CISC 3650
Human-Computer Interaction

MOBILE GAME PROGRAMMING
AND THE ANDROID PLATFORM

guest lecturer:
Matthew K. Meyer
http://www.sci.brooklyn.cuny.edu/~meyer/

Content

Project Portfolio

Why make an mobile app?

- There are really only 3 interview questions:
    1. Can you do the job?
    2. Will you like it here?
    3. Do we like you?
- For a programmer the first question actually has two parts:
    1. Do you have the requisite knowledge to program:
        1. "Tell me about inheritance (polymorphism)."
        2. "What's the difference between an abstract class and an interface?"
        3. "What the most efficient sort for a list that has at most 1 or 2 items out of place?"
    2. Can you complete a large project?
        1. "Show me what you have done before."
        2. "Do you have any examples of your work?"
- In an interview situation, being able to point to a large website, or moderate sized application that you have created is very helpful.
- A complete (simple) mobile application is a great thing to have in your portfolio.

Why make an Android app?

1. It's free to develop and deploy apps on Android mobile devices!
2. Android has a larger installed base than IOS (almost twice the installed base in the US and growing).
3. Slightly easier learning curve for SDK (Java) than for Objective-C
4. Yes, you can make money doing it:
   - http://www.electricpig.co.uk/2011/03/22/meet-the-android-app-millionaires/

Introduction

Introduction I (Devices)

- Mobile Devices (Primary Purpose):
  - Gaming Devices: Nintendo DS, PSP.
  - Music Devices: ZUNE, IPod.
  - Cellphone Devices: Nokia, Samsung
  - Web Devices: Blackberry, IPhone, PDA's
- Line between all of these devices is rapidly becoming blurred.
- For our purposes "mobile device" is network enabled device you can carry.

Introduction II (Technology)

- Modern mobile devices are small computers (simple phones == 1990 computer; iPhone 4, 1Ghz Processor, 512MB RAM).
- Signature feature of these devices is the built in network support (complex protocol suite).
- Mobile devices (in particular "smart phones") driving force behind advances in wireless communication technologies (4G = 1 Gbit/s).
- Fixed (and limited) hardware (RAM) as well as limited input, output, and display capabilities.

Introduction III (Market)

- Mobile devices have an incredible installed base. [1]
  - Over 5.6 billion mobile phones in USE on the planet.
    - World Population of 7,012,000,000
    - 79.86 % of world has a mobile phone
    - About 20% (about 1.2 billion) are "smart phones"
    - NOTE: Smartphone ownership is over 43% for U.S. mobile subscribers (Android most popular)
  - Compare that too:
    - Personal computers in the world 1-2 Billion (2 Billion by 2015).
    - Xbox 360's in the world, 50-60 million.

[1] Philippines - Telecoms, Mobile, Broadband and Forecasts". BuddeComm. 2011-10-12. Retrieved 2012-01-24.

Introduction IV (App Market)

- Worldwide Smartphone Application Market To Reach US$15.65 Billion by 2013. [2]
- According to TechNet the "App Economy" has created over 466,000 new jobs (up from 0 in 2007).
- According to CNET (2011) although the Apple Apps Market are still more profitable (4 times more profitable) the Android Market is now growing at a rate of 1 billion app downloads per month and will soon have far more App offerings.

[2] "Global Smartphone Application Market Report 2010", Ralf-Gordon Jahns, et al. 2011
[3] CNET: http://news.cnet.com/8301-13579_3-57346115-37/iphone-app-sales-kicking-app-on-android-market-says-study/

How "Apps are Implemented"

How Apps are Implemented I

- Embedded Apps:
  - Built into chipset or OS.
  - Ships with device, rarely added after.
  - Example: Snake.
- SMS Apps:
  - Piggy back on SMS system for functionality.
  - Played by sending text messages to other phones and servers.
  - Still very popular over-seas.

How Apps are Implemented II

- Compiled Apps / C Apps (C#, C++, Mobile-C, Objective-C, Bionic)
  - Written then compiled for specific system.
  - Fast, powerful, optimized applications are possible that directly access phone hardware.
  - Different vendors create application development platforms for developers to use; this allows them to have SOME control over what gets put on their devices.
  - Examples: BREW (Qualcomm), .NET (Microsoft), **IPhone SDK** (IPhone), Mophun (Oberon, mult), Android NDK (Android phones)

How Apps are Implemented III

- JAVA (and other Interpreted languages)
  - Most mobile devices support JAVA.
  - J2ME (Micro Edition) is a Java Virtual Machine (JVM) specifically optimized for mobile devices.
  - The JVM "Sandbox" makes it less important for platforms to rigorously control access to the mobile device.
  - Examples: Processing (FREE & Simple), MIDP (J2ME), ExEn, WGE, DoJa, **Android SDK**.

How Apps are Implemented V

- Browser based Apps.
  - Run using an optimized "web browser" for the mobile device.
  - Can be made in any web language (HTML, PHP, Python, Perl, JavaScript).
  - Can be made and displayed using specialized web applications: FLASH LITE.
  - Limitation has been bandwidth… thank you 3G & 4G.
  - Apples refusal to support Flash has changed this market significantly as HTML5 is still NOT in widespread use.

