Joshua Barnett

Registration number 5939968

An Evaluation of HTML5 as a Mobile Gaming Platform

Supervised by Professor Andy Day

University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

This is my dissertation.

Acknowledgements

These are my achknowledgements

Contents

1 Introduction

For the past five years it has been evident that mobile devices have risen as one of the foremost dominant ways in which the modern world consumes media and entertainment. Alongside this, a new web standard has been developing. HTML5 (a popularly abused buzzword) is a new language specification with the primary purpose of succeeding where its predecessors failed, in delivering rich interactive content. Generally "HTML5" is used as an all encompassing term that includes a host of technologies, as it alone could not deliver such user experiences. Technologies such as JavaScript (JS), Cascasing Style Sheets (CSS), Scalable Vector Graphics (SVG) and WebGL. When used together in varying combinations and configurations these form a Web Platform[1] on which to convey content and information in new and interesting ways. However, it is currently the convention to refer to this new platform as HTML5.

"Linux" has similarly been distorted as an all encompassing term for Linux based operating systems such as Debian, Fedora, and Ubuntu. However, "Linux" alone is just the open-source kernel which lies at the heart of these distributions but on its own would not be of much use.

OEMs (Original Equipment Manufacturers) such as Apple, LG, Nokia and Samsung are becoming progressively competitive in producing better mobile devices for the global market. They offer higher resolution displays, more memory capacity, and increasingly faster processors. This escalation in performance has been a key instigator in the growing support and optimization of HTML5 and JavaScript on mobiles. However, it has only just begun to reach the requirements needed for delivering interactive real-time graphics applications through HTML5's features, such as video games.

The purpose of this project is to evaluate the current state of the HTML5 and its companion technologies on the mobile platform, and to assess whether they provide a suitable means to develop and deploy games. To assist my research I will be designing and creating my own game, expanding my personal understanding of the platform along the way.

[1] *http://docs.webplatform.org/wiki/Main_Page*

2 HTML5

HTML5 is the fifth iteration of the hypertext mark up language. During its previous iterations such as HTML4.01 there was minimal support for multimedia content. This resulted in a void that was promptly filled by third-party plugins such as Adobe Flash and Microsoft Silverlight. Such plugins allowed for easy presentation of media such as video, audio and even allowed for interactivity that was used to great effect in games.

Unfortunately many of these plugins are proprietary and closed-source which prevents developers from fixing and debugging inherent issues in the technology. Instead they are dependant on the technology owner for solutions to these problems while being limited to reporting them or finding "temporary" workarounds. Often such plugins also require some form of installation process as they are "add-ons" and exist outside of the W3C standard.

Jobs (2010) famously professed his concerns about Flash and presented sound reasoning as to why Apple had no intentions of supporting or integrating Flash into their iOS devices. Flash was also briefly available for Android but it was short lived and eventually discontinued by Adobe. Adobe is continuing to making strides into the mobile platform with their new runtime AIR[2]. HTML5 is still currently a work in progress but the W3C (2012) have made plans to stabilize the specification and help it reach *Recommendation*[3] status as of this year.

2.1 The Document Object Model (DOM)

The document object model (DOM) is the core of what drives modern web applications. It provides a means to manipulate the structure and style of the original document loaded from a web server at run-time on the client side.

However, what makes HTML different from most compiled languages such as C++ and Java, is its a interpreted language. Compiled languages are typically converted into a binary format that compresses their instructions into machine-code allowing for

[2] *https://www.adobe.com/aboutadobe/pressroom/pressreleases/201002/021510FlashPlayerMWC.html*

[3] *http://www.w3.org/2005/10/Process-20051014/tr.html#rec-publication*

quicker execution. Interpreted languages in contrast are often read line-by-line with their instructions parsed and executed at run-time.

When HTML is interpreted by a web browsers it is parsed into a DOM. The DOM is essentially a representation in memory of what was originally marked down in the static HTML document. This allows the browsers to interpret relations between elements and use them to render the document appropriately. While in memory the information is not longer static allowing for manipulation by other technologies such as JavaScript.

One example of where this manipulation is used to great effect is the single-page application (SPA). SPAs have significant advantages over that of traditional websites as they can provide a seamless user experience similar to that of desktop applications. Instead of navigating through links that loading separate HTML pages they use states to manage the flow, loading information dynamically when required or requested. This methodology shrinks factors such as load times, enables rich interactions, responsiveness while encouraging reuse. (Takada, 2012)

2.2 Web Browsers

Web browsers are client side applications that render the information requested by the user delivered from a web server. Many of the popular browser vendors such as Google (Chrome), Apple (Safari), and Mozilla (Firefox) all have a mobile counterpart. However these mobile counterparts often support a subset of their desktop editions making much of the fringe HTML5 features even less supported.

2.3 Cross Compatibility

Getting a single web application to function and display consistently across a range of browsers can be a huge undertaking. As the specification of HTML5 was being developed much of the features have been implemented in browsers at the discretion of the browser vendor. This has quickly lead to cross compatibility issues as older browsers will support less of these new features and not necessarily be consistent with the other browsers that were also available at the time of release.

