Exemplar Mobile Phone Mathematics Apps

Mike Pearson
gmp26@cam.ac.uk

June 30, 2011

Contents

This document is is heavily hyperlinked and is best read electronically.

1 Proposal

1.1 Context

NRICH designs, develops and produces web-based mathematical enrichment activities for students of all ages and abilities from 4 to 19. In response to a request from Prof. Keith Moffat, the team will scope, adapt, design, develop and evaluate a small selection of these as exemplar mathematical activities suitable for mobile phone technology.

1.2 Aims

1. Evaluate mobile development technologies that will allow the development and publication of mathematical resources across as wide a range of smartphones as possible. We will look at the issues involved in providing for the following mobile operating systems: Apples IOS, Googles Android, RIMs Blackberry, Nokia Symbian and Windows Phone.

2. Develop **m-nrich**: A web application to be installed on the home page of a smartphone which allows easy access to a collection of published mobile maths resources. (end March)

3. Develop a collection of templates for these mobile maths resources to include:

   (a) An interactive resource published within m-nrich

   (b) A text and image based problem

   (c) A standalone interactive resource that can be installed as an application.

4. Develop and document procedures for authors wishing to publish resources that follow these templates. (mid May)

5. Provide examples of each template. (end June)

2 Smartphone Development - a taster

This project has concentrated on smartphones - phones which have a touch sensitive screen and do a lot more than make telephone calls.

2.1 A short history of smartphones

Apple, hitherto a computer manufacturer, launched the iPhone in 2007. It's important to understand that the iPhone launch was a huge shake up for the traditional mobile phone vendors. It was an innovative device supported by an innovative business plan. The phone supported apps developed by third parties who could make money by selling through Apple's App store. It was an immediate success. The wikipedia article is a good introduction.

The iPhone very much defined the smartphone concept. In 2011, 4 years after its launch, there are now many competitors offering their own interpretations of the Apple device, its business model, its user interface, and its software. Competing smartphones copy the iPhone concept, but of course differ greatly in implementation detail.

The leading competitor is Google. Google has developed an alternative mobile operating system called Android, based on Linux, and made it available to 3rd party device manufacturers such as Samsung, HTC and Motorola. Apple's operating system in contrast is only available on Apple mobile devices: the iPhone, the iPad, and the iPod Touch. In the US, total numbers of Android phones sold has now overtaken total iPhone sales. Of course this still leaves Apple doing very nicely as it is still the biggest smartphone vendor.

Other manufacturers are a long way behind. Microsoft has launched a Windows phone, but to date has a very small market share (around 5%). Nokia, the leading but rapidly declining vendor of traditional mobile phones, is giving up on its Symbian operating system and is opting for Windows phone software.

RIM is also suffering. It's Blackberry device gained market share because of business users who needed a good keyboard for emails, and good integration with corporate networks. However they are relatively unprepared for the much larger consumer smartphone market. RIM have recently launched the Blackberry Playbook - an iPad like device - which does not share the same operating system as its phones. The Playbook adopts QNX - a unix based system which is more suitable for the computer-like smartphones.

Apple followed up its early lead with the iPad launch in April 2010, and has kept the pressure on competitors such as the Samsung Galaxy and the Motorola Xoom and the Blackberry Playbook with the iPad2 launch this year. These 'pad' devices are in essence smartphones with a big screen. The big screen removes many of the restrictions of a phone device, and new applications are emerging - particularly for e-books, magazines, newspapers and education.

2.2 What is common to smartphones?

I'll list the obvious features:

- A multi-touch screen
- GPS
- An accelerometer
- A compass (usually)
- A camera
- A media player
- SMS texting
- Wifi
- 3G mobile network
- USB connection to a host computer
- A phone. (Nearly forgot that one!)
- The microprocessor chipset - this is usually an ARM processor
- HTML5 and CSS3 capable web browsers supporting javascript (except Windows Phone?). Many mobile phone browsers are webkit based as is mobile Safari.
- A developer market place (Apple App Store, Android Market, ...)
- Developer support libraries (APIs) - functionally very similar

- Developer support web site, documentation, software development kits and phone emulation

2.3 What features vary between manufacturers?

- Screen size especially between Android phones
- Screen resolution
- Operating System
- Mobile web browsers have differing capabilities
- Development language for native code development. (Objective-C for IOS, Java for Android, Flash for Blackberry Playbook, C# for Windows phone)
- Developer support libraries (APIs) - in use very different
- The microprocessor chipset - there are many versions of the ARM processor.
- Memory capacity. Memory can be a scarce resource in smartphones.
- Developer support web site, documentation, software development kits and phone emulation - these are of course tailored to the devices and languages and APIs they.
- Legal stuff, accepted practice. App acceptance policies.
- Financial stuff.
- Application lifecycle management - packaging and updating

2.4 Native App Development

Native development means developing in the primary language supported and using the application programming interfaces directly supported by the operating system manufacturer.

