# Mobile Game Programming

Just that facts.

# First…. a story.

August, 2008: Ethan Nicholas, a Java engineer at Sun Microsystems (married, two kids) is desperate for money, and has heard that people can make hundreds, even thousands, of dollars making apps for the iPhone. Working at night, sometimes cradling his 1-year old son, he starts working on a game “iShoot”.

October 2008: iShoot is released. Over the next 4 months iShoot earns Ethan over $3,000 (@ $5/copy).

January 2009: Very satisfied with the money he has already made, Ethan releases a “free-trial” version of the game and lowers the price to $3.

http://www.nytimes.com/2009/04/05/fashion/05iphone.html

## Introduction I

- 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.

## Introduction II

- Modern mobile devices are small computers (simple phones == 1990 computer; IPhone == original XBOX).

- Signature feature of these devices is built in network support. Mobile devices driving force behind advances in wireless communication technologies.

- Limited RAM as well as limited input, output, and display capabilities.

## Introduction III

- What they lack in power, they make up for in sheer installed base. Most of the world owns at least one mobile phone.
- Worldwide market for portable and mobile games $5.4 billion in 2008. Estimate $11.7 billion by 2014.
- In the three months ending February 2010, an average of 50.9 million mobile subscribers played at least one game in the past month.
- Apple iPhone/iPod Touch are currently expected to comprise 24% of all portable game software sales, including PSP and DS by 2014.

## How Games are Implemented I

1. Embedded Games:
   - Built into chipset or OS.
   - Ships with device, rarely added after.
   - Example: Snake.
2. SMS Games:
   - Piggy back on SMS system for functionality.
   - Played by sending text messages to other phones and servers.

## How Games are Implemented II

3. C Games (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 control what gets put on their devices.
   - Examples: BREW (Qualcomm), .NET (Microsoft), IPhone SDK (IPhone), Mophun (Oberon, mult).

## How Games are Implemented III

4. JAVA (and other Interpreted languages)
   - Most mobile devices support JAVA.
   - J2ME (Micro Edition) specifically optimized for mobile devices.
   - “Sandbox” makes it less important for platforms to control access.
   - <u>Examples:</u> Processing (FREE & Simple), MIDP (J2ME), ExEn, WGE, DoJa, Android SDK.

## How Games are Implemented V

5. Browser based games.
   - Played 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.

## What’s different about mobile games I

1. Team Size:
   - Conventional platform games require large teams of 50 or more people.
   - Mobile games can be developed by groups as small as 3-5 people.
   - Ethan Nicholas working by himself, created iShoot for the iPhone in 2008. Rereleased in January, 2009 it earned him $800,000 in 5 months.

## What's different about mobile games II

2. Budget:
   - Conventional games have budgets in the 1.5-5 million dollar range.
   - Most mobile games are implemented for less then $100,000.
   - Limited capabilities of the devices being designed for are actually an advantage.

## What’s different about mobile games III

3. Development LifeCycle:
   - Conventional games take on average 2-3 years to develop.
   - Most mobile games are completed in a few months.
   - Small team, with small budget, using iterative development can create a quality game fairly quickly.

## What’s different about mobile games IV

4. 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 games.
   - Portable nature makes short range wireless (blue-tooth) also an option.

## What's different about mobile games V

5. 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.

## What's different about mobile games VI

6. Deployment
   - Conventional games are (mostly) purchased in software outlets.
   - Mobile games are (mostly) downloaded and installed.
   - Distribution channels for mobile games included built in menus, carrier menus as well as wireless/web portals.

## Strengths of the medium I.

1. HUGE potential audience.
   - Over 2 billion mobile phones in use today (More people own mobile phones then computers).
   - Almost ALL new phones coming on the market support JAVA applications.
   - Almost every mobile device manufacture (except Apple) has agreed to support Adobe Flash Player on all of their mobile devices.

## Strengths of the medium II.

2. Portability
   - People like to play whenever and wherever they choose.
   - Greater chance for “viral” exposure to games.

## Strengths of the medium III.

2. Networked
   - Mobile devices come pre-networked.
   - Multiplayer and “social” games already showing tremendous promise.

## Limitations of the medium I

1. Limited Output (not just screen size).
   - Touch screens are cool, but you can’t play a game with your fingers in the way.
   - Harder to get control and help information on the screen.
   - Fewer colors, refresh rates supported.
   - Sound problems (codecs, and the speakers themselves).

## Limitations of the medium II.

2. Limited Application Size.
   - Limited RAM is just a fact of life and graphics add up.
   - Limited processing power must also be considered. Ex: How many collision checks need to be made in each frame.

## Limitations of the medium III.

3. Latency
    - 3G is an improvement, but latency in multiplayer games is always going to be a problem.

## Limitations of the medium IV.

4. Interrupt ability is crucial.
   - If the phone rings, the player better be able to stop the game without getting killed.
   - Application must be able to pause and recover, without crashing or causing the player to “lose” something.

## Limitations of the medium V.

5. Rapidly evolving technologies.
   - All of those poor saps who thought they had the mobile game market covered with BREW got dealt a really rude surprise by the IPhone.
   - Flash-Lite and Android in turn may turn out to be devastating to the 24% prediction regarding Apple.

## Making it Work I.

1. Short Play Times.
   - Short levels, short games.
   - 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.

## Making it Work II.

2. Let people play on their schedule.
   - NEVER force them to wait.
   - Allow for saves, pauses, repeats, skips, etc.
   - One frustrating level, or bad save, or slow load and they may never play again.

## Making it Work III.

3. Use the network.
   - A phone is a social device.
   - At minimum allow the saving and posting of high scores.
   - Multiplayer modes (if you can overcome latency) are a really good (and increasingly popular) choice.

## Making it Work IV.

4. Plan to support multiple devices.
   - At a minimum plan your game to support multiple screen sizes.
   - Better yet, target a large pool of devices.
   - Flash-Lite, and Eclipse are both now supporting tools for multiple output formats.

# Making it Work V.

5. 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 can be seen on screen at once.
   - It’s best if player only has to “control” one object in the world.

## Making it Work VI.

6. Plan for the processor and RAM allotment.
   - Aim to use far far less then what you think is available.
   - Use a smart timing loop (like an update manager) to keep track of the actual speed of your game and make adjustments.
   - Allow processor heavy features (particle effects, 3D effects, complex animations) to be turned on and off.

## Making it Work VII (cont).

7. 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 the U.S.

## For more information:

- IPHONE
  - FREE to develop, but applications must be approved and Apple takes cut.
  - 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/

## For more information:

- FLASH – Part of Adobe CS4
  - Free 30 day trial, then $300.
  - 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

## For more information:

- Mobile Processing – A Java based scripting environment for mobile devices.
  - FREE: http://mobile.processing.org/
  - Learning the Processing Language:
    - http://processing.org/learning/
  - Learning the Mobile libraries:
    - http://mobile.processing.org/learning/

## For more information:

- Android: FREE
- Two ways to develop:
  - Natively: C library (known as Bionic).
  - Android SDK (Java language subset)
    - http://developer.android.com/index.html
    - Includes emulator
    - Numerous major problems with A-SDK.
    - Predicted by 2012 Android 2nd most popular smartphone platform: (1. Nokia's Symbian; 3. iPhone, 4. Win Mobile, 5. Blackberry. )