Challenges to Putting the Real-time Web on Mobile Platforms

Howard Mall
Engineering and Computer Simulations, Inc.
Orlando, FL
HowardMall@ecsorl.com

John Chludzinski
Engineering and Computer Simulations, Inc.
Orlando, FL
JohnC@ecsorl.com

ABSTRACT

Today's populace is increasingly mobile. They are demanding desktop functionality from their cell phones. These platforms provide greater opportunities for people to access training especially for those who are busy and geographically dispersed. The proliferation of smartphones has put very capable pocketable computers in many hands. Blackberrys became the enterprise standard for mobile e-mail. The iPhone changed the game by creating a market for small, focused applications. Android phones are getting everybody on the internet all the time. Let us not forget Windows Mobile and Palm that created the first convergence devices called PDA's.

But, all of these different platforms make it difficult and expensive to write one application and distribute for all to use. They must be ported to different hardware, programming environments, and delivery silos. This paper will describe using web development to deliver rich applications to varying mobile platforms. The domain is synchronous training using the US-Nexus (a virtual world platform being used by a variety of federal organizations.) The approach is to allow mobile phone users to remotely take part in lectures, tutorials, and briefings with access to presentations and on-line activities in-sync with participants who are attending live or in a virtual world.

This paper describes the use of Web 2.0 techniques to deliver a very non-web mobile user experience. The discussion covers strategies for adapting to different screen sizes, designing for touch, server-side client session synchronization, and interoperability with non-web services (like messaging.) It also discusses the trade-offs for designing an effective mobile training experience focusing on a decision process for what features to implement and what to leave out.

ABOUT THE AUTHORS

Howard Mall is Vice President of Engineering at Engineering and Computer Simulations, Inc. He has spent the last five years building various kinds of training systems. He lead efforts for the Navy to develop training solutions deployed on cell phones and hand-held computers. For the Army, he delivered the Tactical Combat Casualty Care (TC3) Simulation used by combat medics to learn triage and medical decision-making on a virtual battlefield. He led the development of the Emergency Management Nexus, a next-generation synchronous training platform for the National Guard Bureau that has become a virtual world platform serving myriad federal agencies. He currently oversees multiple engineering efforts at ECS.

John Chludzinski is a senior software engineer with Engineering and Computer Simulations, Inc. He has over 20 years experience in the military simulation and training industry. Prior the joining ECS, he worked as the lead to the network architecture team for the OneTESS project were he oversaw the development of a large-scale wireless network protocol to support real-time training for the Army and Marines. Since joining ECS, his principle responsibility has been to develop of a mobile interface to the NEXUS system. This worked has focused on a web-based approach. He received his BS from the University of Florida and his MS from Rice University.

Challenges to Putting the Real-time Web on Mobile Platforms

Howard Mall, John Chludzinski
Engineering and Computer Simulations, Inc.
Orlando, FL
HowardMall@ecsorl.com, JohnC@ecsorl.com

INTRODUCTION

The proliferation of "smart" mobile phones has created a population that is increasingly connected all the time. The iPhone's "app" store created the expectation and demand that people should be able to do practically anything on their portable devices including federal training (Corbet, 2010). Through voice, texting, e-mail, and applications such as Facebook and Twitter information is literally delivered into the palm of your hand. Because of their ubiquity and availability mobile phones are great platform to deliver training. Because of their capability, "smart" mobile phones are ideal (Kolb, 2008).

But, there are several issues in designing training experiences using mobile phones:

- **Platform:** There are a number of mobile phone operating systems to be supported to have good proliferation. That includes a variety of development environments and legal requirements that must be navigated when delivering an application.
- **Interface Constraints:** Small screens and varying input methods require creative design to be functional and effective.
- **Internet Lag:** Mobile networks are not as quick or responsive as broadband.
- **Interoperability:** For inter-connected applications this can be an important issue especially if you are delivering a mobile application that duplicates functionality of and synchronizes with a desktop application.

