CS25210 Assignment
Client-Side Graphics Programming for the Web

Due

22 April 2014

Charles Callum Newey (ccn4)

Word count: 1041

Contents

1 Executive Summary

The game that I decided to develop is simple in nature, as I felt that a game need not be complex to be fun to play or visually pleasing. The game (called Zombie Pendemic - *pun intended*) is a simple mouse-based shooter - the user plays as a "turret" in the centre of the canvas, battling against a never-ending onslaught of zombies.

The player would have to "shoot" the zombies by clicking in the same direction. The player's score would increase when zombies were hit by bullets - with different score values for each "type" of zombie. The idea was for the game to get more difficult for the player over time. This was fairly straightforward when it came to implementation - the number of zombies on-screen simply increases as a function of the player's score. The zombies also increase in speed as the game progresses, ensuring that each game lasts for just enough time to stay interesting. Storyboards are included at the end of this document in the "Appendices" section (6).

One of the major design choices that I made early on in development was to make all of the sprites and sounds myself - this ensured that I had complete control of the look and feel of the game. This proved to be a lot more work than I had originally anticipated, but I am satisfied with the result.

2 Technical Overview

2.1 Technologies Used

Obviously, the broad architectural choices for the game were pre-defined - that is, HTML5, CSS and JavaScript. However, web application development is complicated and involves more than just language and rendering choices. In keeping with my original desire to keep the project as original as possible, I have used no external programming libraries (aside from Google Analytics[1]), and nothing other than a font from Google's Font API.[2] This allowed me to have full control over the quality and performance of my code base.

I dislike the idea of using a library such as JQuery or MooTools, as there are few advantages that I could see for this assignment. JQuery is particularly useful for manipulating the DOM and making AJAX calls (amongst other things), but in this instance, I didn't need either of those functionalities - the extra functionality would have been somewhat wasted.

2.2 Alternative Technologies

There are a number of existing technologies that could serve in place of HTML5 and JavaScript, such as Flash or Silverlight - however, these are far less popular (and less common, to an extent) than HTML5 and JS. Flash and Silverlight require additional plugins, and are also proprietary technologies - meaning that support across browsers and operating systems is varied. This makes HTML5 and JS the most sensible (and modern) choice.

There are also external libraries for JavaScript development (such as JQuery and MooTools), that make a lot of complex functions readily available - however, I have decided to forgo the use of these in favour of more control over performance and stability.

I chose to make a slightly unusual architectural decision within the application. Most HTML5 canvas games repaint the entire canvas on each subsequent frame of the animation - this struck me as wasteful, as this involves a lot of unnecessary redrawing. My implementation, then uses three separate layered canvases to hold several UI elements that may overlap. In this case; one canvas to draw the health bar, one to draw the bullets, and another for the zombies and the main turret animation. This helps to minimise CPU overhead - at the expense of a small amount of memory.

3 Testing

The game has been tested as well as possible - on several operating systems using several browsers. This includes (but is not limited to) Firefox, Google Chrome, Chromium, Internet Explorer and Safari. It was also tested on various operating systems and mobile devices; Windows, Mac OS, Linux (various distributions, but mostly Arch Linux), iPads and Android devices. The only problems I encountered really were with the HTML5 audio API, which proved troublesome on some older Android devices. Aside from the audio issues, the game was perfectly playable (and still fun, at that) on an Android device.

There was a particularly strange bug I encountered on the Linux x86 build of Firefox (running on Arch Linux) - Firefox's JS implementation would garbage-collect memory that was still in use by the main application, causing the game to freeze and crash. This hasn't happened in any other browser or operating system that I have tested, so as far as I could tell, this was an isolated case.

I also found that the game had some trouble with freezing on some older iPad devices - I couldn't work out the cause of this easily, as the default browsers on iOS devices don't include developer tools. However, the game worked surprisingly well on a number of mobile devices, and responded just as well to tapping (in the case of touchscreens) as well as clicking.

