A Comparative Evaluation of HTML5 as a Pervasive Media Platform

Tom Melamed and Ben Clayton

Hewlett-Packard Labs Europe, Bristol, UK
{tom.melamed,ben.clayton}hp.com

Abstract. This paper introduces the field of pervasive media, the use of pervasive computing to deliver applications and services into peoples' lives, and the problems of creating and distributing pervasive media applications to mobile phones. It determines the requirements of a development platform for pervasive media applications through analysis of a range of pervasive media applications, including games and tourism based applications. It then evaluates a number of platforms against those requirements, including J2ME and native Smartphone development. The paper then introduces HTML5 and through a description of its features evaluates it against the requirements. Finally it concludes that HTML5 is a good solution to the problems of creating and distributing most pervasive media applications and describes its advantages and disadvantages compared to existing solutions.

1 Introduction

Pervasive media is an emerging field that combines new technologies with rich media and experience design to create compelling experiences, based on the user's context. Pervasive media follows on from two main technological fields, firstly computing technology leaving the "beige box" to pervade our lives, from mobile devices, to large screen displays, to smart cards. Secondly, sensing and instrumentation technologies are becoming cheap, ubiquitous and connected. Pervasive media takes advantage of these technologies to create applications and services that offer value to users which would not otherwise have been possible. For example a pervasive media treasure hunt game might run on a mobile phone providing the user with clues and tasks based on their GPS location. However as the examples provided in section 3.2 show there are many different forms of pervasive media application and more will probably emerge as the technologies and tools become more widely available.

Pervasive media points to a world of embedded technology and sensors in the environment capable of enhancing and augmenting everyday experiences based on the user's unique context. However in the short term mobile phones are a very good delivery platform for pervasive media as they are carried by users throughout their normal life and contain an increasingly rich set of sensors and media delivery options. This allows pervasive media applications to interweave with a users normal life offering interactions or content based on the user's context. Context can be physical and locally sensed such as a user's location sensed via GPS or more abstract such as the identities of the user's friends from Facebook.

This paper focuses only on pervasive media applications delivered through a mobile phone. We enumerate the main problems of developing for mobile phones currently, as a motivation for looking at alternative ways of developing such applications. We then look in more detail at the specific requirements of pervasive media applications by studying several of them from different genres and built using different technologies. We analyze the features from this sample set to identify common requirements for the creation of pervasive media applications. We use these features and the main problems of phone development as criteria to comparatively evaluate several different runtime environments available on different phones. Some runtime environments form the actual OS of the phone such as Windows Mobile or Symbian, others run on top and host applications within a virtual machine such as J2ME or Flash Lite but all are candidates to deliver pervasive media applications and so all are considered equally. We then introduce HTML5 which provides the specification for a browser-based runtime environment. We then evaluate its features in relation to the criteria that we had previously identified and in comparison to the other runtime environments. Based on this analysis we highlight some of the current shortcomings of HTML5 for pervasive media application but conclude that for many such applications it provides a good runtime environment.

1.1 HTML5

This paper is focused on the evaluation of one runtime environment in relation to the others, that environment being the web browser. The web browser is rapidly evolving from a renderer of simple html into a runtime environment capable of delivering rich interactive applications across many application domains. Browser development is primarily focused around the emergence of the HTML5 standard [1]. This standard effort grew from the Web Hypertext Application Technology Working Group which formed with a specific goal: "... to address the need for one coherent development environment for Web Applications." [2]. The original HTML5 specification consists of a number of different sections including offline storage of web content [3], the canvas element and others. There is another standard that did not grow from HTML5 called the Geolocation API specification [4], this is also part of the W3C specification effort and is focused specifically on an API for web pages to determine the browser location. For simplicity and brevity this paper will use *HTML5* to refer the full HTML5 specification and use *HTML5+GL* to refer to the combination of HTML5 and the W3C Geolocation specification. At the time of writing the standards are still in the drafting stages, but they are already being adopted by some web browsers. For example the latest versions of Mozilla's Firefox and Apple's Safari browsers already support a large percentage of these specifications, such as media playback and offline storage.

2 Problems with Mobile Development

Pervasive media is a new medium and as such its full potential is yet to be determined but there is already considerable interest within research, creative and commercial sectors. A number of commercial ventures and research projects have undertaken to create pervasive media applications for mobile phones, such as those discussed in section 3.2. This section will list some significant difficulties with current phone development. Most existing solutions contain certain tradeoffs such as the features available and the size of the potential market, or the ease of development and of distribution.

Development for mobile phones often requires different skill sets and familiarity with different runtime environments than those more commonly used on desktop PCs. Phone development requires knowledge of runtime environments such as J2ME, Objective C on iPhone or C++ on Symbian. There are also issues of memory management and constrained system resources that can be cumbersome and unfamiliar to PC developers.

Distribution is often piecemeal and fragmented with many users unfamiliar with the mechanisms available to download applications to their devices. While some Smartphones such as the iPhone and Android platforms are addressing this issue successfully, delivery to many devices is still problematic.

