# Handset Development

## Introduction

- Quick Survey, are you comfortable with…
  - Java
  - .NET
  - Objective-C / Cocoa
  - C
  - C++
- Every platform is still relevant today

## General Thoughts

- Handset development is awesome!
- Debugging is super painful
  - Emulator != device
  - There is no console (generally)
- Handsets are more buggy than desktops
- “Bleeding Edge” hurts (and changes a lot)
- Handset experience doesn’t generalize

## Summary

Play to your strengths or be willing to work hard to catch up.

## Philosophy of Mobile Development

- NOT just porting a desktop application
- Many new constraints
  - Battery life
  - Environmental
  - User Interface / Form Factor
- Platform often dictates architecture

## iPhone

- Language
  - Objective-C
  - C/C++
- Why?
  - Sexy new device
  - Easy to deploy your app (to the world)
  - Fairly standard and powerful devices
  - Hot market, full of early adopters, blah blah blah
  - Powerful API / Framework

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam

## iPhone

- Why Not?
  - New buggy platform
  - Restrictive SDK
  - Manual memory management
  - Fairly small market
  - NDA, limited support
  - No IMS support

## iPhone

- Workflow
  - Centers around Xcode, gdb, and Interface Builder
  - Initial setup is a headache
  - Application distribution is not very timely
  - Not bad, could be much better
  - A lot to learn for non mac developers

## Android

- Language
  - Java, tweaked

- Why?
  - Big backers (OHA)
  - Java based, fairly friendly
  - Muti-phone / vendor / open-ish

## Android

- Why not?
  - No devices until (earliest) mid-September + delays
  - Java based—incomplete implementation, some bugs
  - Totally inconsistent abilities... maybe
  - The SDK is a bit limited
  - Custom widgets somewhat difficult
  - No IMS support

## Android

- Workflow
  - Nifty eclipse environment
  - Good debugger
  - Emulator (as of previous SDK) can get into Weird States that don't fix themselves on reset
  - Emulator lacks some important features (like a mic!)

## JavaME

- Language
  - JavaME

- Why?
  - JME has great docs
  - Garbage collection
  - Friendly learning curve
  - Deploying to test is easier than most others
  - *Lots* of optional APIs you can use (depending on the phone)

## JavaME

- Why Not?
  - "Write once, debug everywhere"
    - 45 VMs, 600 phone variants, 2 QA engineers
  - One of the slowest solutions (in part because of the VM, in part because of the devices)
  - Unimpressive default UI toolkit
  - No local SQL db by default as in Android/iPhone
  - Deploying (to the world) is harder than iPhone / Android

## JavaME

- IMS Support
  - Ericson has a set of APIs to make SIP & IMS a bit easier
  - Ericson also provides sample code
  - Probably the best of the lot but we haven't done much with it, all our previous work was with a toolkit from NSN which is no longer maintained
- Workflow
  - NetBeans and Eclipse both provide great environments to develop in
  - Sun device emulators are pretty good (for emulators)

## Windows Mobile

- Language
  - .NET (C#)

- Why?
  - MSDN docs are generally pretty good
  - Fairly mature platform
  - Market penetration—WinMo has good coverage in enterprise environments (in the US)

## Windows Mobile

- IMS Support
  - NSN libraries
  - Reasonable docs and sample code
- Why Not?
  - Desktop shoved onto a mobile phone

## Other

- Series60
  - Low level hackery
    - Fast
    - Access to pretty much everything
    - Large learning curve
- BREW
- OpenMoko / LinMo