What's different about mobile app programming

Physical Requirements

- Team Size:
  - Conventional platform applications require large teams of 50 or more people, often working in separate and unique roles.
  - Mobile applications can be developed by groups as small as 3-5 people with individuals supporting several areas.
  - In fact mobile application developers are more likely to be required to wear multiple hats: that is design, code, debug and market.
- Budget:
  - Conventional apps have budgets in the 1.5-5 million dollar range.
  - Most mobile apps are implemented for less then $100,000.
  - Limited capabilities of the devices being designed for are actually an advantage.

Development & Deployment

- Development LifeCycle:
  - Conventional applications take on average 2-3 years to develop.
  - Most mobile apps are completed in a few months.
  - Small team, with small budget, using iterative development can create a quality game fairly quickly.
  - EVERTHING YOU HAVE LEARNED ABOUT SOFTWARE DEVELOPMENT STILL APPLIES!
    - Development Lifecycle
    - Development Methodologies
- Deployment
  - Conventional apps are (mostly) purchased in software outlets.
  - Mobile apps are (mostly) downloaded and installed.
  - Distribution channels for mobile apps included built in menus, carrier menus as well as wireless/web portals.
  - End of the CD, DVD, Blue-Ray?

Standards & Features

- Open Standards:
  - Console development requires “royalties” in order to develop games… in the mobile world, not so much.
  - Standards underlying mobile game development are published, open and available for review.
  - Profit for manufactures is in selling the hardware (the phones) and in “app stores” where they take a percentage of the sales.
- Networked Devices:
  - Mobile devices may be limited in input, output and display but they have powerful network capabilities built-in.
  - Infrastructure supporting devices can be easily leveraged for network apps, very rare to need to write any network support code.
  - The portable nature of the phones themselves makes short range wireless (blue-tooth) also an option.

Strengths & Limitations of the Medium

Strengths

- HUGE potential audience.
  - Around 1.2 billion smart phones worldwide
  - Over 73 million smartphones in the US alone
- Portability
  - Apps can be used whenever and wherever people choose.
  - Greater chance for “viral” exposure to apps where users hear about an app from other users (free marketing).
- Networked
  - Mobile devices come pre-networked.
  - Socials apps already showing tremendous promise.
  - Very unusual to have to write any extra network features.

Limitations I

- Limited Output (not just screen size).
  - Can't see an app with your fingers in the way AND Harder to get control and help information on the screen.
  - Fewer colors, refresh rates supported.
  - Sound problems (codecs, and the speakers themselves).
- Limited Application Size.
  - Limited RAM is just a fact of life and graphics add up (rule of thumb, target < 10% of RAM available.
  - Limited processing power must also be considered. Ex: How many collision checks need to be made in each frame.
  - Efficient algorithms just as important as with regular applications.

Limitations II

- Latency
  - 3G is an improvement, as is 4G where available, but latency in multiplayer games is always going to be a problem.
  - Moreover some of the processor intensive tricks (bit-packing/bit-compressions) used to handle latency in regular applications, don't translate well to mobile devices.
- Interrupt ability is crucial.
  - If the phone rings, the player better be able to stop the app and then pick up where they left off.
  - Again, most mobile languages support the "reflective" paradigm.
- Rapidly evolving technologies.
  - All of those poor saps who thought they had the mobile app market covered with BREW got dealt a really rude surprise by the IPhone.
  - Android in turn has already almost twice the installed based of IOS.

Making it Work

Keep it short and simple

- Don't attempt too much in any single application!
  - Keep the scale of the application and the time required by the application to accomplish any goal low.
    - What if they want to make a call?
    - Don't want to run down the battery.
    - If they had more time, they would choose a different platform.
- NEVER force them to wait (instant on).
  - Allow for saves, pauses, repeats, skips, etc.
- Use the network.
  - A phone is a social device.
  - How can your app tie into a persons existing social network

Understand the limitations of the platform

- Plan for the processor and RAM allotment.
  - Aim to use far less then what you think is available (5-10%).
  - Use a smart timing loop (like an update manager) to keep track of performance.
  - Allow processor heavy features (particle effects, 3D effects, complex animations) to be turned on and off (just a flag in the loop).
- Plan for the form factor.
  - Avoid designs that require a player to look at many places (in a larger world) in a short period of time.
  - Avoid making the player "switch" views often. It's best if entire world, everything user needs, can be seen on screen at once.
  - It's best if player only has to "control" one object in the world.
  - Keep the control scheme as simple as possible!!!
    - 3-4 action possibilities
    - Ask yourself: "Can I do everything with one thumb?"

Plan Ahead!!!