There can be restrictions and costs associated with developing and deploying to certain platforms, such as needing an Apple developer licence and Apple's approval to distribute applications to iPhones.

Device fragmentation continues to be a problem for developers with many different phone operating systems, and programming APIs. Even within a single runtime environment such as J2ME, fragmentation between devices and inconsistencies of implementation can greatly increase the cost and time involved in development.

Access to the device's sensors such as location and accelerometers can be restricted to certain applications or programming languages. Access to the user's context is very important for the execution of pervasive media applications.

An ideal goal would be the ability to write an application once using a well documented and supported language and set of APIs and have this application instantly available to all mobile devices irrespective of device or network. Unfortunately no such solution currently exists; this situation has forced makers of pervasive media applications to find compromises and increased the barriers to entry to this field. For example all of the applications studied in this paper were written for single specific runtime environments, reducing the potential user base.

Emerging web standards are one possible solution to the problems outlined above; the main contribution of this paper is an evaluation of these new technologies with respect to the problems outlined above and the specific requirements of pervasive media.

3 Requirements

In this section we shall enumerate and analyze the requirements for a successful pervasive media runtime environment by first considering general mobile development requirements and then by looking at a number of pervasive media applications and their individual requirements.

3.1 General Requirements

All mobile phone runtime environments require a base level of functionality to be useful for creating popular and or commercially successful applications. While it is possible to build applications on runtime environments without these features the ease of development and chance of commercial success is greatly diminished. For applications built for research purposes these features ease both development and the recruitment of test subjects.

- Large target market of installed devices.
- Good and accessible distribution channels to reach that large installed base.
- Good developer support, including documentation, libraries and community.
- Useable interface options, the user has to be able to find the application within the phone and understand how to use it.
- Efficient execution allowing for complex applications to be run on phones with limited processing power.
- Efficient use of limited resources such as battery life and system memory.

When evaluating different development platforms for pervasive media applications these should also be enumerated and considered.

3.2 Requirements from Existing Applications

In this section we analyse a number of pervasive media applications in order to determine which requirements are common amongst them. We will not try to distinguish which requirements are specific to pervasive media only which are commonly used by such applications. Those are the features that a runtime environment should support in order to support pervasive media applications.

Because all existing pervasive media applications on mobile phones have been constrained by the available technology it is not always easy to determine desired features from existing applications. Constraints within the technology may well have forced the designers of these applications to make certain unknown tradeoffs. However limiting ourselves to applications that have been built does show the features that are immediately desired and useful to pervasive media applications and those that have been used repeatedly by different applications are likely to be useful to other application authors too. Furthermore if pervasive media applications from different subgenres such as games, tourism or theatre and produced by different groups contain the same features then this is an even stronger indication that these features are general case requirements for pervasive media applications.

A number of applications were chosen for study that covered a broad variety of pervasive media genres, created by different groups and focused on different aspects of the technology. As phone technology moves so rapidly it was deemed acceptable to include some applications designed a few years ago for PDAs, while some of the PDAs' features were not then available in mobile phones such as a large touch screen and GPS, all are now available in phones such as the iPhone and Google Android devices. Some of these applications were created using toolkits such as mscape [5] and Mupe [6]. While the features available in these toolkits may have been a restraint on the features used in these applications, those built using toolkits do not appear to have different features or requirements to those built from scratch. For applications to

Table 1. Overview of surveyed applications

	Year	Platform	Toolkit	Loaned device	Genre
Bot Fighters [7,8]	00	Any SMS device	N/A	No	Game
Riot! 1831 [9]	04	Windows Mobile	Mobile Bristol [10]	Yes	Located audio play
Stamp the Mole [11]	07	Windows Mobile	Mscape	No	Game
Uncle Roy All Around You [12,13]	03	Windows Mobile	N/A	Yes	Game
GPS Mission [14]	09	iPhone	N/A	Yes	Game
Insectopia [15,16]	06	J2ME	Mupe	No	Game
'Ere Be Dragons	05	Windows Mobile	Mscape	Yes	Game/ Exercise
Feeding Yoshi [17]	06	Windows Mobile	N/A	Yes	Game
REXplorer [18]	07	Nokia N70	N/A	Yes	Tourist/ Game

be considered relevant they had to use more than just voice telephony and be in some way influenced by the users' context, for this reason applications like Pac-Manhattan (which only used voice) and Day of the Figurines (which was not context-based) were excluded.

The pervasive media applications studied are summarized in Table 1 above. From these applications a number of common features were identified that relate to functionality within any given runtime environment. After the table each relevant feature will be discussed in order of frequency within the sample applications. Some applications where designed to be played in a single session for a limited time period on a loaned device that was then returned at the end of the session.