For IOS, this means developing in Objective C. For Android it means developing Java code. For the Blackberry Playbook it means developing in

Flash Actionscript. For Windows phone, C# or Silverlight coding is necessary. Code developed for one platform does not port easily to another. Furthermore, native code must be tested carefully on a number of real devices as well as in emulators, and it must be updated when new devices and operating system versions are released.

These porting difficulties often lead app developers to develop for one platform first, only moving to other platforms when they know they have a successful product.

2.4.1 IOS Native Development

Apple offers an excellent set of tools to support IOS Native development. These are freely available but only run on Macintosh computers. There are many online and face to face courses in Objective C development and Objective C skills are useful in programming all Apple products. The iPhone and iPad emulators run on Mac computers providing a fast test environment that closely models the behaviour of the various devices and IOS versions.

For programmers familiar with C family languages, Objective C can be tricky to master despite being a superset of C. It uses a verbose syntax derived from Smalltalk featuring long procedure names. Perhaps the biggest hurdle is memory management. Objective C on IOS up to version 4 does not provide automatic reclamation of memory that is no longer used (a process known as garbage collection). Instead, the programmer must carefully maintain reference counts to objects in memory and ensure those objects are released when the counts drop to zero. Version 5 IOS promises some automation of this reference counting.

The penalty for getting memory management wrong is high. If memory is not released, then your application will grow its memory footprint over time, and will ultimately be killed by IOS. If memory is released before it should be, your application will crash. Apps that have poor memory management are unlikely to make it into the App Store, and if they do are likely to be down rated by users.

2.4.2 Android Native Development

Android development tools feel less supportive than their Apple counterparts. Device emulation is more difficult. The Android emulator is extremely slow to start and difficult to configure since it has to support many devices with many

different screen form factors, from many different manufacturers. Emulator slowness makes for long and tedious development cycles during testing. Most Android developers have taken to testing directly on the device.

Android uses Java, and excellent Java development support has been available for some years. Google has created plug-ins for the Eclipse development system to support Android.

Java does do its own memory management, and is therefore a more forgiving language than Objective C.

2.4.3 Windows Phone Native Development

The Windows phone requires C# for native development but also offers Silverlight for graphics intensive apps.. I have not investigated further.

2.4.4 Blackberry Native Development

The native development language for Blackberry phones is C++, with support for Java. On the Playbook, Flash Actionscript is the native language. There is an emulator for the Playbook which runs in a VMWare window. It works most of the time.

2.5 Cross Platform Development

If you've read the previous section you will appreciate that any development system that can support multiple platforms is worth investigating. Unfortunately, the leading device manufacturers do little to help so we find that smaller independent companies are rushing in to fill the gap.

It may be helpful to refer to figure 1 on page 9 for an overview.

For cross platform development there are 3 main choices:

- Flash actionscript compiled against an Adobe library and the AIR runtime, and deployed using Adobe tools;
- Javascript compiled against the Titanium runtime and deployed using Titanium tools;
- Javascript running directly within the HTML/CSS3 environment of the mobile phone web browser. Here there is an option of deploying through App stores using PhoneGap/Titanium or of simply publishing the App on a website so the user can access it through a web browser.

Figure 1: Development Choices

3 Outcomes

In this project we have looked at IOS native code development, at Flash actionscript development, and at Javascript running in a web browser.

3.1 Phase 1

In the first phase of the project we created a proof-of-concept Flash actionscript App known as *FactorBasher*, based on an idea by John Mason. This is a useful animation that works well on a mobile which can easily extend through positive integers, negatives, fractions, decimals, surds, quadratics. The original appears on NRICH in resource 7382 and resource 7452.