- Design for a business model.
  - Application sale.
  - Advertising revenue or product tie-in.
  - Trial versions.
  - One month licenses.
  - Charging for "data traffic" or "airtime".
    - This last model is increasingly popular in foreign markets, but as not yet become normative in U.S.
- Plan to support multiple devices.
  - At a minimum plan your app to support multiple screen sizes.
  - Better yet, target a large pool of devices.
  - Android and Eclipse now support tools for multiple output formats.

For More Information

IPhone

- IPHONE
  - FREE to develop, but applications must be approved and Apple takes cut (30%)
  - FREE online IPhone programming course from Stanford University:
    - http://www.stanford.edu/class/cs193p/cgi-bin/index.php
  - IPhone Developers Network:
    - http://developer.apple.com/iphone/

Flash

- FLASH LITE – Part of Adobe CS5
  - Free 30 day trial, then $300-$500.
  - Flash Lite download (mobile devices):
    - https://www.adobe.com/cfusion/entitlement/index.cfm?e=flashcdk
    - Flash Lite games can be exported as iPhone applications (Castle Crashers).
  - Flash Lite – Best Practices:
    - http://www.adobe.com/devnet/devices/articles/cryptic_capers_print.html

Mobile (Android) Processing

- Processing – A Java based scripting environment for mobile devices.
  - FREE: http://www.processing.org/
  - Learning the Processing Language:
    - http://processing.org/learning/
  - Getting started making processing Apps for Android
    - http://wiki.processing.org/w/Android
    - http://blog.blprnt.com/blog/blprnt/processing-android-mobile-app-development-made-very-easy

Android

History

- Android is a small, light-weight, Linux-based operating system for mobile devices.
- Google acquired the rights to the Android OS by buying out the company Android Inc. in 2005.
- Android is now maintained and developed by the Open Handset Alliance (a consortium of mostly hardware manufacturing firms to develop open standards for mobile devices) led by Google.
- First-ever Android device was the T-Mobile G1 released in October 2008.

Android as an OS

- Although Android is a Linux distro, it is difficult to port existing Linux applications or libraries to Android.
  - Android's uses a Linux kernel outside the typical Linux kernel development cycle (non –Debian) .
  - Android does not have a native X Window System nor does it support the full set of standard GNU libraries.
- Linus Torvalds said that "eventually Android and Linux would come back to a common kernel, but it will probably not be for four to five years".
- In general the standard folder layout is the same as other flavors of Linux and common applications are present or have an easy replacement (example: Android users have SQL-Lite for storage rather than MySQL).

Native Development Kit (NDK) in Android

- The Android NDK is a software package designed to work with the Android SDK and it allows users to build performance-critical portions of their apps in native code (C, C++).
- NOTE: Native code files/applications are still packaged into an .apk file and still run inside of a virtual machine on the device.
- The NDK is designed for use *only* in conjunction with the Android SDK (you will NEED both).
- From the Android Development Site:
  - " If you have NOT run into any limitations using the Android [SDK] framework APIs, you probably do not need the NDK. "

Android Software Development Kit (SDK)

- The Android software development kit (SDK) is set of development tools (debugger, libraries, a handset emulator sample code, tutorials, etc.) for use in creating Android applications that is supported by all major platforms (Linux, Mac, Windows).
- The officially supported IDE for the SDK is Eclipse using the Android Development Tools (ADT) Plugin, though developers may use any text editor to edit the Java and XML files in the SDK (command line compilation with Apache ANT will then be required).
- Android applications are packaged in .apk format and stored under /data/app folder on the Android OS (accessible only to root uses). APK packages contain .dex files (compiled byte code files called Dalvik executables), resource files, etc.

Brush up on your Java

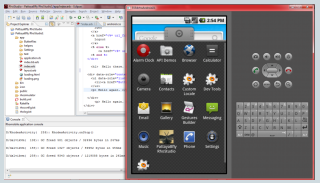
- Key feature of developing Apps for the SDK is using the event-listener classes and interfaces that are included in the SDK.
- Users create graphic objects that can "extend" event-listener classes or "implement" event-listener interfaces.
- Good Sites to Use:
  - http://docs.oracle.com/javase/tutorial/
  - http://www.roseindia.net/java/master-java/index.shtml

Android Getting Help

- Android Developers Website:
  - http://www.developer.android.com
- Basic Tutorials:
  - http://developer.android.com/resources/browser.html?tag=tutorial
- Another "Intro to Android" lecture set:
  - http://www.vogella.de/articles/Android/article.html
- Tutorial on a Database Backed Android App
  - http://hackaday.com/2010/07/12/android-development-101-%E2%80%93-a-tutorial-series/
- Even More Tutorials Covering a Range of Topics:
  - http://webdesignlists.com/top-10-best-android-tutorials-for-developers.html/

What you will be doing in the labs.

- You DON'T need an Android Mobile Device!
- The Android SDK allows you to setup and configure an emulator also know as an Android Virtual Device (AVD) on your computer.
- You will then write programs in Eclipse and run them on the AVD.
- The AVD will allow you to interact with your APP.



YOU CAN DO THIS!

SERIOUSLY,
YOU CAN DO THIS!!!