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. To clarify what people really mean when referring to HTML5 is a host of technologies including but not limited to JavaScript (JS), Cascasing Style Sheets (CSS), Scalable Vector Graphics (SVG), WebGL, and the new Canvas element. 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, as it is becoming the convention to refer to this Web Platform as HTML5 I will follow suit in this report.

Because of the on going competition between OEMs (Original Equipment Manufacturers) such as Apple, LG, Nokia and Samsung in providing the better mobile products, the processing power of mobile devices has been increasing exponentially. This escalation in performance has been a key instigator in the growing support and optimization of HTML5 and JavaScript. 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. During the course of this project

2 HTML5

HTML5 is the fifth iteration of the hypertext mark up language. During the previous iterations such as HTML4.01 there was no support for multimedia content. And as a result this as patched by third-party plugins such as Adobe Flash and Microsoft Silverlight. This allowed for easy presentation of media such as video, audio and even

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

allowed for interactivity that was used to great effect in games.

However with these third-party plugins came limitations. One of which is they were proprietary and closed source. Meaning if there was an issue of debug with the runtime any developers working on these platforms would have to wait for a update or a patch. Or would have to find a work around.

Another disadvantage is the barrier to entry for most users. To view any content built for these runtimes they would have to be installed.

Steve Jobs vocalised his concerns with Flash and present sound reasoning as why Apple had no intentions of supporting or itergrating Flash into their iOS devices.

It was briefly avaliable for Android but it was short lived and eventually Adobe threw in the towel. However they are still making strives into the mobile platform with their new runtime Adobe AIR.

HTML5 is still currently a work in progress but the W3C have made plans to have the specification finalized as of July this year.

2.1 The Document Object Model (DOM)

a

2.2 Web Browsers

When a user navigates to a site it requests a page at a specified location via a URL. This address is in turned passed to a DNS server which returns the current IP address of where the requested page is located. Once the browser recieves the requested page it begins parsing it

2.3 Cross Compatibility

a

3 Graphics

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

3.1 Cascading Style Sheets (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[2] is a popular alternative to JavaScript with its focus on readability similar to that of other syntactically sugary Raymond (1996) 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

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

6.2 Modular JavaScript

a

6.3 Model View Controller (MVC)

a

6.4 Templates

a

6.5 Build Pipeline

a

7 Mobile Deployment

a

7.1 Packaging

a

7.2 Debugging

a

7.3 Profiling

a

References

Raymond, E. S. (1996). *The New Hacker's Dictionary - 3rd Edition*.