Mobile Application Development
Platforms: The Options

Wilfred M. Mworia

Mobile BootCamp 2009
Objectives

- Overview – focus on breadth understanding not depth (that’s what the labs are for 😊)
- Hopefully, help you choose which labs to take
- Main focus on Smartphone dev since this is what will be taught in the labs
Agenda

- Overview – why is mobile such a compelling platform
- Categorization of mobile app dev
  - Low-end platforms
  - Mid-level phones platforms
  - Smartphone platforms
Overview

The Mobile Landscape
Gartner press release (2005):

- Estimated there would be three billion mobile subscribers in the world by 2010, a doubling of current subscriber levels – most growth in Asia/Pacific (lower incomes, cheaper handsets)
- Nick Jones, Research VP and Gartner Fellow, “One of the greatest challenges CIOs will face is the proliferation of wireless devices… “
- Gartner's annual survey of 1400 CIOs showed that wireless is a top three technology priority for European CIOs in 2005.

http://www.gartner.com/it/page.jsp?id=492131
• Gartner (more recently, 2008)
  ◦ 23% of workers will use only a mobile phone in 2012, up from about 4% today.
  ◦ Predicts that business users with both a mobile phone and a desktop phone will make up the overwhelming majority but will decrease from more than 80% today to less than 70% in 2012.
  ◦ Strong trend toward corporate support of cellular services

http://tinyurl.com/qf6pyl
Africa is not left behind

- Mobile phone use in Africa is growing faster than anywhere else in the world, according to a report.
- According to a survey by Vodafone:
  - More than 85% of small businesses run by black people, surveyed in South Africa, rely solely on mobile phones for telecommunications.
  - 62% of businesses in South Africa, and 59% in Egypt, said mobile use was linked to an increase in profits - despite higher call costs.
  - 97% of people surveyed in Tanzania said they could access a mobile phone, while just 28% could access a land line phone.
  - A developing country which has an average of 10 more mobile phones per 100 population between 1996 and 2003 had 0.59% higher GDP growth than an otherwise identical country.

http://news.bbc.co.uk/2/hi/business/4331863.stm
Africa Subscriber Growth

http://www.slideshare.net/whiteafrican/mobile-phones-in-africa-picnic-08-presentation
http://www.africantelecomsnews.com/Africa_Subscriber_Data.html
Mobile Applications
Development
The Platforms
Categorization

- Categorization of mobile app dev
  - Low-end platforms
    - SMS
    - USSD
    - JavaCard & SIM Application Toolkit
  - Mid-level phones platforms
    - Java ME
  - Smartphone platforms
    - iPhone
    - Android
    - Symbian
    - Windows Mobile
    - Blackberry
    - Palm WebOS
<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Low-end</th>
<th>Mid-level</th>
<th>Smartphone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Form factor</td>
<td>Small screen, black/white or greenish screen, little memory</td>
<td>Larger screen, colored; more on board memory + expansion;</td>
<td>Larger screen; QWERTY keyboard, touch or both; huge memory + expansion;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>great screen;</td>
</tr>
<tr>
<td>Web browser</td>
<td>None</td>
<td>WAP browser</td>
<td>Advanced browser (WebKit, IE, OperaMini)</td>
</tr>
<tr>
<td>Native 3rd party apps</td>
<td>Not really</td>
<td>Java runtime; J2ME</td>
<td>Full SDK, great dev tools, native apps run by the OS</td>
</tr>
<tr>
<td>OS capabilities</td>
<td>Very basic,</td>
<td>Mid-way</td>
<td>Advanced OS; multiprogramming etc iPhone OS, WebOS, Android, WM;</td>
</tr>
<tr>
<td>Example handsets</td>
<td>Nokia 1100</td>
<td></td>
<td>iPhone, Android based (G1 etc), Windows Mobile based (Samsung Omnia, HTC), Symbian based (Nokia N-series); Palm</td>
</tr>
</tbody>
</table>
Low-end devices

Nokia 1100
Low-end device platforms

- Options:
  1. SMS based apps
  2. USSD
  3. SIM Application Toolkit

- Pros:
  - Ubiquitous (a beautiful property)
  - Not so complex
  - If you are a dev, no worries about interface design; but you must consider interaction design

- Cons
  - Very limited
  - Possible dependency on a mobile service provider
Low-end: SMS

- A lot of free and open source SMS libraries and tools out there
- Very simple:
  1. Get SMS in some expected structure
  2. Parse text for keywords
  3. Perform actions
  4. Formulate and send response
- Literally no need to be concerned over user interface issues (Yay for the dev!)
- But what about the user’s interaction with your system???
  - #OPERATION#001#[ACCOUNT NO.]
  - #GET#BALANCE#[ACCOUNT NO.]#
  - GET BALANCE [ACCOUNT NO.]
- You might need to depend on a service provider’s SMS Centers (SMSCs) or gateway like Kannel – if scaling out; optionally tools like FrontlineSMS can use a modem or phone connected to the PC (no internet needed)
Low-end: SMS Free Tools & Libraries