One such feature I had planned on using when initially developing my mobile game was *viewport units*. This feature allows for the scaling of elements based upon the viewport's dimensions which is especially useful on mobile devices for scaling document elements relative to the screen's aspect ratio. However the stock web browsers on mobile devices have almost non-existent support for this feature and it would severely limit my target platforms.

iOS Safari	Opera Mini	Android Browser	Opera Mobile	Blackberry Browser	Chrome for Android	Firefox for Android	IE Mobile
		2.1					
		2.2					
3.2		2.3					
4.0-4.1		3.0	10.0				
4.2-4.3		4.0	11.5				
5.0-5.1		4.1	12.0				
6.0-6.1		4.2-4.3	12.1	7.0			
7.0	5.0-7.0	4.4	16.0	10.0	33.0	26.0	10.0

3 Graphics

When it comes to rendering in HTML5 are I many different options with varying advantages and disadvantages.

3.1 CSS

CSS is used to style the way the document elements are rendered. By instructing the browser to use instructing the browser

3.2 Canvas

a

3.3 SVG

a

3.4 WebGL

WebGL is a distillation of the popular OpenGL making GPU accelerated rendering in the browser now a reality. WebGL, a JavaScript API is based on the OpenGL ES (Embedded Systems) subset, and as the name implies it was designed and optimized for embedded systems such as mobile devices. One streamlined modification OpenGL ES made was the removal of the fixed-function API introduced in OpenGL 1.0 enabling the use and compilation of modern shaders.

WebGL has been in development for the past three years which came to maturity as of early last year when the first a stable version was released. As this is a new technology it has only just begun surfacing in the wild, one notable example is the PlayStation 4's user interface. In a revealing presentation given by Dom Olmstead it was declared that the move to WebGL was done to ensure cross platform support. This potentially hints at Sony's future plans to bring their games to multiple platforms including mobile devices through their PlayStation Now streaming service.

However, the support for WebGL through mobile browsers is essentially nonexistent to date. One company striving to change that is Ludei, who are in the early stages of rolling out their new technology CocoonJS. CocoonJS is a means of packaging and publishing HTML5 applications on mobile devices. In amongst this framework is the facilitation of WebGL letting it tap into the onboard GPU available on most modern mobile devices. Many of the demos they provide demonstrate close to near native performance. This makes WebGL a feasible avenue for game development on mobile devices, with the added bonus of being cross platform ready.

4 Audio

a

4.1 `<audio>` Element

a

4.2 Web Audio API

a

4.3 Native Audio

a

5 Game Design

a

5.1 Player Interaction

a

5.2 Gameplay & Mechanics

a

5.3 Feedback

Throughout the development and prototyping of MindFlip it was important to take into consideration the reactions and feedback of players. So whenever the chance arose I would hand over my phone containing my latest stable build of the game and observe others playing it from a third-party perspective.

Preconceptions can be a double edged sword when used in game design. Often they can be used to draw parallels between activities in contemporary games such as my own. This acts as a shortcut when teaching a player new or similar gameplay mechanics. However, these can also be stumbled upon accidently from a developer's perspective as their own preconceptions about games will likely be vast by comparison to the average player.

This was the case with the initial draft of introductory level. In the beginning the first two card symbols the player encountered were a red nought and a blue cross on a three-by-three grid. After showing the game to my sister while providing no tutorial, the first actions she made were to produce a winning *Tic-tac-toe* game state. This made it apparent that there was a flaw in my game design, because the symbols are common to a pre-existing game namely *Tic-tac-toe* the player receives mixed signals about how to play the game from the offset leading to frustration and confusion. I later decided to go for simpler more abstract symbols such as circles, squares, and triangles in my initial set of levels do to their generic associations they imply simplicity leaving the player open to learn the game mechanics.

6 Development Stack

When creating a game targeting HTML5 it can be difficult to know what workflow to invest in. Developers each have their own preferences and beliefs as to what is the best way to work with their priorities often being similar but achieved in different ways.

6.1 Preprocessors & Alternatives

Preprocessors are vastly popular these days amongst developers with their focus often on efficiency. Transpilers (source-to-source compilers) are one such common way in which to improve a developers workflow. With current web languages such as JavaScript and CSS widely supported and heavily standardised by W3C it often takes along time for improvements to make their way into the next specification iteration.

The advantages of transpilers often mean that less can be written in order to achieve the same functionality in its target language. Simple common activities such as writing a for loop to iterate over an array or providing the vendor prefix and polyfills required to support backwards compatibility with older web browsers can be taxing on a developers time. Transpilers can offer an alternative means in which to achieve the same results bit with less code and effort required on the part of the developer.

6.1.1 CoffeeScript

CoffeeScript[4] is a popular alternative to JavaScript with its focus on readability similar to that of other syntactically sugary (Raymond, 1996, p. 432) languages such as Ruby and Python.

6.1.2 Sass

Sass is a popular CSS preprocessor which addresses many of its shortcomings.

6.1.3 Dart

a

6.1.4 TypeScript

a

6.2 Modular JavaScript

a

6.3 Model View Controller (MVC)

a

6.4 Templates

a

6.5 Build Pipeline

a

[4] *http:// coffeescript.org/*

7 Mobile Deployment

Once a HTML5 game

7.1 Packaging

a

7.2 Debugging

a

7.3 Profiling

a

References

Jobs, S. (2010). Thoughts on flash. *http://www.apple.com/hotnews/thoughts-on-flash/*.

Raymond, E. S. (1996). *The new hacker's dictionary*. Mit Press.

Takada, M. (2012). Adventures in single page applications. *http://youtu.be/BqDJqKGfliE*.

W3C (2012). Plan 2014. *http://dev.w3.org/html5/decision-policy/html5-2014-plan.html*.