# Notes on implementation

## Flash

The initial implementation in Flash took only a few hours to complete, including setup of the project, finding assets and little messing about making it look decent. No specific performance optimisation has been performed, but the game runs at 60fps with negligible cpu load.

Implementation was a single game class, which is loaded by a “main” class. Originally it was the one class, but due to the way Starling works I converted it to match. The main class is only responsible for creating the game class and staging it. I also stuck in a quick-and-dirty fps and memory monitor.

I originally went with vectors for the border, paddles and ball for expediency. This had to be changed later to bitmaps in order to fit with Starling.

## Starling

Converting to use the Starling framework was quite straight forward and would have been more so had I implemented the original using bitmaps rather than vectors. Total time to convert was around 2 hours, but would have been faster had I not had to rewrite the original vector implementation to use bitmaps. The actual conversion process once that was done took only about an hour – and a lot of that was reading documentation.

The Starling framework is all bitmap-based so a new project intending to make use of it would do well to go all-bitmap at the outset. I converted the original to bitmaps rather than sprites and vector drawing.

Interestingly the big performance bottleneck appears to be tracking the mouse position. Just waving the mouse around the stage can halve the frame rate. Not sure what’s going on there.

## HTML/JS

Converting the flash implementation to HTML/javascript was relatively easy and took only a couple of hours. I had to write replicate a couple of utility classes (Sprite and Rectangle) to make life easier, but on the whole it was very nearly a direct copy. Some of the resize operations were made somewhat easier due to the ability to use CSS. That’s probably cheating a little in terms of comparing implementations so I limited the stylesheet to positioning stuff that wasn’t an active game sprite (ball, paddles).

Syntactically it’s annoying to not have access to getters/setters like Flash. The feature exists but isn’t cross-browser (read: doesn’t work in IE yet).

Performance varies by browser but Chrome had no trouble holding 60fps.

## HTML5 / Canvas

A bunch of time reading docs and more time in missing the crucial “update” method on the stage object (which does the draw). Other than that a fairly smooth process. I reused the sprite object from the previous JS implementation to interface with createJS, which meant not having to rewrite much at all.

You get mouse tracking more or less for free. Much like the first Flash version, you can just test mouseY without having to attach a specific listener.

You also get performance metrics in terms of frames per second (FPS) built in.

Total implementation time ~2 hours.