams from the sockets

Facebook SDK for Android

- http://developers.facebook.com/docs/reference/androidsdk/
- Provides "frictionless" integration of Facebook within your Android app
- Study sample apps in Facebook SDK
- Enable apps that
  - Post on your behalf
  - Arrange games with friends
  - Check on status of Friends
  - Import their pictures and other data
- Exciting possibilities

Facebook SDK Key Concepts

- UiLifecycleHelper
- used to help manage lifecycle transitions
- Session:
  - Handle a login to Facebook
  - Use this to post messages etc
- For some code:
  - See Facebook SDK, getting started
  - http://developers.facebook.com/docs/getting-started/facebook-sdk-for-android/3.0/

Twitter SDK

- `https://dev.twitter.com/twitter-kit/android`
- Automate your tweets?

Publishing on Google Play

- Need a developer account
  - Easy when you have a gmail account
  - But costs $25 USD to register
- Need a launcher icon
  - Including a 512 x 512 version
  - Plus four smaller ones for various device resolutions (auto-created by Eclipse/IDEA)
  - At least two screenshots
  - Some descriptive text
- Category
  - E.g. Multi-arm bandit is Cards & Casino

Signing the App

- Need to use keytool in order to create a private key to sign the app with
  - This is often accessible from within the IDE
- Caution
  - Keep track of this somewhere safe
  - If you need to upgrade the App you MUST SIGN IT WITH THE SAME KEY!!!
- Create the Signed APK file
- This can be done from within the IDE
- Upload it via the developer dashboard
- Upgrading existing apps can be done quickly
- After clicking "Publish" it can take a few hours to appear on Google Play!

A good icon is also important ...

- http://developer.android.com/guide/practices/ui_guidelines/icon_design.html
- Which are your favourites? Read up on the style guidelines (see link above)

Google Analytics API for Android

- https://developers.google.com/analytics/devguides/collection/android/v2/
- Monitor app installs
  - The number of active users using an app
  - Where in the world they are using it
- Monitor within app
  - Adoption and usage of specific features
  - In-app purchases and transactions
  - The number and type of application crashes
- Many other useful metrics ..

Before using

- Need a Google Analytics Account
- Takes a couple of minutes to set up if you already have a gmail account
- And the .jar file for Google Analytics in the libs directory of your app
- The API is straightforward

Examples will use two tracker classes

- EasyTracker
  - Monitor app start up and shutdown
  - Good for tracking number of active users
- Tracker
  - More flexible
  - Need to get an analytics instance first
  - Then get the tracker
  - Can now track specific events
    - And how long they take

Enable Tracking of Active Users

- Add EasyTracker calls to onStart() and onStop()

```
@Override
public void onStart() {
    super.onStart();
    EasyTracker.getInstance().activityStart(this);
}

@Override
public void onStop() {
    super.onStop();
    EasyTracker.getInstance().activityStop(this);
}
```

Button Presses

- Can go in to any code
- This example is in onClick
- Uses myTracker.sendEvent(String category, String action, String label, Long value)
- Use these values to encode something that can be usefully analysed later

```
public void onClick(View v) {
    //Where myTracker is an instance of Tracker
    myTracker.sendEvent("ui_action",
                        "button_press",
                        "play_button",
                        opt_value);

    ... // Your other click handling code.
}
```

Timing events

- Example – loading a resource
- Measure the time taken (e.g. using System.currentTimeMillis())

```
public void onLoad(long loadTime) {
  // Where myTracker is an instance of Tracker.
  myTracker.sendTiming(loadTime, "resources", "high_scores", null);
  ... // The rest of your onLoad code.
}
```

Social Interactions

- Monitor usage of social widgets from within an app
- E.g. suppose you've added a Tweet button
- Can then track its usage

```
// Get tracker object.
Tracker tracker = EasyTracker.getTracker();
// now tweet e.g. using an intent
// now send social interaction to Goolge Analytics
tracker.sendSocial("Twitter", "Tweet",
    "https://developers.google.com/analytics");
```

Marketing Campaign Measurement

- https://developers.google.com/analytics/devguides/collection/android/v2/campaigns
- Google Play Store Campaign Measurement See which campaigns, websites, and apps referred a user to your app's Google Play Store page to download your app
- Measuring referrals
- See which referring traffic source, such as websites or other apps, launched your app after it was installed.

Main Idea

- When launching a marketing campaign use some specific keywords
- When launching the app (i.e. in onCreate())
  - Make an EasyTracker
  - Request the URL of the referring App
  - Send an event to Google Analytics

Google Play Store Campaign Measurement works

- A user clicks on a link, from an ad, website, or app, that takes them to your app's Google Play Store page. The link is tagged with Campaign Parameters.
- After the user downloads and installs your app, the Google Play Store will broadcast an INSTALL_REFERRER intent on the device that includes those same campaign parameters.
- Your app will then respond to that intent, using a BroadcastReceiver object
- App reads the campaign parameters and using them to update the Google Analytics campaign information.

