Create a WebGL based application that calculates and visualises the OLAN(One letter aerobatic notation) catalogue in the form of a 3D plane performing aerobatic manoeuvres

---

Report Name	Outline Project Specification
Author (User Id)	Craig Heptinstall (crh13)
Supervisor (User Id)	Neal Snooke (nns)
Module	CS39440
Degree Scheme	G601 (Software Engineering)
Date	February 3, 2015
Revision	1.0
Status	Release

---

1 Project description

My major project will be looking into the implementation of a WebGL flight simulator, though more importantly it will be based from manoeuvres outlined in the OLAN [4](One letter aerobatic notation)/Open Areo [7] (Newer, updated and open source version of OLAN) format. The simulator should be web based, so run through any WebGL compatible browsers(Chrome, Opera, Firefox).
In more specific detail, the simulator should firstly allow for a range of different inputs(as string values) each of which should represent different manoeuvres according to OLAN. These will be space separated, and the previous move should link to the next in the most fluid means possible. Once the string of notations have been read in, then the system will use a list of predefined instructions from a JSON file which will allow each of the notations to be converted into a set of broken procedural movements(rotations, flips, angled movements).
Currently, there is a standard for drawing out these manoeuvres known as Aresti [8]. In addition to this, there is already current systems that allow input of OLAN, and ribbon diagrams are produced. These ribbon diagrams entail a 3d ribbon shape of the moves, showing where both tips of the wings would be on a plane. However, I am to improve on this, by making my application show the moves live, in a more aesthetic format. The ribbon will not be shown, but instead a plane will be shown flying the course defined, with smoke trails showing where it has already flown.
To help the system achieve the different manoeuvres and physics required to perform them, I will be considering the use of a set of libraries such as Three.js [9] and glMatrix [6]. Both these will provide some easier predefined methods allowing to perform some of the movements described previously.
Alongside the main functionality, additional features such as different camera angles, allowing the saving and loading of entire diagrams and adding pyshics will be considered. These though will be only be implimented once a good basis for the program is established. Overall, the main challenges that I will come across during this project will be issues in turning each OLAN figure into the appropriate translation in terms of the plane. With the project delivering a WebGL based product, it is easy to see the vast amount of the project will be created in Javascript.

2 Proposed tasks

In order to perform my project, I can break down the process into a selection for different tasks:

1. Read up on the OLAN and Aresti [8] notations- This will involve looking through the various possible manoeuvres that aerobatic planes can fly.

2. Investigate various WebGL technologies, especially Three.js, to see what forms of movements are possible using the library. For this I can spend plenty of time looking at other projects [5] around the web to see how certain transformations are done to objects on a canvas.

3. Look into how the site should look on completion- This will consider the size of the canvas on the site, and if mobile users should be able to view the product.

4. Look at the example OLAN to Aresti online program- Analyse each of the different ribbon diagrams, and see how different OLAN figures relate to one another. This key process will allow me to see what primary moves the entire collective notations can be made from. This could be compared to how every colour can be created from red, green, blue.

5. Create a set of functions relating to the findings to allow the program to create some form of ribbons, which can then be 'flown" by an object(a plane). This should utilise some form of stored list of moves and required actions.

6. Combine these into a clean, good looking WebGL product, to allow for inputs of OLAN, and controls such as camera angles.

7. Throughout the process, blog daily to log each task and time taken- This will allow an easier to create set of graphs, tables and references to project time taken on certain processes.

3 Project deliverables

Following on from the previous section regarding the list of proposed tasks, I can relate these to a list of deliverables that the project shoudl produce. Although the simulator should primarily be an internet application acessable through a web browser, there will be a set of other items that will be created. I have chosen to create a blog, that will act like a form of self- documentation that should build up to create a work log. This will eventually be usefull in terms of relating it to a Gantt chart to see how well I am following the project plan. However, because I have decided on using an FDD (feature driven development) methodology to create my project the Gantt chart should allow for more flexibility.
Before I can impliment the simulator, I will also have to deliever a number of different planning documentation pieces. The first of these will be an analysis of botht the Aresti and OLAN langauges, which will allow me to understand how each manoeuver is broken down, and which primary elements can be combined to make other moves. After discovering these, it will be much easier to then create more broader methods to allow easier creation of more complicated simulations.
Alongside this, a table shoud be produced to further explain each move, and break down any where parameters should be needed. For instance, the size of a loop can be depicted by a radius size and height. Again, this will furtehr broaden the methods I will need to code thus producing much more flexible algorithms that can handle the maximum amount of different manoeuvers. The table should form a basis for my planning documentation. My project plan will then begin the FDD approach I am wanting to take, starting with a break down such as a list of features I would like the simulator to contain on completion. Once this is generated, I will then be able to break these into groups, each of which will be designed, implimented and tested.
Together with the blog, my implementation environment will be a GitHub based one, to allow for ease of working on the project in different places/ on different systems, and also to integrate it with any build servers I may want to use. In this case, I will be using Travis CI [2] to build my tests (which will be written in QUnit [3] and run with Grunt [1])which will be useful for finding out any bugs or errors I may create in the process of implementing certain features.
To generalise on the final delverable and to come back to the first section of this paper, there should be a Web based application allowing the uer to input OLAN and hit a button allowing them to view (at different angles), control physics, speed and size of a set of defined manoevers with the functionality to save and load entire routines.

Annotated Bibliography

[1] B. Alman, "Gruntjs task runner," http://gruntjs.com, Sept. 2011, accessed January 2014.

GruntJS task runner for running QUnit tests.

[2] T. CI, "Travis ci for my github project," https://travis-ci.org/craighep/Dissertation, Jan. 2014, accessed January 2014.

Travis CI set up for my dissertation GitHub project. Shows status of build after running QUnit tests.

[3] T. J. Foundation, "Introduction to unit testing," http://qunitjs.com/intro/, Jan. 2014, accessed January 2014.

QUnit introduction and example code for performing QUnit tests on Javascript.

[4] M. Golan, "Olan- one letter aerobatic notation," http://web.archive.org/web/20080420201845/http://www.aerobatics.org.il/olan/welcome.php, Apr. 2008, accessed January 2014.

Introduces the concept of OLAN, and the different range of notations. Please note, this has been taken down, due to copywrite reasons, so OpenAreo has replaced it. See Open Areo reference for details.

[5] D. S. Lyons, "Three.js example and introduction," http://davidscottlyons.com/threejs/presentations/frontporch14, 2012, accessed January 2014.

A demo and introduction to the Three.js library. Cool demos of different affects within a powerpoint style presentation explaining how to do roatations, movements, orbits, lighting etc.

[6] C. MacKenzie, "Openaero aerobatic software," http://glmatrix.net, Nov. 2011, accessed January 2014.

glMatrix library for WebGL products. Allows easier transformations to matrices and handling of vertices.

[7] R. Mass, "Openaero aerobatic software," http://www.openaero.net/, 2012, accessed January 2014.

An online software that creates sequenced diagrams based on OLAN. Used to find out what each notation looks like, and how they link up.

[8] L. Richardson, "Openaero arobatic software," http://www.slopeaerobatics.com/articles/an-introduction-to-slope-aerobatics/aresti-notation-tutorial/, Dec. 2011, accessed January 2014.

Explanation of the Aresti diagrams, including how parameters are used.

[9] G. user:Mr.Doob, "Three.js library," https://github.com/mrdoob/three.js, 2009, accessed January 2014.

The Three.js library, which I will be using throughout my project. Allows for a vast array fo transfomation and effects.