FactorBasher was developed using a preview release of Flash Builder and has been tested on Android and on the Blackberry Playbook emulator. The preview development system did not support IOS, but now that the production development system is available we should be able to compile this code for iPhone and iPad too. Flash turns out to be surprisingly good as a cross platform development environment now that Apple have started approving apps written in Flash. Blackberry use Flash as their native language for the new Playbook. Current generation Android phones which use the ARM version 7 processor are also Flash capable but Android users must first download and install the Flash Player and the AIR runtime. I have not yet seen any figures for Flash availability on Android, but I would guess it to be around 40% for flash inside browsers and less than 10% for AIR based apps.

3.2 An intensive breather

I then took an intensive 3 day course on iphone development in Objective-C. During the course we created a native app which used GPS and the search engine Bing to locate Coffee Shops in your area. I had ideas of converting this app to one which located things of mathematical interest in your area, but I see that something close has already been done. See Wikihood on the Apple App Store. As already commented, native development for IOS requires care and expertise. However the resulting apps are noticeably faster and often better looking than those developed by other means. Use of Apple's frameworks ensures that the apps adopt the standard look and feel. Understanding IOS frameworks makes it easier to understand the apple-like frameworks offered by all the other manufacturers.

3.3 Phase 2

In the final phase of the project I concentrated on developing web apps which can be published through a link to the app on a web site. This final phase resulted in two completed web apps and a proof of concept app for video delivery.

Mathmo is a mobile version of the A-level question and answer generator previously published on NRICH. Since the mobile version also runs on desktop browsers, and now incorporates the further maths and statistics examples, we have replaced the original with this implementation. On NRICH it can be found at resource 7088. It's direct link for mobile users is currently http://nrich.maths.org/mobl/mathmo/mathmo.html.

Mathmoka is a proof of concept app. It demonstrates the possibility of adding video help to Mathmo by embedding suitable videos from the Khan Academy.

m-nrich is a mobile friendly navigator for the whole NRICH website. Like Mathmo, it works on desktop browsers, iPads, and Android and IOS phones, but it's real strength is in providing a usable small screen interface to most of the material on NRICH.

All three of these projects were developed using Mobl. Mobl is a domain specific language under development at the University of Delft in Holland. It offers an extremely succinct way to describe apps where the bulk of the user interface is navigational. It is possible to describe screens and persistent data statically, with active controls coded in a strongly typed language that compiles to Javascript. It naturally supports AJAX - an asynchronous protocol allowing a web page to update without requiring full page reload. It is easy to support on the server simply by providing JSON (Javascript Object Notation) data to populate navigation menus.

Mobl is one of a family of domain specific languages developed by the Delft group. It is supported by a plugin for the Eclipse development which offers modular code development, syntax highlighting, autocompletion of language elements while editing, and immediate as-you-type compilation and test. The end result is a very fast and efficient development cycle.

3.4 Looking back

Flash Builder certainly provides a reliable cross platform development tool which is good for graphics intensive work. I am however disappointed with its speed. After touching a button in FactorBasher there is a small delay before anything happens. Not much, but enough for it to feel slightly lethargic. This may get better as Adobe continues to optimise the Flash library for mobiles by implementing more functions using native APIs rather than in slower actionscript. There is also considerable resistance to Flash in the Apple world, led by Steve Jobs.

Mobl is a research project and future support is not assured. However all code used to create Mobl is open source and there is a growing community of users and developers.

Other web app development tools are available which may well be more attractive to a larger project. Mobl fits with the way I like to write code, but if I were managing a team of app developers I would not choose it. Instead I would go for a commercially supported Javascript framework with complete documentation. Things are changing extremely fast in this field but the current leaders are The M-Project, Sencha, and SproutCore. Before starting a larger web app project, I would take another look at all of these.

With a larger budget capable of supporting professional software engineers, I would avoid web apps altogether, and develop native code for iPhone first, and Android second. I would expect the quality of the end result to justify the extra cost.

3.5 App templates

I have published the full source code for FactorBasher, Mathmo, and m-nrich on my GitHub site. See the deliverables section on page 13 for details. This has already proved to be beneficial as we now have an active volunteer Li-Hsuan Lung helping with Mathmo. Li Hsuan works for 8th Light based in Chicago and has Friday afternoon of company time (and some weekend time too it seems) available to dedicate to open source projects of his choosing. He is picking off issues from the Mathmo issues list. Li-Hsuan has his own fork of the Mathmo repository.