The decision process must take into account these considerations when designing a mobile application for training. The particular application that we address is for synchronous (live-on-the-web) training that is facilitated by a traditional instructor. We have the added constraint of connecting with and duplicating a set of features of an on-line virtual world (Nexus). We will describe how we approached the design of our mobile application based on these requirements. We will then describe a general decision process extrapolated from this work. One we hope will be found useful for future developers creating training applications for mobile platforms.

BACKGROUND

The design of mobile applications is not new. The original Palm Pilot worked because it was simple and effective (Obendorf, 2008). It also had a number of developers creating applications for it years before the iPhone Appstore. It also had an excellent set of design guidelines. Now, there even more choices in operating systems and each have their own design guidelines that should be reviewed before taking on development (Jacko, 2007).

Mobile Operating Systems

We had experience with creating applications using Flash on Windows Mobile before taking on this task. It was expected at the time that Android would have Flash before the iPhone. We would develop our Nexus Mobile application on Android and still be able to support the number of medical training applications that we had already written. The popularity of the iPhone, however, was not to be discounted lightly.

Windows Mobile

Windows Mobile has now been coalesced and upgraded to the Windows Phone 7 release. Over the course of our experience and work we used Windows Mobile running on Personal Digital Assistants (PDA's). Microsoft supports development for Windows Mobile through the Visual Studio suite of tools that provide Application Programmer Interfaces (API's) in a well-known Integrated Development Environment (IDE) with which most Windows developers are familiar. The OS is open to installing software from any source without a silo much as you might purchase or write software for the Windows OS on your desktop. It is typically stylus-based but can be used with a fingertip if the buttons are made large enough.

Windows Mobile has had a version of Flash for years. Flash is a development environment that is used to create "rich web applications" through a browser plug-

in. Flash was selected so digital artists could compose highly visual applications leaving the programmers to concentrate on the logical elements. Flash was run inside of an ActiveX control on the Windows Mobile device. Using this technique, it was not a web application but was a standalone program that leveraged a web development process.

A feature of this technique was that the application could be tested within a web browser without having to install to the Windows Mobile device. It was expected that this would also make porting the application to other devices much easier as Flash was deployed on more and more platforms. However, Flash on some of the most popular mobile operating systems have yet to been problematic and in the case of the iPhone it looks like it is an impossibility (Chen, 2010).

Android

Android development was selected as the first platform to target for the Mobile Nexus. Android is fundamentally a linux kernel with phone oriented libraries. It has an application layer that can run Java applications as well as those natively compiled.

The internet search company Google develops Android. Google periodically releases the operating system as open source software. Google also supplies a Software Development Kit freely available for download and use. There is also an application store called Android Market similar to the iPhone Appstore. This is the easiest way to install applications but there are alternative ways that are free to the user.

The development environment took some time to set up. Documentation at the time was sparse making the learning curve for application development high. Skills developed for Android are focused and have very little crossover to other mobile development platforms. It became quickly apparent that becoming a productive Android programmer would take a significant amount of time.

iPhone

iPhone development requires Apple hardware, a purchased developer's license, and an iPhone software development kit for use with Apple's IDE: Xcode. Distribution of iPhone applications depends on the purchased license. With the standard license applications must be distributed through the Appstore. The Enterprise license allows applications to be distributed throughout a particular company or organization.

Devices intended for testing must be registered with the Apple iPhone developer website. Applications must also be digitally signed and deployed for those registered devices. The process takes a while to get right the first time, but becomes less onerous with practice.

The iPhone has distinctive graphical user interface components (i.e. widgets) and a unique multi-touch input system. This makes having a consistent user experience with an application developed for multiple platforms difficult. The iPhone is a singular device that drives applications to conform to a particular look and feel.

Other Platforms

The Blackberry OS, Nokia's Symbian OS, and Palm's WebOS was surveyed as part of this work but no significant programming occurred for these platforms. Blackberry offers Java and web technologies (HTML and javascript) to create applications for their OS. Nokia's Symbian has been around for a very long time and is installed on many mobile phones throughout the world. It is a C/C++ Software Development Kit (SDK) that is available for a variety of IDE's. WebOS's SDK uses HTML and javascript to create applications, but now offers C/C++ plugin development as well.