Local Execution

Any application that actually runs programmed instructions on the phone counts as local execution. While this is common among all phone applications it is not required. Bot Fighters is the only pervasive media application surveyed that did not have local execution; instead all interactions took place via SMS messages sent to a central server.

Interactive User Interface
Again this is a common requirement across most phone applications. Any application that the user controls through either a touch screen or buttons is counted. However because pervasive media applications also use the users context, it is possible to create applications that do not require traditional input methods. Of those surveyed only Riot! 1831 did not use an interactive interface. It should also be noted that ‘Ere be dragons did not have any input on the device but did have a dynamic display based on the users context.

Audio
Only applications that play pre-recorded sound files are, for the purpose of this paper, considered to play audio, simple beeps do not count. Many pervasive media applications use sound extensively as it has been found to be a good way to provide information or content to the user without distracting their gaze from their soundings. Sound is used in these pervasive media applications; Riot 1831, Stamp the mole, GPS Mission, ‘Ere be dragons and REXplorer.

Location Sensor
This feature is of primary importance to pervasive media as location is a key form of context. This paper does not distinguish between the different location-sensing technologies such as Cell Tower based positioning or the much more accurate GPS. Not all pervasive media applications use a location sensor but the following surveyed applications do: Riot 1831, Stamp the mole, GPS Mission, ‘Ere be Dragons and REXplorer.

Server Interaction
Any application that relies on a server interaction for parts of its functionality fits into this category, for example an email program. This is a common requirement for many applications outside pervasive media. Within pervasive media applications servers are used for many reasons such as coordinating a multiplayer game, retrieving context that is not sensed locally or updating a high score table. Within our sample Bot Fighters, Uncle Roy, Insectopia, GPS Mission, and ‘Ere be Dragons all used a server.

Non-location Sensors
While location is a key form of context there are many others that can be sensed from some mobile phones, such as acceleration or proximity. Most mobile phone applications do not use such sensors but an increasing number of them do, such as those that use accelerometers to re-arrange the interface for the devices orientation. The sensors used in our sample survey are Bluetooth and WiFi proximity in Insectopia and Feeding Yoshi respectively, heart rate in ‘Ere be Dragons and accelerometers in REXplorer.

Local Persistence
Local persistence is characterized as any application that can save state between one instance of being executed and another, for example saving the status of the game or the user history. This is a common requirement among applications from many fields however only three of the sampled pervasive media applications used local persistence, this is probably due to the high number of applications that were created to be loaned out with devices for a single session. Of those that were not designed to run on devices that were loaned out, all but one used local persistence: Stamp the Mole, Insectopia, GPS Mission, and Feeding Yoshi.

Video
Some applications play video as part of their interface or content. For example some games use videos as an introduction to the game. Within the surveyed pervasive media applications video does not appear to be used very often. This could be due to its ability to draw users into the screen. While seated at home this is often advantageous but while moving through a rich environment it can be preferable and safer to not get too absorbed in the screen of the device. Only REXplorer used video in our sample set.

Alert Users
It is often advantageous to alert the user to some opportunity or change within an application. A common example of this is a phone alerting the user to a new email or SMS message. Within pervasive media applications one could imagine the user being alerted when their context matches certain criteria – such as moving to a location where a game can be played.

In our sample only Bot Fighters and Feeding Yoshi alerted users to situations they may want to respond to. This low number of applications may have been due to the number of applications that were designed to be lent out for single sessions where the application is already assumed to be the main focus of the user's attention.

Content Capture
Any application with the facility to record content such as audio, video or photos would fit within this category. Uncle Roy, GPS Mission and REXplorer used content capture, for photo and/or voice recording.

3.3 Requirements Analysis

From our survey of pervasive media applications we can draw a table of feature usage.

While it is not possible to draw strong conclusions from the exact frequency of each feature, overall patterns are worthy of consideration. The most common requirements appear to be local execution and interactive user interfaces. Other common requirements appear to be playing audio, interaction with servers and location sensing. The fact many different and independently produced applications use a similar feature set implies that even if this sample is somewhat small these features would still feature regularly in other applications.

Table 2. Feature usage by surveyed application

	Bot Fighters	Riot! 1831	Stamp the	Uncle Roy	GPS MIssion	Insectopia	Ere Be Dragons	Feeding Yoshi	REXplorer	Total
Local execution		X	X	X	X	X	X	X	X	8
Interactive UI			X	X	X	X		X	X	6
Play Audio		X	X		X		X		X	5
Use Location sensor		X	X		X		X		X	5
Interact with server	X			X	X	X	X			5
Non-location sensors						X	X	X	X	4
Local persistence			X		X	X		X		4
Content capture				X	X				X	3
Alert users	X							X		2
Play Video									X	1

4 Evaluation of Existing Runtime Environments

From the requirements obtained in section 3 above we now have a list of general phone development requirements and required features used in pervasive media. These requirements will now be used to evaluate a number of different mobile phone runtime environments to assess their suitability for building and distributing pervasive media applications.