Monetization: Making Money from Your App

- Need a Google merchant account
  - Get this using your gmail account, plus Company Info or credit card
- Three options
- IAP: In App Purchase – buy tokens, level up etc
  - Involves a financial transaction API from Google
- Sell the game: simply fix the price when submitting to Google Play
  - An initially free app (identified by name) can never be changed to a paid one
  - Instead release two version, one "Free" and on "Paid" in the title
- In App Advertising
  - Use the Google AdMob API

AdMob Advertising API

- `https://developers.google.com/mobile-ads-sdk/docs/admob/fundamentals`
- Straightforward to use
- But need an AdMob account
- Requires providing some financial information such as VAT code, Tax Code
- Here we'll just consider the sample Banner app (.zip file from above URL)
- However, more sophisticated adverts are possible, such as Interstitials

Main Steps: in onCreate

- Find a place in your layout for a banner add
  - Often at top or bottom of the app
- Create an AdView (a subclass of View)
- Add an event listener to it
  - Listen for advertising events
- Add it to the layout
- Create an AdRequest
  - If testing, then add test devices to adRequest
- Add adRequest to adView

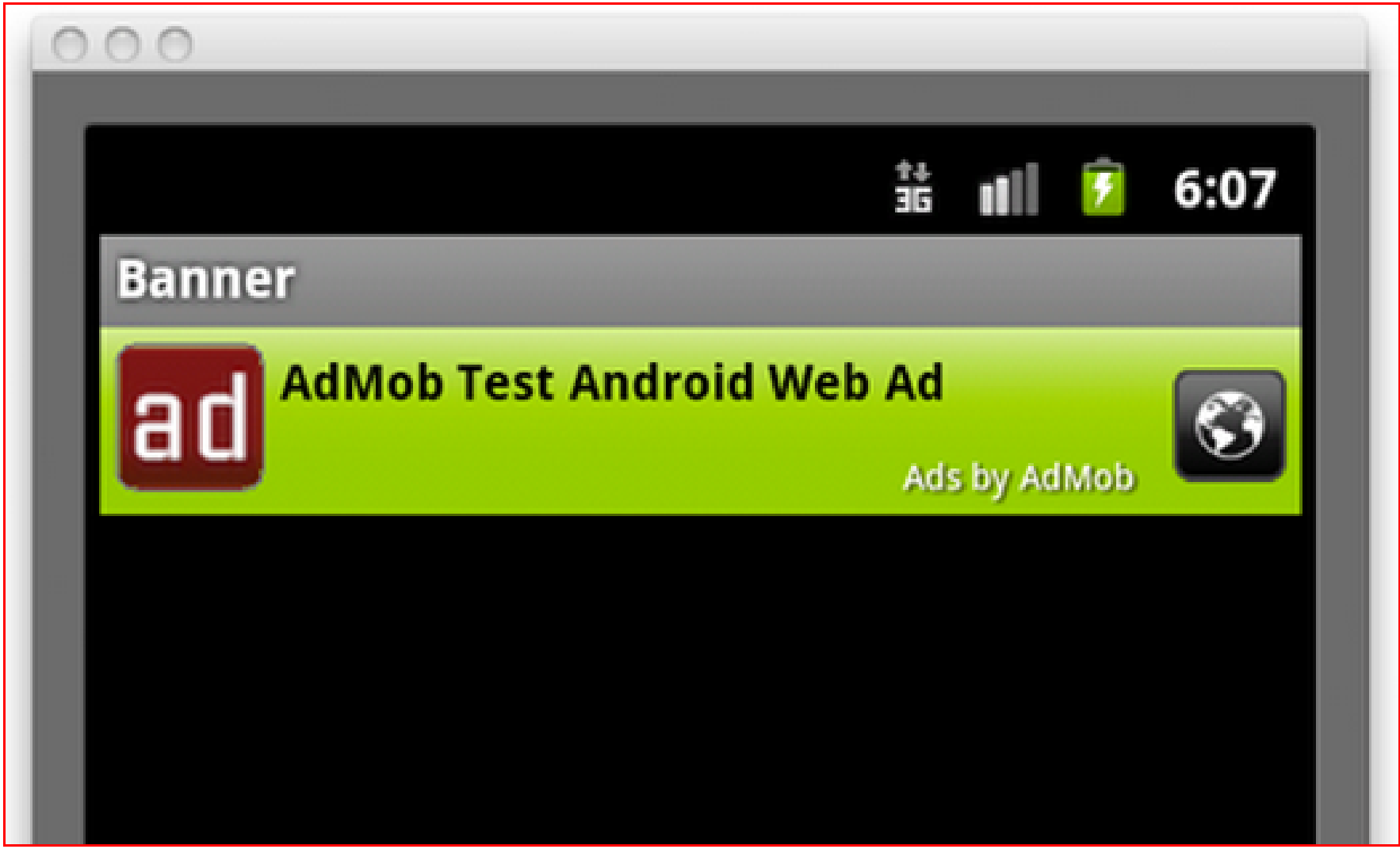
Requesting Test Ads

- Useful for checking correct use of the API before publishing

```
AdRequest adRequest = new AdRequest();
// Emulator
adRequest.addTestDevice(AdRequest.TEST_EMULATOR);
adRequest.addTestDevice("TEST_DEVICE_ID");
```