REQUIREMENTS

As part of a larger research effort, we were tasked to bring the capabilities found in the Nexus virtual environment to mobile devices. The Nexus provides 3-D environments in which multiple users may enter as 'avatars' (virtual representations of themselves) and meet and take part in training. The first Nexus use case was for on-line synchronous classrooms where multiple students could take part in virtual lectures. This use case required the implementation of:

- voice over internet protocol (VOIP) so students could listen to an instructor, ask questions, and participate in class,
- text chat as an alternative way to communicate and ask questions,
- synchronized presentation slides (i.e. Powerpoint) so instructors could share content and visuals with students as they do in a live setting,
- an attendee's list so students and instructors can see the names of who is in class, and
- class scheduling so that instructors could set up classes and students could find and attend them in the virtual world.

These tools and services were developed in addition to the geographical immersion and presence supplied by a user's avatar in a 3-D virtual world.

This use case became the driver for the development of the Nexus-Mobile research and development effort. There were many questions to be answered:

- What platforms should we target? Which ones should take priority?
- How would we have to redesign the interfaces in order to be useful on a mobile platform?
- How could we achieve interoperability with the current Nexus client/server desktop solution?
- Is 3-D viable on mobile platforms?
- But most importantly, just how far could we get in supplying a similar experience of a virtual classroom on a mobile device?

DESIGN CONSIDERATIONS

Because of the inherent constraints on screen size and computing power imposed by mobile devices several design decisions had to be considered before implementation.

3-D or not to 3-D

A survey of the platforms showed that there were nascent 3-D capabilities available on many of these devices especially the iPhone, Windows Mobile, and Android. However, there was a dearth of development tools or reasonable art and modeling pipelines available on these devices. To deliver a 3-D capability would require the development of a game engine for the selected platform. The resources required would not allow any of the application requirements to be achieved if 3-D became a priority, thus it was decided that 3-D capabilities would be addressed later as time and resources permitted.

Platform

The selection of a mobile operating system(s) to target became the most difficult design decision. Many false starts occurred before a good general approach was settled upon.

Windows Mobile has a proven development environment but its market penetration was small. Nexus Mobile would need to reach a wide audience to be successful. Windows Mobile devices no longer experience the same public enthusiasm that other platforms such as the iPhone and Android now have (comScore, 2010).

The iPhone is very popular and has a significant amount of capability. However, the public distribution of applications afforded by Apple's Appstore was seen as making the distribution of sensitive training materials problematic. The Nexus being used primarily by federal agencies was expected to require tighter control of the software and user accounts for access.