The ideal runtime environment would address the problems discussed in section 2 above and meet the requirements determined from the analysis in section 3.

There are many existing runtime environments for the creation of pervasive media; all the examples discussed in section 3 were built using one of these platforms. The features of the most common runtime environments are summarized in Table 3 above. Neither mscape nor Mupe are analyzed as runtime environments. This is because they act as abstractions on top of other environments and the small number of devices with mscape or Mupe installed makes them of limited direct interest. Mscape runs within Windows Mobile which is a Smartphone operating system and Mupe runs with J2ME.

It can be difficult to categorize a platform as specifically having or lacking a certain property such as developer support or good distribution. Table 3 below is an attempt at an impartial categorization. Details and justifications will then be described for each runtime environment.

Table 3. Requirements against runtime environments

	Voice /SMS	J2ME	HTML4	Flash Lite	Native Code	HTML5+GL
Large installed base	X	X	X	X	X	Δ
Good distribution	X		X	O	O	X
Consistent implementation	X		O	X		X
Developer support	X	X	X	X	X	X
Launching applications	X	X	X	O	X	X
Efficient execution	X	X	X	X	X	X
Use of limited resources	X	X	X	X	X	X
Local execution		X	O	X	X	X
Interactive user interface	X	X	X	X	X	X
Play audio		O		X	X	X
Use location sensor	O	O			X	X
Interact with server	X	O	X	X	X	X
Use non-location sensors		X		X	X	X
Local persistence		O			X	X
Play video		O			X	X
Alert users		O		O	X	X
Content capture	X	O			X	

X Criteria met.
O Partial complience.
Δ Critera that is expected to rapidly become met.

4.1 SMS and Voice

A user can interact with a service or application by sending and receiving messages or talking through their phone. This is often used for televised competitions or booking cinema tickets. While this is not strictly a runtime environment it does offer a means to deliver a pervasive experience to a phone as Bot Fighters demonstrated. All phones are capable of voice communications and the vast majority can also send and receive SMS messages with most users very familiar with using these features. It is easy for developers to send and receive SMS or voice calls programmatically.

However without any local execution or access to local sensors the scope of applications is limited and latency and the cost of round trips can become prohibitive. While some operators can determine the crude cell location of SMS as used in Bot Fighters this tends to cost the developer on a per usage basis and normally requires a separate agreement with each operator.

4.2 J2ME

Java is the most common runtime for mobile phone development. A version of the J2ME Java profile is installed on millions of phones and there is a large and mature developer community.

However J2ME is a fragmented platform with a many different versions and a number of different combinations of optional APIs for features such as media playback, network access, and access to sensors. This fragmentation makes it very difficult to run the same program on any large subset of that installed base without resorting to the lowest common denominator, which may lack the features required for pervasive media. This means that there is effectively a smaller installed base if one takes these requirements into account.

Wide distribution is also a problem for J2ME as each network operator runs their own distribution and billing channels. These systems can be hard to access and generally require the developer to provide versions of the software that can run on all of their supported phones, making use of optional high-end features problematic.

4.3 HTML4

Most modern phones now come with some form of web browser that can render a subset of HTML, CSS and JavaScript in addition to WAP content. This provides an easy route for developers to deliver content to users, with few issues of signing or operator approval and a wide potential installed base of capable phones. Such pages can show text and images and include simple web forms and being server-based can easily be updated to reflect the current state of the application.

Being confined to the browser means that these pages cannot access much of the device's native functionality such as location-sensing or rich media playback. Also due to the inherent networked nature of this method, without the ability to store data locally, data costs and latency can become issues. As with J2ME incompatibilities between phones can be a problem but as the content is dynamically generated from servers it is possible to match the content to the capabilities of the device's browser without having to issue and distribute new binaries. Another issue is the variance in JavaScript support making some browsers capable of much more interactive interfaces and local execution than others.

4.4 Flash Lite / Silverlight

Flash Lite is a version of Adobe Flash specifically targeted at embedded devices such as phones and set-top boxes. It provides much of the rich interactivity and media capabilities of desktop Flash with compromises for the constrained devices it is targeted at. There is a relatively large development community but the tools used to create Flash Lite content are proprietary and not free. While there are different versions of Flash Lite they are much less fragmented than J2ME. However, due to Apple's App Store terms and conditions restricting virtual machines, Flash will not be available for the iPhone for the foreseeable future.

There is no access to the device's location sensors and earlier versions which make up most of the installed base [19] have limited or no video playback options, depending on the native device. Distribution and execution of flash content varies from device to device, for example some devices can run Flash Lite content from within a browser while others can't.

Silverlight is a technology similar to Flash from Microsoft, though at the time of writing the player is restricted to the desktop, with mobile versions currently in development.