- **Kannel SMS & WAP gateway** - [http://www.kannel.org/](http://www.kannel.org/) connect to a mobile operator's SMS center (SMSC), or using a GSM mobile phone or GSM modem as a virtual SMS center
- **OpenSMPP API** – Java API to communicate with an SMSC [http://sourceforge.net/projects/smsstools/](http://sourceforge.net/projects/smsstools/)
- **SMSLib** – for Java, has a .NET port [http://smslib.org/](http://smslib.org/)

[http://www.developershome.com/sms/freeLibForSMS.asp](http://www.developershome.com/sms/freeLibForSMS.asp)
Low-end: USSD

- USSD = Unstructured Supplementary Service Data
- Think of it a bit like instant messaging
- (Unlike SMS) no store and forward (no SMSC)
- User sends a code which fires an action which returns a response (maybe in the form of an SMS)
- E.g. *144# - gets balance on safaricom, *141#458789651236# take the 12 digit code following the USSD code as input and do something
Low-end: Java Card & SIM Application Toolkit (SAT)

- How does MPESA work?
- The application resides on the SIM card
- Enables menu based interaction
- Get user input in a more natural, step by step way – e.g. enter your PIN -> enter the amount....
- Java Card...
- You can use the SIM Application Toolkit to write these kind of apps...
JavaCard

- Smart cards and other devices with very limited memory to run small applications, called applets, that employ Java technology e.g. SIM cards
- Secure and interoperable execution platform
- At the language level, Java Card is a precise subset of Java
- Java Card bytecode run by the Java Card Virtual Machine is a functional subset of Java bytecode run by the JVM
- However, there are differences in the Library and Runtime e.g. Java Security Manager not supported
- Java Card 3.0 has two editions:
  - **Classic edition**: targets more resource-constrained devices that support traditional applet-based applications
  - **Connected edition**: enhanced execution environment; new VM; network-oriented features such as web apps
SIM Application Toolkit (SAT)

- GSM Standard
- A set of commands which define how the card should interact with the outside world (handset & network)
- The SIM initiates commands independently of the handset and the network
- Allows the flexibility to update the SIM to alter the services and download new services over the air
- Designed as a client-server application. On the server side, SimCard platform and the client side, phone that supports the SIM Application Toolkit
SIM Application Toolkit (SAT)
I speak SIM Toolkit
I have a STK application
I wait for your instructions
Display
1. Weather?
2. Sports?
3. Bank?
And send me back user selection

http://www.gemalto.com/techno/stk/
Mid-Level: Java Micro Edition (Java ME)

- Gray-area – Smartphones also support this, but not all phones that support this are Smartphones
- Version of Java aimed at devices with limited resources
- Highly optimized runtime environment
- **Problem:** differences in device capabilities, memory, processor
- **Solution:**
  - **Configurations:** provides the most basic set of libraries and virtual machine capabilities for a broad range of devices—
    - Connected Limited Device Configuration (CLDC);
    - Connected Device Configuration (CDC)
  - **Profiles:** For a config., a set of APIs that support a narrower range of devices e.g. Mobile Information Device Profile (MIDP)
  - **Optional Packages:** a set of technology-specific APIs; means of providing additional functionality
http://java.sun.com/javame/technology/index.jsp
http://www.sematopia.com/?p=127
Smartphone: iPhone

iPhone 3G
Smartphone: iPhone

- **Language**: Objective-C; CocoaTouch framework
- **Difficulty**: Very steep learning curve but gets much easier after that
- **Tools**:
  - **Hardware**: Mac (pretty expensive)
  - **Software**: Xcode (free); iPhone SDK
- **Pros**:
  - Very comprehensive APIs, very well documented
  - The iPhone is a great platform esp for consumer level apps
  - Ready online market place for your apps
  - Free tools (once you have the costly hardware)
- **Cons**:
  - Very different programming style
  - Expensive to get a handset to test on
  - You need specific hardware
  - Marketplace controlled by Apple; 70-30 revenue sharing; Apps have to be approved by Apple first
  - Only one device manufacturer and device type
#import <UIKit/UIKit.h>

@interface Button_FunViewController : UIViewController {
    IBOulet UILabel *statusText;
}
@property (retain, nonatomic) UILabel *statusText;
-(IBAction)buttonPressed:(id)sender;
@end