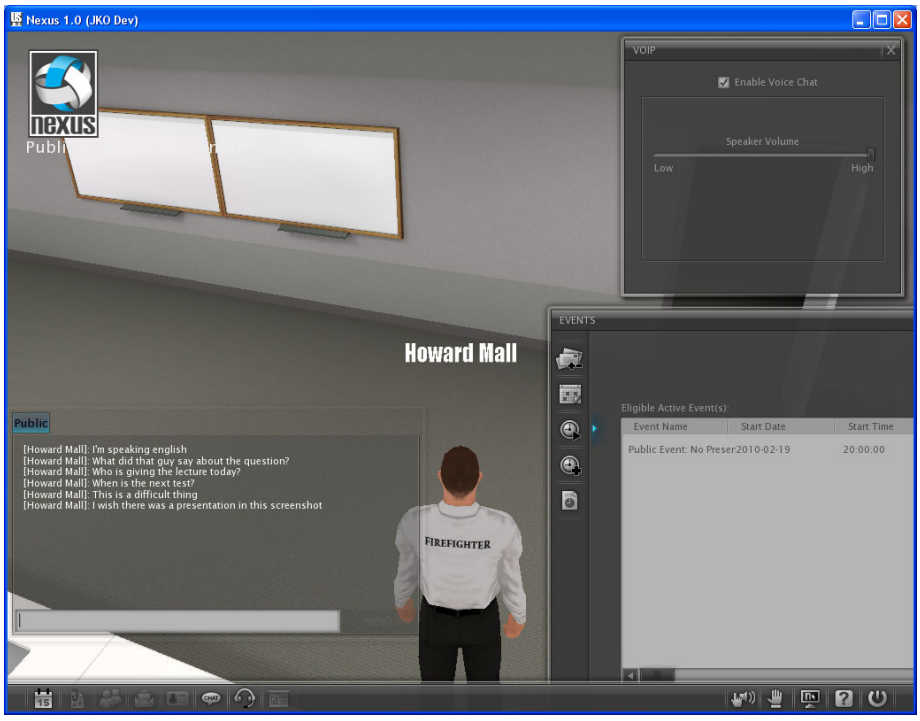

Figure 1: *Nexus Desktop with open windows*

iPhone development also required Apple desktop and laptop computers increasing development costs.

Android was seen as the next up-and-comer and a good platform for experimentation. However, the development environment and documentation at the time it was being evaluated proved immature and sparse. One developer would experience a steep learning curve that would delay release of any useful software.

Blackberry was also considered because of its ubiquity in government organizations where it used primarily for e-mail. At the time of the review, the Blackberry hardware was not very powerful and (depending on the reseller) the ability to deploy applications or access certain capabilities of the device (e.g. WiFi) took it out of the running.

Web Browser as Platform

After some experimentation, the final decision was to choose the web browser as the platform. Many of the most popular devices come pre-installed or can install from after market vendors (i.e. Opera) very capable web browsers capable of advanced features like Asynchronous Javascript And eXtensible Markup Language (XML) (AJAX). Using advanced web development technique allows for applications to approach the capabilities of desktop (or in this case palmtop) applications.

The program logic would reside on a web application server and the use of Cascading Style Sheets (CSS) would provide rendering to different screen sizes. This has several distinct advantages:

- This would cover Windows Mobile, iPhone, and Android out of the box and it was found that Opera for the Blackberry could also support the requirements.
- There is no installation of software on the device avoiding security restrictions and obviating the need for upgrade strategies.
- There is a significant amount of development environments, languages, documentation, and resources available for developing web applications.
- Features could be authored and deployed rapidly to fuel the research and development effort.

User Interface

The size of the screen is the single most important design constraint when developing applications for mobile devices. This effects what information the user can see and how they interact with it.

Figure 1 shows a screenshot of the Nexus virtual world client. There are a number of tools that can concurrently be viewed and accessed as windows that overlay the 3-D environment. A mobile screen does not have the pixel real-estate or viewable area to allow this kind of interface to be useable.

The decision was that each tool would be a single screen on the mobile device and that the interface would allow the user to rapidly switch between all of them. A learning event (virtual class) would be the context for all these screens and would tie them all together.

The user experience was designed to flow like this:

1. The user logs into the Mobile Nexus environment using their credentials
2. They select an icon that takes them to a screen listing all of the training events to which they are subscribed.
3. The user clicks on a training event and now all of their "tabs" have information related to the context of the event.
4. The attendees list now has a list of all the people in the event. These could be people using the virtual world or people like the user who are using a mobile device.
5. The presentation tab now shows a slide that the instructor has chosen to display. The user's slides stay in sync with the ones the instructor is choosing to show.
6. The chat tab is a chat room for all the attendees. They can ask questions or document key points in text communicated on this channel
7. All of these tabs should be available from any screen so the user can quickly jump between any of the tools within the context of the education event they are attending.

The discussion of the implementation of this flow resides below.

Architecture

Asynchronous Javascript And XML (AJAX) is a technique for making web applications feel like desktop applications (Malan, 2009). The most telltale sign of an old style web application is the refresh screen. The web pages take input from the user and enact transactions with the web server. A call to a particular web page supplies data to it, the web page carries out some program logic and then sends HyperText Markup

Figure 2: *Agenda Screen*

Language (HTML) to the browser to be rendered as the result. The state of the application is held by the server through a backing store or the web server's limited session context.