4.5 Native Code

Native applications for Smartphones include software written in C++ for Symbian, Objective C for iPhone, Java for Android, and C++ for Windows Mobile. Programs written for the device's native runtime environment have rich APIs and tend to be able to use all the facilities of the phone, including sensors, media playback, file management, and network access. They generally run faster than software run in managed execution environments such as J2ME or Flash Lite. However porting code between different Smartphone operating systems can be costly due to the different programming languages used and the relatively low level of some of the languages, for example with C++ and Objective C the developer has to handle their own memory management.

There is a wide discrepancy within this group with respect to addressable market and distribution options. For example the iPhone has a relatively small but rapidly growing addressable market (about 45 million devices including the iPod Touch) and simple application distribution once you have obtained a developer license from Apple and bought an Apple Macintosh Computer to develop on. While Symbian has a larger addressable market and fewer restrictions on development it also has a far more fragmented distribution model. Most operators run their own distribution channels for Symbian apps and independent developers also distribute content via their own sites. Nokia, the largest distributor of Symbian devices will soon release an Application Channel similar to Apple's however that software itself will have to be distributed to the existing installed base so its uptake is uncertain.

5 Evaluation of HTML5+GL

As stated in the introduction HTML5 is being specifically designed to provide a single platform for the delivery of web based applications, but it was not specifically designed for pervasive media applications on mobile phones. In this section we evaluate HTML5+GL with respect to the previously determined problems and requirements.

An application based on the HTML5+GL runtime environment consists of a HTML web page created in the HTML5 mark up and some amount of JavaScript as the programming language. This is loaded in to a Browser which constructs a Document Object Model (DOM) from HTML and executes the JavaScript. The APIs for the features described below form part of the HTML5+GL DOM. The program receives inputs via events within this DOM and then processes the data using an arbitrary amount of JavaScript; the responses are then typically changes to the DOM which change the page contents and media being used.

5.1 Large Installed Base

At the time of writing HTML5+GL has a relatively small installed base as only the latest versions of a few desktop and mobile browsers support significant subsets of its functionality.

Increasing competition within the Smartphone sector and consumers' increasing reliance on web-based services such as web-based email has made the mobile web experience a focus of development and competition within the Smartphone sector. The lowering cost of cellular data and the prevalence of web content aimed at mobile phones have also increased the expectation among users that even low-end phones will have web browsers. Generally speaking popular features, applications and services found on Smartphones migrate to non-Smartphones over time; there is no reason that HTML5+GL will not follow the same pattern. An example of this migration is the camera. Current technologies following this trend are GPS and WiFi connectivity both of which are starting to appear in non-Smartphones.

Another significant factor in the uptake of HTML5+GL within mobile web browsers is the prevalence of the Webkit rendering engine across both desktop and mobile web browsers. Webkit is an open source rendering engine for web browsers that is used within Safari and Chrome on the desktop and Mobile Safari, the Android browser, the Nokia S60 browser, the Nokia S40 browser and WebOS from Palm. As the team behind Webkit are involved in drafting HTML5 as well as being at the forefront of adoption of HTML5 it is assumed that these browsers will continue to adopt these emerging standards as Webkit does. The use of Webkit in the Nokia S40 platform is particularly interesting due to the general popularity of this platform and the fact that it is not a Smartphone platform.

The forces driving the adoption of location sensing within mobile phones include the rise of GPS-based navigation and location-based services such as Nokia's Map 2.0 and Google Maps.

Mobile browsers that do not use Webkit are also adopting the draft HTML5 specification or providing very similar specifications. Opera is actively supporting HTML5 and its desktop support of HTML5 is very good. Opera has recently announced that it will soon support a large number of HTML5 features in its mobile browser, however it is doing so via a Google Gears implementation rather than direct HTML5 support. While it will have most of the same features, excluding media playback, the APIs will differ [20]. This difference in the APIs, though not the missing features, can be mitigated by an intermediate JavaScript Layer [21]. It is expected that the APIs for Google gears and HTML5 will converge [22].

As an example of the pace of HTML5+GL deployment within mobile phones Apples recently released iPhone Software 3.0 has updated some significant portion of the 45 million deployed iPhones and iPod Touch devices with a new browser featuring most of the HTML5+GL features discussed in this paper.

5.2 Good Distribution

Most users in the developed world are already familiar with using the internet in some form [23]. On a HTML5+GL equipped mobile phone any user who can access a web page can download a pervasive media application as they are the same thing. The server-side technology for distributing a HTML5+GL pervasive media application should be essentially the same as the technology and requirements of distributing normal web-based content or services as they both consist of the HTTP distribution of HTML, JavaScript, CSS and content files.

5.3 Consistent Implementation

The HTML5+GL specification requires implementations to be consistent across any number of operating systems and devices. Due to the detailed specification and test suites currently being devised by the HTML5 group within W3C [24] all browsers should in time be highly consistent application environments. In the mean time the prevalence of Webkit, which is highly standards compliant, across many mobile browsers will help to establish a de facto standard within the mobile browser space.

