

A WebGL based flight simulator derived from OLAN (One letter aerobatic notation) inputs.

Craig Heptinstall (Crh13)

February 2, 2015

This document describes initial requirements and features proposed for the simulator and a broad set of terms detailing technologies and processes used during the project.

Initial general requirements

The listed general requirements are as follows:

1. Provide a web-implemented tool¹ that allows input of the OLAN characters as a string format, alongside possible click functionality.
2. Relate each notation or set of notations to a certain procedural movement² (rotations, movements etc.).
3. Provide a means of linking up these movements in such a way they produce a fluid manoeuvre.
4. Display this using WebGL³. Libraries⁴ to consider that could help with some of the movements:
 - glMatrix- Javascript library for helping with performing actions to matrices- <http://glmatrix.net>
 - ThreeJS- Another Javascript library, good with handling cameras and different views- <http://threejs.org>
5. Allow user to add different effects such as wind, gravity changes and other physics⁵.
6. Add functionalities of different viewpoints(on-board views, side views) to application.
7. Possibility to add function to save (using local storage?) users different sets of manoeuvres?

¹ None-IE due to WebGL capabilities. Will it use a simple JSON file to store notations?

² Must consider parameters in some of the notations, such as the speed of entry into moves, or the angle of the plane.

³ Begin by initially testing simple shapes to move and fly around, then add textures, and plane structure.

⁴ Are libraries ok to use?

⁵ Could be better to implement these last, as it will be easier to test pure functionality of rolls etc first, then figure out natural physics.

Development environments, testing and bug tracking

To develop the project, I have decided on a set of technologies I wish to use:

- To develop on a Github basis- Easier to maintain, links up to build trackers, good room for documentation, code comments etc.
- Use a Travis build server to run tests after each commit- This can be done automatically, provide me some nice statistics, links up to test libraries well.

- Testing frameworks- I plan to use either libraries such as PhantomJS or Grunt to test my client side code.⁶
- Bug-trackers- For instance inbuilt into Github. Allows me to prioritise higher importance issues. Also helpful for time tracking when adding functionality.

⁶ Need to ensure good de-coupling between data and the shaders etc within WebGL.

Project course specifics

Alongside the functionality of the actual simulator, there are the methods and stages of the project I need to consider:

- Using FDD as a methodology through the process.
 - Using the list given in the first section, make a list of requirements(change these into features).⁷
 - Plan each feature, and design functionality logically and narratively.⁸
 - Implement each feature accordingly, running through coding and testing, then reviewing.
 - Iterate over each feature, until all(or as many as possible) have been completed.
- Document throughout process, any issues, and findings
- Use LaTeX to document⁹

⁷ This could include an overall plan of the project, timing using Gantt charts?

⁸ Sketch-up of plane and angles, and maths behind different transformations.

⁹ Perhaps initial tests or large sets of data can be done by hand and transferred later