Mobile Game Programming

Just that facts.
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.
Mobile devices have incredible installed base. *International Telecommunications Union* states that:

- “Most of the world owns at least 1 mobile phone.”
- 4.6 billion subscription accounts, worldwide in 2010 (5B next year; World Pop 6.6 Billion)
- Estimated that 1.3B new phones will ship in 2010, 250-300 million of them “smart phones”.

Compare that too:

- Personal computers in the world 1-2 Billion.
- Xbox 360's in the world, 40-50 million.
The worldwide smartphone app market was worth $2.2 billion dollars in just the first six months of 2010.

Worldwide mobile applications market is on track to be worth $17.5 billion by 2012.


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.
Embedded Games:
- Built into chipset or OS.
- Ships with device, rarely added after.
- Example: Snake.

SMS Games:
- Piggy back on SMS system for functionality.
- Played by sending text messages to other phones and servers.
Compiled Games / 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), Android NDK (Android phones)
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.
- **Examples:** Processing (FREE & Simple), MIDP (J2ME), ExEn, WGE, DoJa, Android SDK.
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 & 4G.
What's different about mobile games.
What’s different about mobile games I

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.
- Mobile game developers are more likely to wear multiple hats: that is design, code, debug and market their games.
continuing with our story...

- Ethan Nicholas working by himself, created iShoot for the iPhone in 2008.
- Rereleased in January, 2009 it earned him $800,000 in 5 months (total $1.1 million).
- At one point the game was earning him $22,000/day.
- iShoot's design, gameplay, development and marketing present a case study on what to do when creating a mobile app.
What’s different about mobile games II

Budget:

- Conventional games have budgets in the 1.5-5 million dollar range.
- Most mobile games are implemented for less than $100,000.
- Limited capabilities of the devices being designed for are actually an advantage.
What’s different about mobile games III

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

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

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 the hardware and in “app stores”.
What’s different about mobile games VI

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.
- End of the CD, DVD, Blue-Ray?
Strengths of the medium
Strengths of the medium I.

HUGE potential audience.

- Over 4.6 billion mobile phones in use today (More people own mobile phones than 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.
Portability

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

- Mobile devices come pre-networked.
- Multiplayer and “social” games already showing tremendous promise.
- Very unusual to have to write any extra network features.
Limitations of the medium I

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.

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.
- Efficient algorithms just as important as with console/computer games.
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 used to handle latency in console/computer games, don't translate well to mobile devices.
Limitations of the medium IV.

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.
- Again, most mobile languages support the "reflective" paradigm.
Limitations of the medium V.

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...
Making it Work I.

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.
- If a game is popular, you can always make a sequel.
Let people play on their schedule.

- NEVER force them to wait (instant on).
- 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.

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.
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.

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.

Plan for the processor and RAM allotment.

- Aim to use far far less then what you think is available (10-20%).
- 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 (just a flag in the loop).
Making it Work VII (cont).

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.
For more information...
For more information:

**IPHONE**

- **FREE** to develop, but applications must be approved and Apple takes cut (30%)
- **FREE** online IPhone programming course from Stanford University:
- IPhone Developers Network:
For more information:

- **FLASH – Part of Adobe CS4**
  - Free 30 day trial, then $300.
  - **Flash Lite download (mobile devices):**
    - Flash Lite games can be exported as iPhone applications (Castle Crashers).
  - **Flash Lite – Best Practices:**
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: NDK, C library (known as Bionic).
  - Android SDK (Java language subset)
    - Includes emulator
    - Predicted by 2012 Android 2nd most popular smartphone platform: (1. Nokia's Symbian; 3. iPhone, 4. Blackberry, 5. Winmobile.)