5.4 Developer Support

JavaScript used in conjunction with the HTML DOM is already one of the most widely used programming languages in the world with a huge community of support, tools and documentation. For developers with experience in this field, creating pervasive media applications is an incremental step up from the development of a website, and should not require investment in significant new skills.

This also raises the possibility of existing web sites targeted at mobile phones starting to take on pervasive media features such as adapting to the users location. Despite these benefits, a new user learning to develop JavaScript applications may take a significant time due to the number of technologies required (HTML, JavaScript, CSS, and optional server-side components such as PHP).

5.5 Launching Applications

The ease of finding an HTML5+GL application within a phone's menu will be determined by the phone manufacturer or browser developer and not the HTML5+GL specification. However as each application is effectively a web page any interface options available to find and launch bookmarks will also work for HTML5+GL applications.

5.6 Use of Limited Resources

JavaScript is an interpreted language meaning that well-written JavaScript will never run as efficiently as a well-written natively compiled application. However recent developments within desktop browser technology have greatly increased the execution speed of JavaScript with each major browser vendor competing on the basis of JavaScript execution speed [25]. The technology and techniques used in this competition are already starting to reach the mobile browsers [26].

For games and other-graphics heavy applications DOM-manipulation is often used such as dynamically creating or modifying DOM elements. This can result in fairly slow applications as the browser has to keep track of all the page elements to keep the layout up-to-date in real time. Using SVG (Scalable Vector Graphics) is another option but this also can become slow with large data sets. HTML5 includes a new Canvas element that makes drawing and animating graphics considerably faster [27]. Currently there is no support for text, though this is planned.

The offline aspects discussed below also help to reduce the amount of bandwidth required to run applications. However due to its interpreted nature a HTML5+GL application may well never be as efficient as a natively written application. But as phones become more powerful this should become a less significant constraint.

5.7 Play Audio and Video

The HTML5+GL specification contains new elements for the native playback of audio and video. Until now audio and video within web pages have been played through the use of plug-ins such as Adobe Flash. This means that browsers which do not support plug-ins, such as non-Smartphones browsers, but do support HTML5+GL will be able to playback video and audio.

However, at the time of writing attempts to standardize the codecs for the HTML5 audio and video elements have stalled. This means it may still be required to transcode video into different formats on a per-browser and per platform basics in order to make applications truly portable.

The Safari browser on iPhone 3.0 supports both the audio and video HTML5 elements. However, when audio or video is played a full-screen media player is shown, including transport controls. The user then needs to press play manually to access the content. Furthermore, only a single audio file can be played at any one time. These behaviors are contrary to the HTML5 specification, so it is hoped that future versions of the iPhone software will correct them.

5.8 Use Location Sensor

The W3C Geolocation specification provides a mechanism for the browser to query the device's location irrespective of the specific location technology being used. It also allows for the browser to specify conditions based on accuracy and receive information relating to the accuracy of the location determination. Mobile Safari supports this API in iPhone Software 3.0 and above.

5.9 Interactive User Interface

HTML4 is already used to provide rich interactive interfaces to users such as the Google Docs web-based office productivity suite. HTML5 does not remove any of the current APIs for doing this; in fact new APIs like the Canvas API create new opportunities for developers to create compelling application interfaces for users. An interface written in HTML5 will probably always be slower and less responsive than a native interface but there are large categories of application for which an HTML5 interface will be fast enough. This is already being demonstrated by the apps mentioned above and some simple arcade type apps such as Space War [33].

5.10 Server Interaction

Server-side interaction is used by web-based applications using various technologies based around AJAX web development. HTML5 improves on this by adding a property to indicate if the browser is online or not.

Furthermore there are now two technologies within HTML5 that allow web pages from one domain to communicate with servers or elements from another domain. *Access Control* allows an XMLHttpRequest object to make a request to a domain other than the one the page came from. *PostMessage* is a JavaScript method that allows two frames within a page from different origins to communicate in a secure way. Taken together these new features allow a web page to use web services and components from other domains where the servers and web pages within both domains have allowed it.

5.11 Local Persistence

HTML5 contains a number of complementary technologies for local offline storage and the retrieval of content and data [28]. HTML5 specifies client-side SQL database access which allows web pages to store structured data beyond the lifetime of any one page. This is already implemented in the Safari browser on iPhone, while the Android browser supports the same features with a different API via its implementation of Google Gears.

There is also a specification for web pages to be accessible even when the device is not connected to the internet. The *Offline Application Cache* allows the HTML page to specify that it and related content such as images and CSS files are stored locally for retrieval even if the page's original server is not accessible.

5.12 Non-location Sensors, Interruption of Users, and Content Capture

Pervasive media applications need to be able to monitor a user's context in order to deliver valuable services or content to the user. As more phones become available with other sensing technologies like accelerometers, more applications may wish to make use of them. However, non-location sensors are not currently covered within the HTML5+GL specification. From analysis of the sample pervasive media applications this would not appear to be a significant omission as very few of those applications used these features. However we believe that these features may become more significant over time.