Snippet from: Beginning iPhone Development; Dave Mark, Jeff LaMarche
#import "Button_FunViewController.h"

@implementation Button_FunViewController
@synthesize statusText;

- (IBAction)buttonPressed:(id)sender
{
    NSString *title = [sender titleForState:UIControlStateNormal];
    NSString *newText = [[NSString alloc] initWithFormat:
        @"%@ button pressed.", title];
    statusText.text = newText;
    [newText release];
}

- (BOOL)shouldAutorotateToInterfaceOrientation:(UIInterfaceOrientation)interfaceOrientation
{
    // Return YES for supported orientations
    return (interfaceOrientation == UIInterfaceOrientationPortrait);
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning]; // Releases the view if it doesn't have a superview
    // Release anything that's not essential, such as cached data
}

- (void)dealloc {
    [statusText release];
    [super dealloc];
}
Smartphone: Android

T-Mobile G1
Smartphone: Android

- **Language:** Java
- **Difficulty:** Intermediate to advanced Java & J2ME
- **Tools:**
  - **Hardware:** PC or Mac
  - **Software:** Eclipse is the editor of choice; Android SDK

- **Pros:**
  - Familiar programming model
  - Very comprehensive APIs, very well documented
  - Ready online market place for your apps
  - Open source OS
  - Open Handset Alliance creates a strong backing
  - Free tools

- **Cons**
  - Handsets not readily available (at least in Africa). Google offers a developer handset (not free)
  - The platform is still young, not many Android based handsets out there; but that will change – potentially many device manufacturers Samsung, HTC etc
Smartphone: Symbian

Nokia N-series
Smartphone: Symbian

- **Language**: Mainly Native Symbian C++; Java ME, Flash & Python Runtimes
- **Difficulty**: Challenging for C++; Java ME more familiar; more control with C++
- **Tools**:
  - **Hardware**: PC
  - **Software**: Symbian Application Development Toolkit (ADT); SDKs e.g. Nokia S60 SDK
- **Pros**:
  - Familiar programming model
  - Very comprehensive APIs, very well documented
  - Ready online market place for your apps
  - Open source OS
  - Symbian Foundation backs the development of Symbian
  - Handsets are readily available
  - Many device types and manufacturers – Samsung, LG, Nokia
  - Free tools
- **Cons**
  - Needs some good experience with C++
Smartphone: Blackberry

Blackberry Storm
Smartphone: Blackberry

- **Language**: Several options: Java, .NET,
- **Difficulty**: For Java, if you are familiar with J2ME you’ll be great. .NET is by nature not so hard
- **Tools**:
  - **Hardware**: PC
  - **Software**: For Java: Plugin for Eclipse, Blackberry JDE; For .NET: Plugin for VS; MDS Studio
- **Pros**:
  - Options for programming languages and tools
  - Very comprehensive APIs, very well documented
  - Good for enterprise apps
  - Handsets are readily available
  - Free tools
- **Cons**
  - IMHO perhaps not the best for consumer apps
  - Writing apps for different Blackberry devices can get tricky; have to pick the right version of tools for the right OS version
  - Only one device manufacturer RIM
Smartphone: Windows Mobile

Samsung Omnia
Smartphone: Windows Mobile

- **Language**: Mainly .NET or C++
- **Difficulty**: Not so hard, you should be very comfortable if you are familiar with .NET
- **Tools**:
  - **Hardware**: PC
  - **Software**: Visual Studio, SDKs (Pocket PC, Smartphone)
- **Pros**:
  - .NET is not so hard
  - Very comprehensive APIs, very well documented
  - Good for enterprise apps and consumer apps
  - Handsets are readily available
  - Visual Studio will cost you, the SDKs are free
  - Lots of device manufacturers, types of devices – HTC, Samsung, i-mate etc
- **Cons**
  - It has been said that WM is on the decline due to competition from the other platforms esp. Blackberry (in the enterprise) & iPhone (consumer level) and now Android
Smartphone: Palm WebOS

Palm Pre
Smartphone: Palm WebOS

- Linux based OS
- Very new, the Palm Pre was launched earlier this year at CES.
- Developer SDK (Mojo) yet to be released. Later this year
- The Palm Pre is the only device currently running this OS
- Palm will provide an on-device application catalog to deliver your apps directly to users.
- HTML, CSS, JavaScript based applications
General Guidance to Smartphone Dev

- Pick a platform
- Learn a bit about the OS in general
  - Understand the environment in which your app will run
  - How the OS treats scenarios like multi-tasking
  - Could help with things like performance tuning
- Get the tools
- Learn the APIs
  - Learn as you go – it’s easier to pick what you need and learn it when you need it instead of trying to learn all the APIs beforehand
- Design is key for consumer apps
Thank You