

**A WebGL-based aerobatic visualiser using the
OLAN(One letter aerobatic notation) catalogue
to provide users with a means of generating
interactive 3D representations of manoeuvres.**

Final Report for CS39440 Major Project

Author: Craig Heptinstall (crh13@aber.ac.uk)

Supervisor: Prof. Neal Snooke (nns@aber.ac.uk)

29th March 2015

Version: 1.0 (Draft)

This report was submitted as partial fulfilment of a MEng degree in
Software Engineering (G601)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature

Date

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature

Date

Acknowledgements

I am grateful to...

I'd like to thank...

Abstract & Background

The aim of this project is to create a 3D representation of aerobatic manoeuvres, primarily using the OLAN notation(or one letter aerobatic notation). Both real-scale and remote control aerobatic planes and helicopters use the notation to describe a set of manoeuvres in an overall routine, usually including translating the notations into Aresti symbols. The Aresti symbols came before OLAN, but OLAN was developed to make it possible for pilots to write down quickly their planned routines.

The primary element of this project will involve allowing users to insert their notated routines into the application(via an input box on a web-page) thus producing first the ribbon shapes of the manoeuvres, followed by the ability to see a craft fly the route. Although there is only a finite amount of manoeuvres possible from the Aresti catalogue, each OLAN notation can have its own parameters. This can range from entry length into a loop or turn, to the number of rolls in a section of a manoeuvre. The application will also need to be taking into account flight speeds, and possibly wind and gravitational effects. Due to a long list of possible features, the application will be created in a feature-by-feature means (FDD).

WebGL will be the language used to create this application, with hopefully object-orientated Javascript to power the application. Libraries helping towards the graphical/ visual side of the application may be required to give greater flexibility and better aesthetic value. All considerations will be found in the analysis and design stages of this report.

CONTENTS

1	Background & Objectives	1
1.1	Background	1
1.2	Analysis	1
1.3	Process	1
2	Design	2
2.1	Overall Architecture	2
2.2	Some detailed design	2
2.2.1	Even more detail	2
2.3	User Interface	2
2.4	Other relevant sections	2
3	Implementation	3
4	Testing	4
4.1	Overall Approach to Testing	4
4.2	Automated Testing	4
4.2.1	Unit Tests	4
4.2.2	User Interface Testing	4
4.2.3	Stress Testing	4
4.2.4	Other types of testing	4
4.3	Integration Testing	4
4.4	User Testing	4
5	Evaluation	5
	Appendices	6
A	Third-Party Code and Libraries	7
B	Code samples	8
2.1	Random Number Generator	8
	Annotated Bibliography	11

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

During the beginning

1.1 Background

1.2 Analysis

1.3 Process

Chapter 2

Design

You should concentrate on the more important aspects of the design. It is essential that an overview is presented before going into detail. As well as describing the design adopted it must also explain what other designs were considered and why they were rejected. The design should describe what you expected to do, and might also explain areas that you had to revise after some investigation. Typically, for an object-oriented design, the discussion will focus on the choice of objects and classes and the allocation of methods to classes. The use made of reusable components should be described and their source referenced. Particularly important decisions concerning data structures usually affect the architecture of a system and so should be described here. How much material you include on detailed design and implementation will depend very much on the nature of the project. It should not be padded out. Think about the significant aspects of your system. For example, describe the design of the user interface if it is a critical aspect of your system, or provide detail about methods and data structures that are not trivial. Do not spend time on long lists of trivial items and repetitive descriptions. If in doubt about what is appropriate, speak to your supervisor. You should also identify any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc. Some example sub-sections may be as follows, but the specific sections are for you to define.

2.1 Overall Architecture

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

The implementation should look at any issues you encountered as you tried to implement your design. During the work, you might have found that elements of your design were unnecessary or overly complex; perhaps third party libraries were available that simplified some of the functions that you intended to implement. If things were easier in some areas, then how did you adapt your project to take account of your findings? It is more likely that things were more complex than you first thought. In particular, were there any problems or difficulties that you found during implementation that you had to address? Did such problems simply delay you or were they more significant? You can conclude this section by reviewing the end of the implementation stage against the planned requirements.

Chapter 4

Testing

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully. Have you tested your system on real users? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix. The following sections indicate some areas you might include. Other sections may be more appropriate to your project.

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree. There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved. Review the discussion on the Evaluation section from the lectures. A recording is available on Blackboard.

Appendices

Appendix A

Third-Party Code and Libraries

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

As an example, you might include a definition such as:

Apache POI library The project has been used to read and write Microsoft Excel files (XLS) as part of the interaction with the clients existing system for processing data. Version 3.10-FINAL was used. The library is open source and it is available from the Apache Software Foundation [2]. The library is released using the Apache License [1]. This library was used without modification.

Appendix B

Code samples

2.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs [6].

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
    /*-----*/
    /* Minimum Standard Random Number Generator */
    /* Taken from Numerical recipies in C */
    /* Based on Park and Miller with Bays Durham Shuffle */
    /* Coupled Schrage methods for extra periodicity */
    /* Always call with negative number to initialise */
    /*-----*/

    int j;
    long k;
    static long idum2=123456789;
```

```
static long iy=0;
static long iv[NTAB];
double temp;

if (*idum <=0)
{
    if (-(*idum) < 1)
    {
        *idum = 1;
    }else
    {
        *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7; j>=0; j--)
    {
        k = (*idum)/IQ1;
        *idum = IA1 *(*idum-k*IQ1) - IR1*k;
        if (*idum < 0)
        {
            *idum += IM1;
        }
        if (j < NTAB)
        {
            iv[j] = *idum;
        }
    }
    iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
    *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
    idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
    iy += IMM1;
}
```



```
if ((temp=AM*iy) > RNMx)
{
    return RNMx;
}else
{
    return temp;
}
}
```

Annotated Bibliography

- [1] Apache Software Foundation, “Apache License, Version 2.0,” <http://www.apache.org/licenses/LICENSE-2.0>, 2004.

This is my annotation. I should add in a description here.

- [2] ———, “Apache POI - the Java API for Microsoft Documents,” <http://poi.apache.org>, 2014.

This is my annotation. I should add in a description here.

- [3] H. M. Dee and D. C. Hogg, “Navigational strategies in behaviour modelling,” *Artificial Intelligence*, vol. 173(2), pp. 329–342, 2009.

This is my annotation. I should add in a description here.

- [4] S. Duckworth, “A picture of a kitten at Hellifield Peel,” <http://www.geograph.org.uk/photo/640959>, 2007, copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

- [5] M. Neal, J. Feyereisl, R. Rascunà, and X. Wang, “Don’t touch me, I’m fine: Robot autonomy using an artificial innate immune system,” in *Proceedings of the 5th International Conference on Artificial Immune Systems*. Springer, 2006, pp. 349–361.

This paper...

- [6] W. Press *et al.*, *Numerical recipes in C*. Cambridge University Press Cambridge, 1992, pp. 349–361.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content*.

- [7] Various, “Fail blog,” <http://www.failblog.org/>, Aug. 2011, accessed August 2011.

This is my annotation. I should add in a description here.