Similarly, HTML5 does not directly cover capture of content via cameras/ microphones, though this will most likely be a significant contribution to future pervasive applications.

Once the user's context is appropriate for the application to offer them some kind of interaction or service the application may well want to alert the user to this opportunity. This is likely to become more common if more pervasive media applications become available that are designed to run on a users own phone over a long period of time. HTML5 does not have a specific notification API, though audio or pop-up windows may be a reasonable substitution, provided that the originating page is open.

6 Application Trial

In order to test HTML5's suitability for pervasive media applications an existing application was chosen and re-implemented in HTML5+GL. *Stamp the Mole* was chosen as the application because it requires many of the most commonly used features making it a fair representative of a typical pervasive media application.. Additionally, the creative commons licensing of the application and publically-available source code meant it was easier to port and legal to redistribute.

The new application was developed entirely in JavaScript and HTML5+GL using the NetBeans 6.5 IDE [29] to develop the JavaScript. It was tested using Safari 4 on the desktop, and on iPhone running version 3.0 software.

A HTML5+GL based pervasive media application is created in much the same way as a normal web page. A main HTML file is created; this is the scaffolding upon which the rest of the application is constructed. JavaScript libraries are included, by including scripts within the body of the HTML page. Execution starts when the page is loaded and the "onload" event is fired by the browser. From there the JavaScript can register with sensors and start to communicate with servers via AJAX. The JavaScript can process input from user interaction, sensors and server and modify the interface via manipulating the HTML page's DOM or drawing directly to a Canvas object within that DOM. The application continues to execute until it or the user navigates away from the page or closes the window or tab.

To allow maximum code reuse for future work the application was written in three layers. A base layer abstracted out the exact sensor and rendering technology, this meant that for example a flash movie could have been used for the media rendering. And on browsers that did not support the W3C Geolocation API a map, fire eagle [30] call or AJAX call to a location server could be substituted. For this application all media was rendered by direct manipulation of the HTML5 DOM and location was provided by repeated AJAX calls to a server.

The second layer consisted of a library of classes and methods that were expected to be re-used for other pervasive applications using HTML5+GL. They provided utilities such as inclusion testing for spatial regions defined in JSON. The third layer consisted of the Stamp the Mole's application-specific JavaScript that specified the exact behavior of the game.

The application was found to run well and be playable on the desktop and fully functional but less easy to play on the mobile browser. The major difference was the media playback limitations on iPhone already discussed in section 5.7.

While a single application can obviously not be extrapolated to the entire field of pervasive media it does provide an existence proof that such applications with typical features can be developed and deployed using HTML5+GL.

7 Discussion

As has been demonstrated HTML5+GL provides a platform independent and easily deliverable means to create and deploy pervasive media applications that use a limited set of the most common requirements. But there are other features that HTML5+GL does not support and it is not currently very easy to make any strong assertions as to how important they will be in future applications. One distinct possibility is that as phones become more capable more applications will target the user's existing device and not be designed with a single session device loan model. This will mean that features such as persistence, running as a background process to monitor the user's context and alerting the user to get their attention may well become more common requirements. HTML5 already has a persistence model but the other features are not currently supported.

7.1 Background Tasks and Notifications

Currently web browsers do not have a notion of background tasks; however this feature can be very useful for pervasive media applications. A common design pattern for pervasive media applications is to monitor various local or remote sensors to determine the users' context and then to attract the users' attention and deliver the appropriate service, content or interaction based on that context. This can be achieved by multi-tabbed browsers as they do tend to allow JavaScript execution in background tabs. However this requires that the browser stays open and does not provide a facility for the application to alert the user except through audio.

If the user's context is already measured or available off the device on a server then the process which monitors the user's context and decides when to alert the user could reside on that server. The user could then be alerted and the application restarted through for example the sending of an SMS message or WAP push. This does of course assume that the application knows the users phone number which cannot be assumed but could be asked of the user.

7.2 Access to Other Sensors and Content Capture

One of the strengths of the core HTML5 specification is its uniformity and lack of the kind of fragmentation that causes problems for J2ME developers. However just as location is supported by a different and complementary specification other sensors and content capture systems could also be introduced through standardized specifications and then optionally included in browsers. Of course any optional components may lead to fragmentation between devices' browsers, and whilst the time between new features being introduced into the HTML5 spec and their appearance in beta browsers is fast (a few months is common), the total time between an idea and its mainstream adoption is far longer, often years.

Several groups are finding ways to extend the browsers API to provide richer functionality. For example PhoneGap and Titanium Mobile [31] are frameworks that allow applications to be written in HTML and JavaScript on a variety of different Smartphones [32]. These applications are run in a native wrapper that gives access to features usually unavailable to JavaScript such as the accelerometer, the file system, contact databases, and vibration. From these systems it is likely that new standards efforts will result in the development of standard APIs for such services. However in the meantime those who are willing to trade platform independence for richer functionality have a means to do so.

