



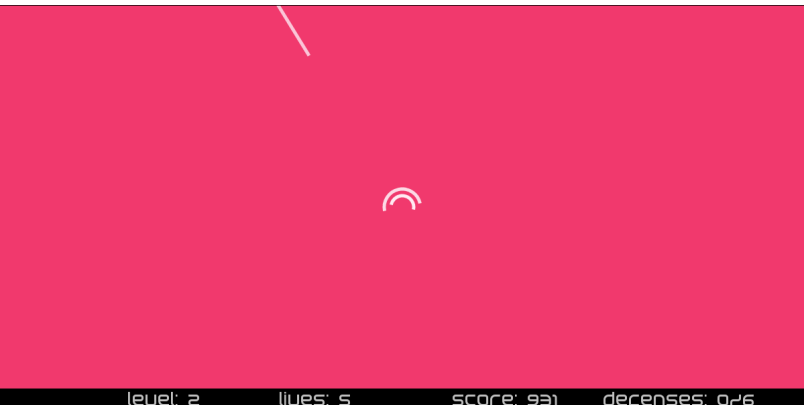
CS252: Interactive Web Programming

Achievement Unlocked

Gideon MW Jones (gij2)

Executive Summary

The idea of the game is to prevent lines from reaching the centre of the screen, where their target is. At your disposal is the ability to draw lines by clicking and dragging with the mouse, if an incoming lines (a threat) intersects with one of your lines (defences) it will be destroyed and thus prevented from hitting the target – which would cause the loss of a life.

Brief storyboard of gameplay	
	<p>A threat line enters the canvas. These lines come in at a rate relative to the level, which iterates every five hundred points (approximately every seventeen seconds).</p> <p>The speed of these lines is also determined by the level. The higher the level, the faster lines come.</p>
	<p>The player draws a defensive line by pressing the left mouse button, dragging the mouse and releasing.</p> <p>The player has a limited numbers of defences they can draw, if they draw too many the first line drawn is removed. This limit is affected by the current level.</p> <p>The number of defences available is detailed in the HUD.</p>
	<p>The threat is destroyed, also destroying the player's defence.</p>

The game is simple and effective and its mechanics reflect that. The player can only essentially do one thing – draw lines.

There is a random element to the angle at which threats come in as well as their speed, this is so not every line comes in at the same speed for a level. The speed is capped by the current level to prevent unfair speeds at early stages. This can also result in no lines appearing in the first level due to their slowness.

Stylistically, the game has gone for a clean, modern look inspired by the games Super Hexagon and WipEout Fusion. Each level consists of six different colours for threats, defences, background and inner and outer target rings. These are effectively randomised, which can lead to some issues with regard to being able to see the lines as there's no moderation of combinations.

The game stores the local high scores as well as a number of achievements, all of which can be viewed in game and reset.

Technical Overview

Regarding technologies used, the game makes use of HTML5 features in terms of both HTML tags and JavaScript functions, such as the audio tag, the canvas and local storage. JavaScript is the obvious choice for language due to the fact it is very widely supported and removes the need for browser plugins. The game uses some common libraries, like jQuery for an API request to get colours for levels from a colour palette website (colourlovers.com) in JSON. Another library the game uses is the Mousetrap library, as it's a simple effective way of binding and unbinding keys with game functions. All libraries are included via their CDNs, as the game is already reliant on an API, it makes little sense to have just this part self hosted.

The game is written roughly following the Model View Controller design pattern. The drive for using such was to separate the aesthetics of the game from the core game engine. By this design, the game can actually run without any view or control, with lines colliding and levels iterating .etc (though it's not very fun that way).

The core mechanic – line collision – is calculated using the maths for an intersection of two lines and then checking if the point of intersection lies within the range of the two lines. This is a fairly complex operation, but is performed quickly enough to be called for each threat for each defence, thirty times a second (the game update and paint loop) without being too demanding or introducing lag.

The game works best on a larger screen, so the player has more chance to see incoming lines. While the game is coded to be fully dynamic in terms of canvas size (things scale well), it uses quite a large canvas – 1200x600. This was deemed suitable as analytical viewing data (provided by Google) shows that large screen sizes are common:

Screen Resolution (pixels)	Views
1366x768	29
1920x1080	24
1280x1024	11
1280x800	8

Software Testing

Thanks to a lot of testing on multiple set-ups (enabled via Twitter), many bugs were ironed out in the process of development. This helped cross-platform compatibility – the game has been tested on Windows 7 and 8 as well as Ubuntu and Android (Jelly Bean) on recent versions of

Chrome (Chromium), Firefox, Internet Explorer and Opera. Data helped focus development on certain browsers,(though it works on most):

Browser	Views
Chrome	61
Firefox	25
Internet Explorer	8
Safari	8
Android browser	5
Opera	3
BlackBerry	1

Older versions of these browsers were found not to interpret some features, such as the sound effects, and some not the canvas at all, this is likely due to the fact HTML5 is not yet a standard. It should be noted that while the game (model) does run and display perfectly fine on mobile devices, the controls do not function properly. The game relies on the onmousedown/onmouseup events of canvas which do not handle touch screen press and drags.

Reflections and Future Work

Further extensions could involve tighter integration of music with gameplay (example: the pulsing of the target to be in time with the music), different types of threats such as ones which need more than one defence to destroy, more achievements and award notifications, power-ups like slowing down time and global leaderboards (would require server side work).

Among other areas where the game could be improved, better level balancing and improved is mobile support are the strongest. If this could be got working well, then the game could be released on mobile devices, such as Android, in a non-native fashion, or even for Firefox OS (which is based entirely on web technologies).

I do believe there's money-making potential in the game. Having not seen a game like it before, it is bound to attract an audience, money could be made simply through using adverts around the game on the web page. Alternatively, if got working on mobile devices, a small charge would be suitable.