4 Accessing the App Stores

With help from Julia Hawkins (MMP) and Maggie Wilkinson (Cambridge Enterprise) we had been looking at how we may publish applications in the various vendor markets such as iTunes, the Blackberry Appstore, and Android Market. We're also in contact with Fred Lewsey who has published the Cambridge University app on iTunes. The end result of these negotiations was that there was no need for Cambridge Enterprise to be involved, and instead we should be able to sign up for these services directly with some help from Ted Krawec, a university lawyer. Julia is now following this up with Ted. Things do appear to be much simpler for the university if you offer free apps.

Apple polices its App Store in an attempt to control the quality of apps found there. It may be possible to get FactorBasher and Mathmo accepted there if wrapped in IOS packaging, but it is very unlikely that m-nrich would make it. One of Apple's stated policies is that apps should not simply provide a mobile view of one web site. Part of the reason for this is that web apps install and look exactly like native apps on IOS devices, so it is not so important to give them a native gloss.

5 Deliverables

Source code for the three examples is published on github.

- mathmo: at http://github.com/gmp26/mathmo
- m-nrich: at http://github.com/gmp26/nrichm
- FactorBasher: at http://github.com/gmp26/FB

Demos for mathmo, mathmoka and m-nrich are available at

- mathmo: at http://nrich.maths.org/mobl/mathmo/mathmo.html
- mathmoka: at http://nrich.maths.org/mobl/mathmoka/mathmoka.html
- m-nrich: at http://nrich.maths.org/mobl/nrichm/nrichm.html

A generally accessible FactorBasher demonstrator must wait until we have completed the legal necessities to allow us to submit apps to the Android marketplace. In the meantime I can use the development system to install the app on individual devices on request.

6 Evaluation

Mike Croucher at the University of Manchester maintains his Walking Randomly blog where he often reviews mathematical apps. I offered Mathmo to him requesting some critical feedback and got a detailed and very useful user report back by return. His comments have been converted to development issues and are published on the Mathmo GitHub issues list. So far Li-Hsuan and I have closed 6 out of 16 issues. Not all of these were from Mike.

M-nrich is too young to have had any serious evaluation, but people in the CMS have responded positively to it.

7 Development Resources

Apple http://developer.apple.com/

Android http://developer.android.com

RIM http://us.blackberry.com/developers/

Windows Phone http://create.msdn.com/en-US/

PhoneGap http://www.phonegap.com/

Titanium http://www.appcelerator.com/

Adobe AIR www.adobe.com/products/flash-builder.html. See also these blogs where the process is demonstrated for the current release RIAgora. For the FactorBasher App I followed Writing multiscreen AIR apps

jQTouch http://www.jqtouch.com/

jQuery Mobile http://jquerymobie.com/

Sencha http://www.sencha.com/products/touch

Mobl-lang http://www.mobl-lang.org

8 Mathematically useful frameworks, libraries and APIs

Frameworks are libraries of useful code. Here are some of the important ones for mobile apps:

MathJAX http://www.mathjax.org/ TEX and LATEX markup. We have converted the main NRICH site to use this library so TeX content on NRICH is now accessible from mobile smartphones. Mathmo makes use of MathJAX.

Flot http://code.google.com/p/flot is a useful graph plotting package. Mathmo uses it to construct curve sketching answers.

RaphaelJS http://raphaeljs.com/ provides vector graphics for mobiles capable of supporting the SVG element. Currently, this does not include the standard Android browser though it does include Firefox on Android.

EaselJS http://easeljs.com/ is a useful animation library. It makes use of the HTML5 canvas element.

Cocos2D http://www.cocos2d-iphone.org/ the leading game development framework for IOS.

9 Future Developments

The app structure represented by m-nrich looks to be very useful in creating selected views of NRICH targeted at specific audiences. It can potentially be applied to the various StemNRICH hierarchies developed by Steve Hewson, to the Curriculum Mapping documents, to themed collections of NRICH resources, and to simplified child-friendly versions of the site. It would make sense to cross-link these mini views of NRICH with the main site so they can be discovered easily.

Not all NRICH resources behave correctly in m-nrich. Over the summer Chris Clarke will be identifying these and correcting them with the aid of summer students. This is mostly a matter of identifying resources that do not scale to the small screen size correctly due to larger fixed width elements.

I think we need to go through the process of getting an app approved on one of the App stores. Possibly FactorBasher, but I'd like to see a more responsive version. I'm sorely tempted to make a native IOS app for a first submission.

Plus.maths.org could have a mobile navigator similar to m-nrich. This could be done fairly quickly, but I'm not entirely convinced it's appropriate. Articles are best read on a larger screen and for that either the standard Plus site, or an App that behaves more like the Times App would feel more natural.