Sample Banner App

- May need an AdMob id in order to run properly



onFailedtoReceiveAd()

```
/**
 * Called when an ad was not received.
 */
@Override
public void onFailedToReceiveAd(Ad ad,
                                AdRequest.ErrorCode error) {
    String message = "onFailedToReceiveAd (" + error + ")";
    Log.d(LOG_TAG, message);
    Toast.makeText(this, message, Toast.LENGTH_SHORT).show();
}
```

onReceiveAd()

```
/**
 * Called when an ad is received.
 */
@Override
public void onReceiveAd(Ad ad) {
    Log.d(LOG_TAG, "onReceiveAd");
    Toast.makeText(this, "onReceiveAd",
            Toast.LENGTH_SHORT).show();
}
```

onLeaveApplication()

```
/**
 * Called when an ad is clicked and going
 * to start a new Activity that will
 * leave the application (e.g. breaking out
 * to the Browser or Maps
 * application).
 */
@Override
public void onLeaveApplication(Ad ad) {
    Log.d(LOG_TAG, "onLeaveApplication");
    Toast.makeText(this, "onLeaveApplication",
            Toast.LENGTH_SHORT).show();
}
```

onPresentScreen()

```
/**
 * Called when an Activity is created in
 * front of the app (e.g. an
 * interstitial is shown, or an ad is
 * clicked and launches a new Activity).
 */
@Override
public void onPresentScreen(Ad ad) {
    Log.d(LOG_TAG, "onPresentScreen");
    Toast.makeText(this, "onPresentScreen",
            Toast.LENGTH_SHORT).show();
}
```

Events to listen for: onDismissScreen

```
/**
 * Called when an ad is clicked and
 * about to return to the application.
 */
@Override
public void onDismissScreen(Ad ad) {
    Log.d(LOG_TAG, "onDismissScreen");
    Toast.makeText(this, "onDismissScreen",
            Toast.LENGTH_SHORT).show();
}
```

Summary

- Easy to publish apps – takes an hour or two (or much more if you have to design an icon!!!
- "Analytics" can be used to track the behaviour of an app as it interacts with a user
- Can use it to check that App works properly, including automatic bug reports
- Even more interesting
- Use it to analyse user choices
  - Use of levels etc.
- User demographics
- Alternative ways of doing this
- Google Analytics API
- Can also roll your own – need a own web server
- Also note Campaign Management and AdMob API

Clash of Titans

- http://www.gamasutra.com/blogs/WolfgangGraebner/20140402/214504/Clash_of_Clans__Time_Monetization_Formulas_Demistifyed.php
- Monetise time
  - Players get bored of waiting for certain game mechanisms
  - Players can skip time
- "Coerrsive Monetisation ?"

Candy Crush

- http://www.thedrum.com/opinion/2013/08/07/keys-candy-shop-how-candy-crush-offers-masterclass-mark
- $633k a days (!?)
- Pay or Market
  - Limited lives at each level
  - You need to wait OR
    - Ask friends for lives
    - Pay
- 90% of users never pay

What's up Messenger

- http://finance.yahoo.com/news/facebook-plans-monetize-whatsapp-large-162853681.html
- Build a really large userbase
- Collect their habits
- Sell users stickers (!?)
- Sell an premium version of the app
- Sell your business to a third party

Worms 3

- Take an old classic title
- Packet it for android phones
- Sell it
- Profit (!?)

Puzzle and Dragons

- http://www.gamasutra.com/blogs/RaminShokrizade/20130626/194933/The_Top_F2P_Monetization_Tricks.php
- Reward Removal
  - Give something to the the user
  - Take it back if they don't pay
- Fight through a dungeon
- If you don't "pay to win" a final boss, lose whatever you collected

Podcast Addict

- Ads at the bottom of the screen
- Very common model (and not that aggressive)
- Paid app actually a donation
- No hidden gems

Thank you!

- Android offers a wide range of connection possibilities
- Internet, WiFi, Bluetooth, Nearfield
- Some of these are very easy to use
- Also third party offerings such as Facebook and Twitter offer access to sophisticated API
- And vast network of social data
- **Polish your final app as much as you can**
- Some slides from Simon Lucas
- Thank you!