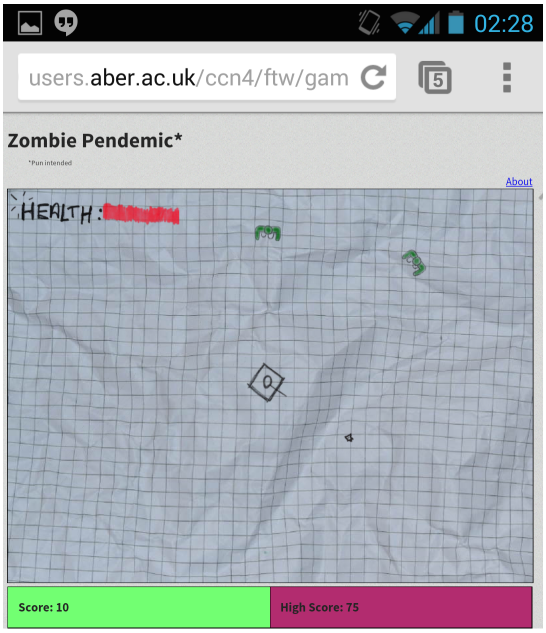

Figure 1: Gameplay on an Android device

Figure 2: Gameplay on Mac OS X (in Chrome)

4 Reflections and Future Work

If I were to repeat the project, I would have probably implemented a slightly more complex game. Although I am happy with the outcome, I would have liked to create something a little more challenging - both to make and to play. Also, I would have chosen a different style of game - one that was more conducive to keyboard control as the only keyboard interaction I could realistically fit in was pausing and resuming the game.

One thing I would have added to this project (given more time), would potentially be some form of multiplayer functionality. This is possible (and in fact, quite simple) using JavaScript and WebSockets. WebSockets are an emerging web technology that is useful for full-duplex communications over open sockets in a web browser.[4] This differs from AJAX as AJAX opens new connections upon every request. Because WebSockets have a greater longevity, they have a better latency (due to needing to open sockets far less frequently) - and this makes them far more suitable for a multiplayer online web game.

Overall, I'm very happy with the decision to make all of the game sprites and sounds myself - I am happy with the drawn effect of the entire game - it's exactly the effect I wanted to achieve. Graphically, the game isn't ground-breaking - but I think it looks pleasant, and the game is definitely good fun to play.

5 References

[1] Google, Inc. Google Analytics JavaScript Library.
http://www.google.com/analytics/.
Accessed: Apr 19, 2014.

[2] Google, Inc. Google Fonts API.
http://www.google.com/fonts/.
Accessed: Apr 19, 2014.

[3] Pavlic, T. Programming Assignment template for LaTeX.
http://www.latextemplates.com/template/programming-coding-assignment.
Accessed: Oct 23, 2013.

[4] W3C. WebSockets Specification.
http://dev.w3.org/html5/websockets/.
Accessed: Apr 19, 2014.