Old-style web applications can be a plodding and non-responsive experience for the user. AJAX provides a way to make transactions with the web application server that are transparent to the user and are processed in the background. Some of the program logic can also be shifted to the client using javascript.

The architectural design of a mobile web application must take into account these capabilities. The developer must figure out how all the different interactions with the server will be orchestrated and how results will be displayed to the user.

A functional breakdown of user interaction is useful in determining what atomic activities will be provided to the web application on the server. Thinking of web pages as remote procedure calls made by the client is a useful approach for analysis and design.

IMPLEMENTATION

Platform

The web browser is the platform, but even so we selected the iPhone as the initial device on which to test. The iPhone's browser is based on rendering code called webkit. Webkit is also the same code found in Android's stock web browser. The iPhone also has high expectations as far as capabilities due to its large number of high quality applications (Yang 2010). If the web experience can be shown to be usable and attractive then it becomes a good exemplar for the approach.

Architecture

We selected Seaside as our application server. This is a development environment based on the Smalltalk programming language geared toward web applications. It has a very robust environment for debugging. It supports the scale required. It also supports a programmatic rather than transactional model of web applications that allows multi-user synchronized sessions to be written more easily than other web application frameworks. It implements an abstraction layer using the JQuery javascript library that delivers many conveniences for web interfaces in the browser.

Interface

Figures 2, 3, and 4 show some of the screens from Nexus-Mobile. They demonstrate the design principal of do one thing and do it well. Each screen supports one tool supporting the use case of a virtual classroom.

Figure 2 shows the screen for joining a learning event (i.e. classroom). All of the eligible learning events are in a list and clicking on them does only one thing- give the user access to the event and populate their tools with the data associated with the event. Chat becomes the event's chat room. Presentation becomes the event's presentation. The attendee's list is populated with the names of the students attending the learning event.

Figure 3 shows the screen for chat. This was kept very simple with a single input, a list of posts, and send button. The button was made large enough to support a finger tap. In most all mobile web browsers clicking in the message box will invoke the on-screen keyboard.

Figure 4 shows the screen for presentation slides. Buttons were first tried for navigating the slides if you were the presenter. However, this took up too much screen real estate. Keeping it simple and taking advantage of touch, the new implementation required

the user to tap on left side of the slide to move to the previous slide and tap on the right side of the slide to advance to the next slide.

RECOMMENDATIONS

This Mobile Nexus research and development project is on-going, but with regard to mobile application design especially for federal training we can make some recommendations for others building their own mobile applications.

Build web applications unless the requirements just don't support it. If you have to do intense graphics or multimedia then you will have to write a native application. Being able to use the application off-line also precludes a web application. However, many mobile browsers are supporting HTML5 capabilities that support off-line running of web applications now.

Build a Service Oriented Architecture (SOA) to support your multiple clients. The logic of the web application server should be written as a set of services available to any interface. This will allow you to build any number of clients that utilize these core capabilities.

A SOA is also a good approach if you need to create a hybrid native/web application. Some native applications are simply site specific browsers- a native program that accesses and utilized a single website. This is often done so that a web application can go through the sales cycle of a native application. This approach can also be used if a single feature must be native but the rest can be a web based. You are able to gain the benefits of a web application while still satisfying requirements that can only fulfilled natively.

Finally, leverage toolkits liberally but judiciously. Toolkits really quicken the development cycle. They provide convenience for complicated code. They also abstract much of the non-standard behavior of many of the most popular web browsers.

