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

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

Before commencing the design of the application and the project planning, it is important to have analysed what I hope to have achieved at the end of my project time, and also what steps I will need to be taking to implement each feature. As I will mention later, using FDD will play a big part of how I shape my project and create each feature whether it be by priority, size or difficulty. This section details my understanding for my project requirements, steps I am going to need to take, and also the first sections of an FDD project; developing an overall model and building the list of requirements and features.

1.1 Background

The choice of undertaking a project such as this one was due to two combining factors: maths and an interest to learn graphical programming. The fact that this application will require me to learn graphics, and how to implement visual effects representing the requirements of the project in ways completely new to myself. As graphics is something I have not had much to do with in the past, this project appears both exciting and daunting task due to the learning curve I will need to take. As for the maths factor, I can assume quite a lot of maths will be involved(especially for creating curves, rolls and turns along most of the manoeuvres) which I enjoy learning about.

In terms of the history of the topic, OLAN was originally developed by Michael Gorden in **2006** and was designed to provide shorthand notation for pilots planning out aerobatic routines without having to draw out the full Aresti diagrams. In recent years, the OLAN notation became used much more until because of licensing issues with the original owner was taken offline. Because of this, a new form of the notation has been created in a more open source way paving the way for applications such as this project's intended aim. Although in this report and my planned application itself will be still referring the notation as OLAN, the new re-make of the language is known as the **'OpenAero language'**. This is based off of the original, yet is open and allows anyone to use it. In combination with this, the creators of the OpenAero language also developed a web-based **application** that allows the conversion of the notations to 2D Aresti diagrams. This is somewhat similar to what I hope to achieve, but alongside plenty more features most importantly the ability to see a plane perform the moves.

As for Aresti, named after its conceiver Jos Luis Aresti Aguirre **https://www.aerobatics.org.uk/aresti** is the diagram format that OLAN achieves, and represent informative diagrams showing the shape

of the routine, direction of travel, rolls and sharpness of turns. Aresti diagrams also can include angles or turns, ranging from 90 degrees to 270. Each diagram usually has a name, relating normally to the shape of the manoeuvre, though some are more commonly known to pilots rather than the regular user. The OLAN notation for each diagram usually attempts to try describe the manoeuvre with the letter used, such as 'o' for a loop, or 'z' for a shark tooth. The full list of manoeuvres, including their OLAN notation and full name can be found on the **OpenAero site**.

Upon starting this project, several meetings with the project supervisor are planned each providing more detail of the initial requirements. Each of the meetings will be found in the Gantt for the project, with a smaller document in the **appendix** showing the results of each meeting. Because I have already attended several meetings at this stage, I can provide a fairly accurate list of initial requirements of the application.

The initial required application can be broken down into an extensive(but less detailed) list of main functional requirements. These are as follows:

1. Provide a web-implemented tool that allows input of the OLAN 1 None-IE due to WebGL capabilities. Will it use a simple JSON file to store notations? characters as a string format, alongside possible click functionality.
2. Relate each notation or set of notations to a certain procedural movement(rotations, movements etc.).
    - Must consider parameters in some of the notations, such as the speed of entry.
3. Provide a means of linking up these movements in such a way into moves, or the angle of the plane. They should produce a fluid manoeuvre.
4. Display this using WebGL. Libraries to consider that could help. Begin by initially testing simple shapes to move and fly around, then add textures, and plane structure.
5. Are libraries OK to use? 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
6. Allow user to add different effects such as wind, gravity changes and other physics. 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.
7. Add functionalities of different viewpoints(on-board views, side views) to application.
8. Possibility to add function to save (using local storage?) users different sets of manoeuvres?

The list shown above also has an **accompanying** report which I created after my first project meeting. This report includes the list of initial requirements, alongside footnotes, and also a detailing of the methodology and process I plan to follow.

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.