6 Appendices

6.1 Storyboards

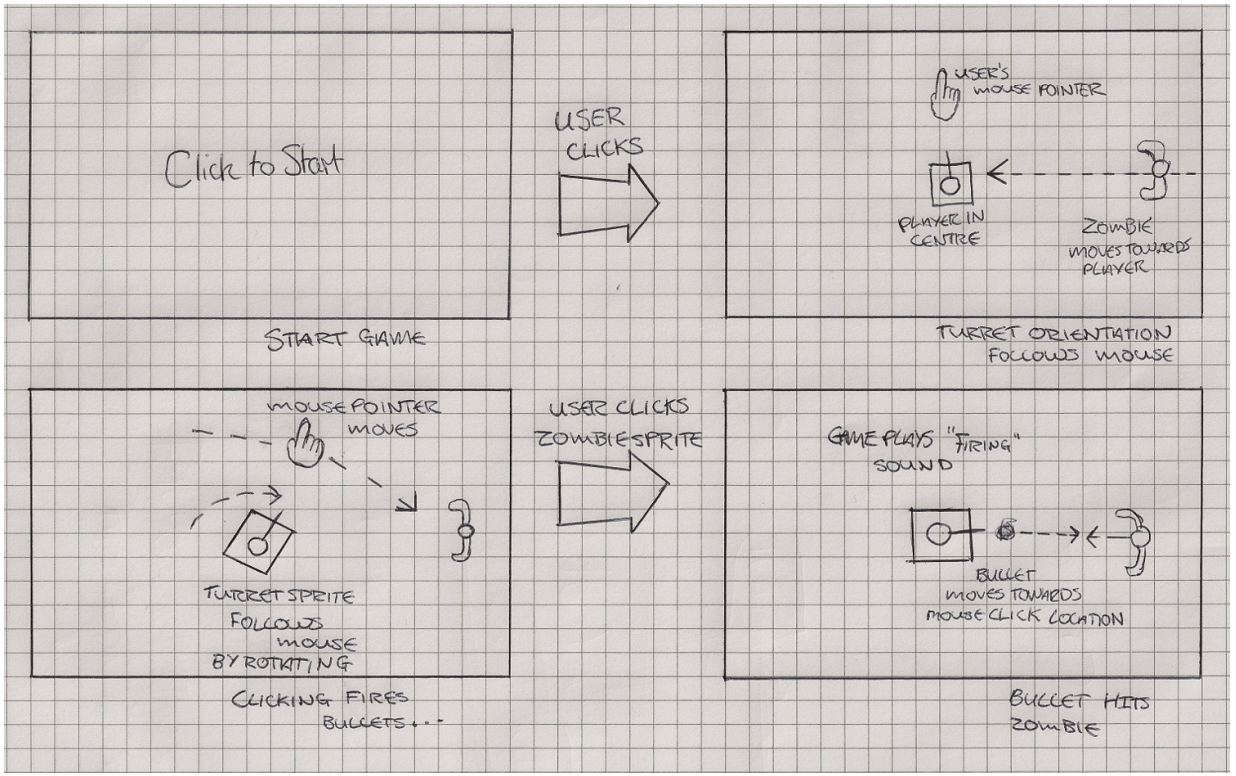

Figure 3: First storyboard in sequence

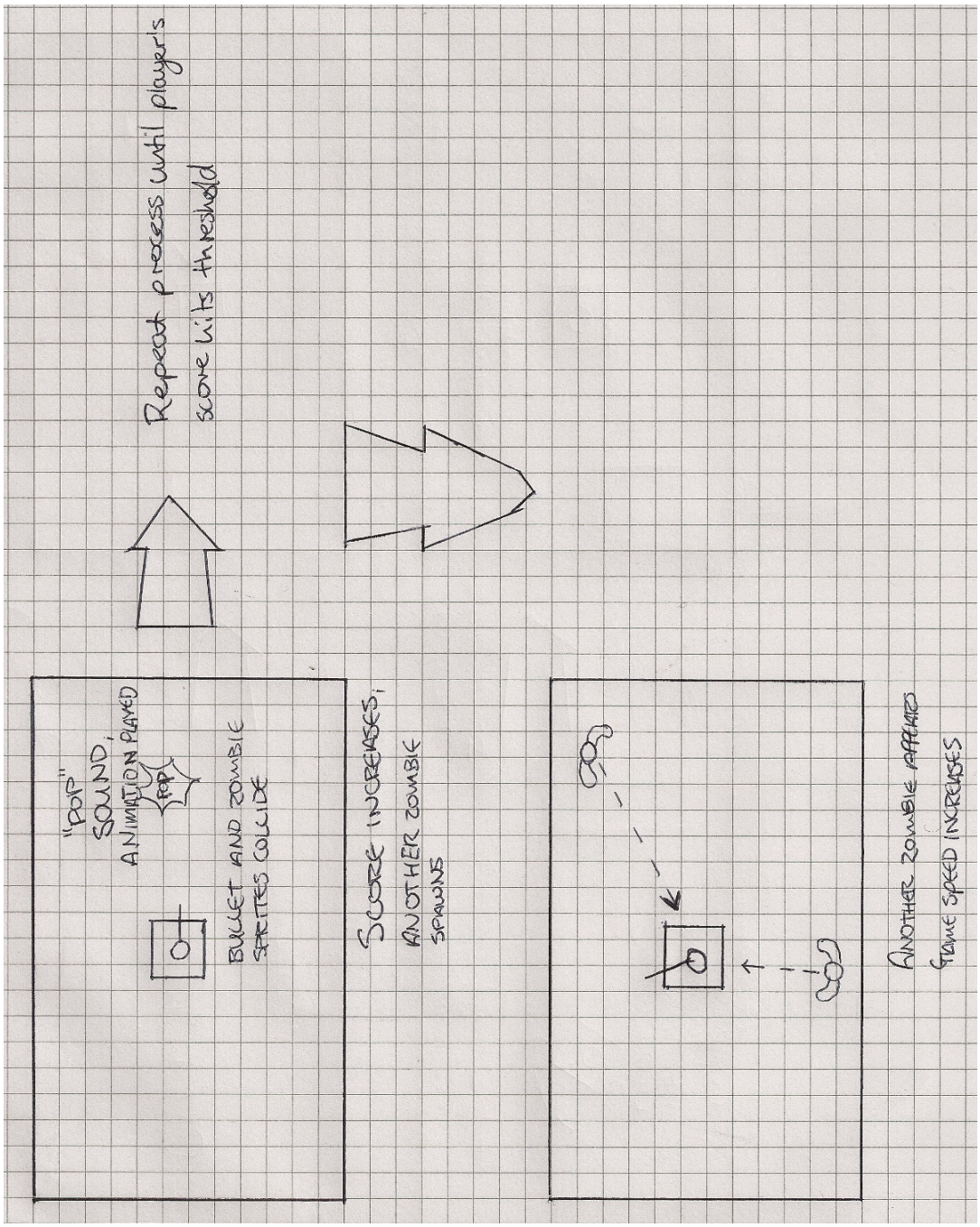