8 Conclusion

Distribution has long been a problem for mobile software; this can observed by the amount of interest, developer support and downloaded applications that the Apple App Store has generated out of all proportion to its installed base. This could be in part due to the publicity and marketing of the iPhone and related factors, but the ease of distribution is a significant factor. With HTML5+GL distribution will be easy as each application will simply be a web page, which most users are already familiar with.

As this paper shows HTML5+GL provides all of the major features apart from background processing required to create pervasive media applications for mobile phones. As detailed in section 4, it is one of the best platforms in terms of coverage of the features that were found to be common requirements from section 3. There are potential issues with access to non-location sensors but this may improve over time through the introduction of new web standards or via proprietary extensions. It is not yet clear how background processes could be integrated with HTML5+GL and this may be a limit on the applications that can be built using this runtime environment but there is still a huge potential for applications that do not require such features.

HTML5+GL is also platform independent, this should provide a real boost for application developers as it reduces the cost of porting and increases the addressable market without compromising the application. If browsers supporting HTML5+GL become as commonplace as current HTML4 browsers, the addressable market will be very large indeed.

This paper has highlighted some of the problems of existing solutions to pervasive media and has shown how with HTML5+GL it should soon be possible to write a pervasive media application once distribute it anywhere and run it everywhere.

References

1. HTML5 specification, `http://dev.w3.org/html5/spec/Overview.html`
2. Web Hypertext Application Technology Working Group news, `http://www.whatwg.org/news/start`
3. Web Storage HTML5 specification, `http://dev.w3.org/html5/webstorage`
4. Geolocation API Specification, `http://dev.w3.org/geo/api/spec-source.html`
5. Mscape, `http://www.mscapers.com`
6. Mupe, `http://www.mupe.net`
7. Ready, aim, text, The Guardian, `http://bit.ly/XFABk`
8. Sotamaa, O.: All The World's A Botfighter Stage: Notes on Location-based Multi-User Gaming. In: Proceedings of Computer Games and Digital Cultures Conference 2002, pp. 35–44. Tampere University Press (2002)
9. Reid, J., Hull, R., Cater, K., Clayton, B.: Riot! 1831: The design of a location based audio drama. In: Proceedings of UK–UbiNet (2004)
10. Mobile Bristol, `http://www.mobilebristol.com`
11. Stamp The Mole, `http://www.mscapers.com/msin/ABA0000044`
12. Uncle Roy is All Around You, `http://www.blasttheory.co.uk/bt/work_uncleroy.html`
13. Benford, S., Flintham, M., Drozd, A., Anastasi, R., Rowland, D., Tandavanitj, N., Adams, M., Row Farr, J., Oldroyd, A., Sutton, J.: Uncle Roy All Around You: Implicating the City in a Location-Based Performance. In: Proc. Advanced Computer Entertainment, ACE (2004)
14. GPS Mission, `http://gpsmission.com`

15. Peitz, J., Saarenpaa, H., Björk, S.: Insectopia - Exploring Pervasive Games through Technology already Pervasively Available. In: Advancements in Computer Entertainment, Salzburg, Austria (2007)
16. Insectopia design document, `http://bit.ly/dUXfu`
17. Bell, M., et al.: Interweaving mobile games with everyday life. In: CHI 2006: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 417–426. ACM Press, New York (2006)
18. Ballagas, R., Kuntze, A., Walz, S.: Gaming Tourism: Lessons from Evaluating REXplorer, a Pervasive Game for Tourists. In: Proceedings of the 6th International Conference on Pervasive Computing. LNCS. Springer, Berlin (2008)
19. Flash Lite Devices Worldwide, `http://bit.ly/CcvoK`
20. Gears-enabled Opera Mobile 9.5 technology preview, `http://bit.ly/4bEQP`
21. HTML5 offline format implemented using Google Gears, `http://bit.ly/O6HIH`
22. Google Gears as a bleeding-edge HTML5 implementation, `http://bit.ly/y7B6T`
23. Internet Users Per 100 Inhabitants (1997-2007), `http://bit.ly/ewYuW`
24. HTML5 Editor Ian Hickson on features, pain points, adoption rate, and more, `http://bit.ly/eK37C`
25. SquirrelFish Extreme promises to speed JavaScript in Safari 4.0, `http://bit.ly/OIiN7`
26. JavaScript to get 3x speed boost in iPhone OS 3.0 – Ars Technica, `http://bit.ly/9MpgR`
27. Canvas vs SVG Performance, `http://bit.ly/tHf7s`
28. Offline Web Applications, `http://bit.ly/WwZXb`
29. NetBeans IDE, `http://www.netbeans.org`
30. Fire Eagle Location Service, `http://fireeagle.yahoo.net`
31. Appcelerator Titanium, `http://titaniumapp.com`
32. PhoneGap - Cross platform mobile framework, `http://phonegap.com`
33. Space War, `http://virtualgs.larwe.com/iPhone/space/`