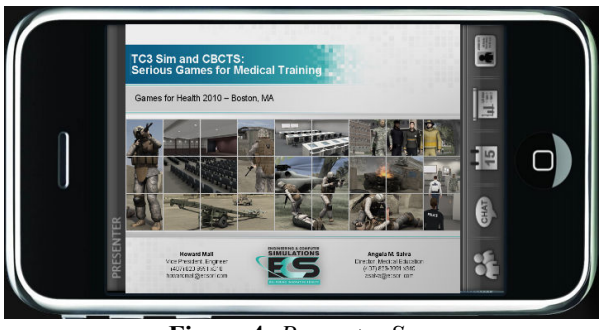

Figure 4: *Presenter Screen*

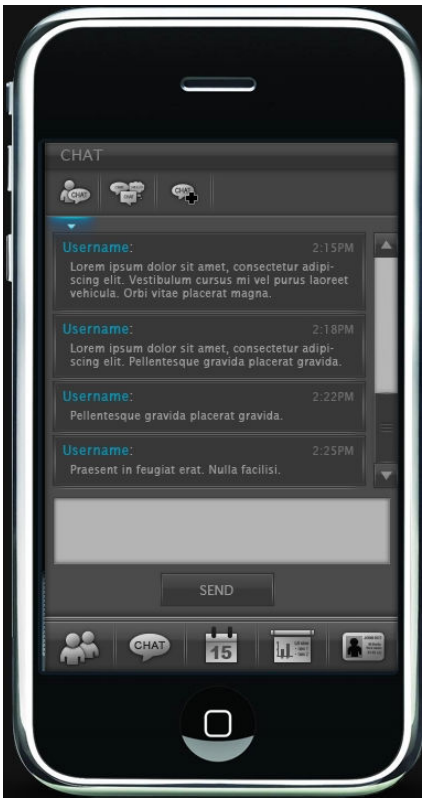

Figure 3: *Chat Screen*

Their simplicity, however, comes at the cost of flexibility. They can be as constraining as the platform you are targeting. Be very careful when mixing toolkits, especially javascript web toolkits (e.g. JQuery), they often do not play well together.

FUTURE WORK

The future of Nexus-Mobile falls into two main areas: interoperability and native capability.

Nexus core services must be integrated into the web application for complete interoperability. This will require the core services of the Nexus to provide web frontends that can be accessed by web applications. For example, the chat server will be accessed both by client applications and the web application server acting as a client.

The 3-D environment and VOIP are two areas that still need to be addressed by this work. 3-D Virtual World capabilities have two avenues to explore: a native application that renders 3-D content using the graphical capabilities of the mobile device or "framethrower" technology which renders 3-D content on a server and

then feeds video down to the client. Some initial experiments using a framethrower on the iPhone have worked but experience significant lag. Native 3-D mobile applications still have the problems mentioned at the beginning of the paper, but now many open source engines are being ported to mobile platforms (e.g Ogre3D (Rogers, 2009)).

REFERENCES

Chen, Brian X. (2010). Adobe Gives Up on Flash for iPhone, iPad. Retrieved June 21, 2010 from http://www.wired.com/gadgetlab/2010/04/adobe-flash-iphone/

comScore (2010). January 2010 U.S. Mobile Subscriber Market Share Report. http://www.comscore.com/Press_Events/Press_Releases/2010/3/comScore_Reports_January_2010_U.S._Mobile_Subscriber_Market_Share

Corbet, Peter (2010). Apps for the Army builds 53 Apps in 75 Days. Retrieved May 25, 2010 from http://www.istrategylabs.com/2010/05/apps-for-the-army-yields-53-apps-in-75-days/

Jacko, Julie A (2007). Human-computer Interaction: Interaction platforms and techniques. Springer Press.

Kolb, L. (2009). Cell Phones as Learning Tools. Journal for Michigan Association for Computer Users (MACUL). Fall 2007.

Malan, David J. (2009). Building Dynamic Websites. Computer Science Lecture. Harvard University.

Obendorf, Harmut (2008). The Making of the Palm Pilot – Reflections on a Minimal Information Appliance. Department for Informatics - University of Hamburg.

Rogers, David (2009). Q&A Ogre Iphone. Retrieved June 28, 2010 from http://www.3d-test.com/interviews/ogre3d_iphone_1.htm.

Yang, Daylen (2010). iPhone Case Study: The App Store and SDK. Retrieved June 19, 2010 from http://issuu.com/daylen/docs/iphone.