Figure 4: Second storyboard in sequence

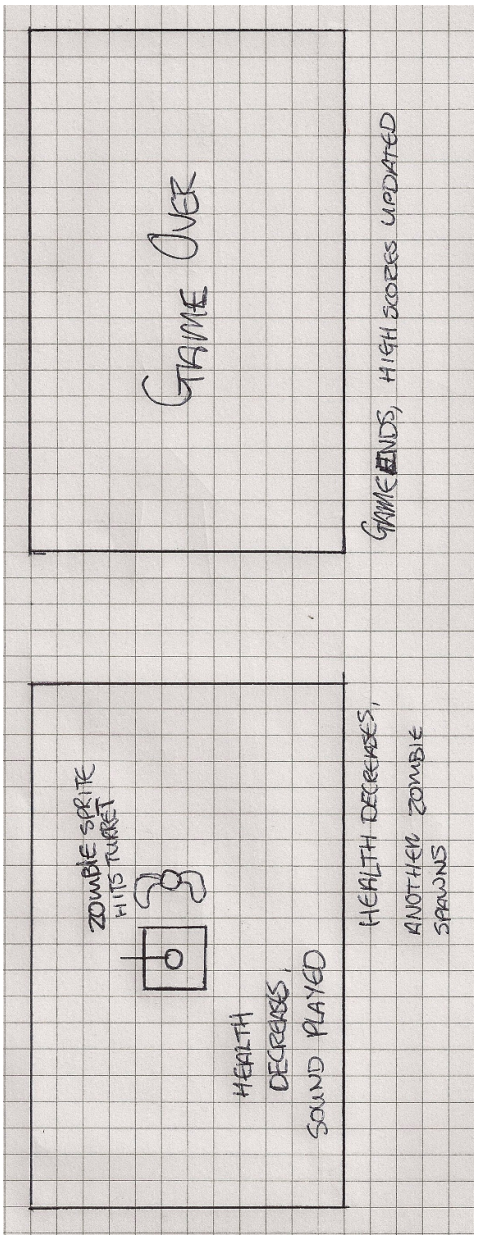

Figure 5: Third storyboard in sequence