OLANdroid

Gideon MW Jones **Computer Science Department, Aberystwyth University**

Project aim

OLANdroid (a name derived from the encoding system and the platform) aims to provide a tool for the visual, 3D modelling of the acrobatic flight of planes. These are dramatic flights flown by full-sized and, more commonly, model aircraft.

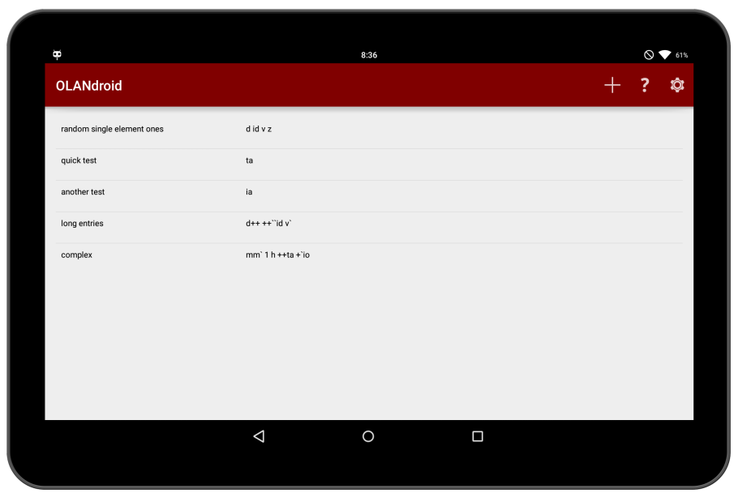
These aerobatics are broken down into a series of manoeuvres, varying in complexity, for example: a straight line up and a loop-the-loop. These can be expressed with either diagrams or letters – the Aresti and OLAN (One Letter Aerobatic Notation) systems respectively. The project should provide functionality to interpret a flight description (OLAN was chosen), build a flight and visualise it.

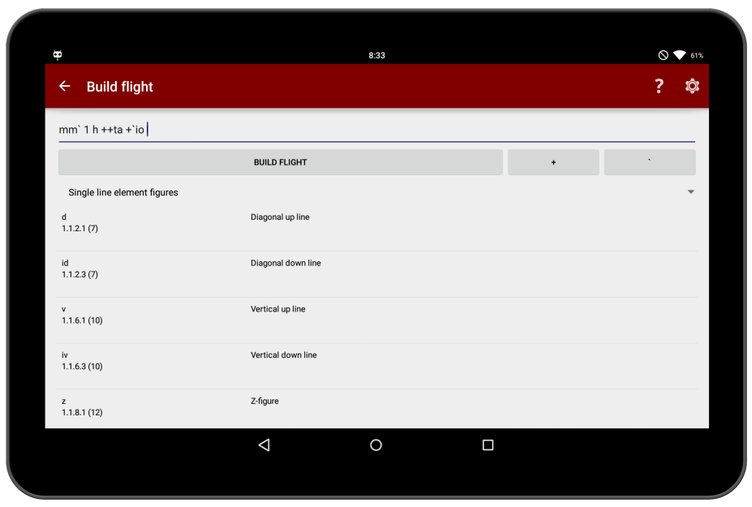
The visualisation aspect of the project aims to deliver a (visually) simplified version of the flight, demonstrating the way the aircraft moves during the flight. Alongside the full flight, an animation of flying it provides a useful description.

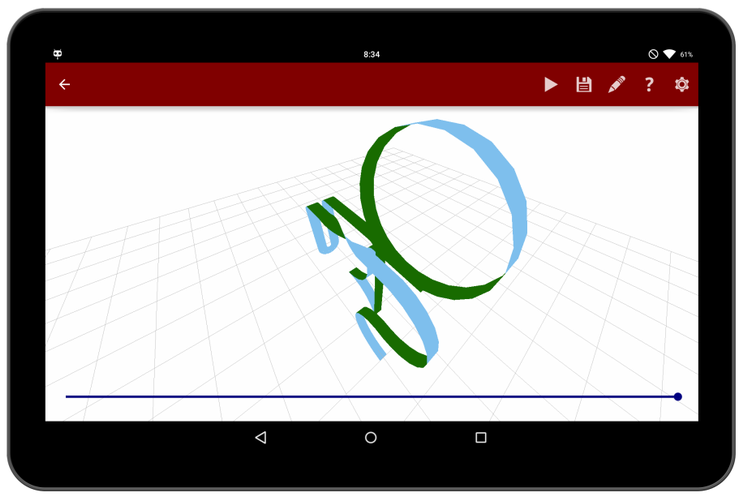
Alongside these features, the project offers some extra functionality to make using the system easier. This includes things like offering a catalogue of manoeuvres and the saving/loading of flights.

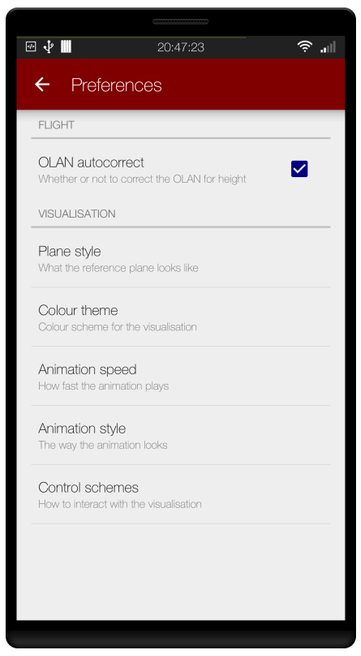
The platform this project is delivering this functionality on is Android – a portable format allowing for use anywhere.

Overall, the aim of this project is to provide a tool for building and visualising realistic flights. It is aimed at beginner hobbyists who wish to understand and learn various flying manoeuvres.









Examples OLANdroid in action – a flight being built and visualised with various options.

Current progress

This project has been under development for about two and a half months. The primary goals of developing an representation of a manoeuvre, building a catalogue, interpreting a flight description and rendering a 3D model of the flight have been more or less completed.

OLAN strings can built by selecting manoeuvres from a catalogue built up of about forty manoeuvre (including variants). In accordance with the OLAN system, the various modifiers available are implemented and these modify the manoeuvres.

Flights are built by interpreting this OLAN string, and modelled in fairly abstract way, consisting of a coloured band which follows the path a plane would take. It can be animated through using two different methods – a progressing band starting with nothing and building up the flight, and a 'flying wing' following the flight with a small section highlighted. The model of the flight can explored in the 3D space with rotation, zooming and movement around the environment.

The application also supports the saving, loading and editing of flights (associated with names) so users can build and develop flights for taking out into the field. Other extra things include an alternate control schemes (one and two finger touch), various different colour schemes for the visualisation as well as a choice of plane style – a grid or a textured grass surface.

Useful links

- The project on GitHub, with a full series of development diary blog posts in the wiki: http://github.com/GideonPARANOID/olandroid
- The closed beta community on Google+, joining gives access to the app for testing on Google Play: http://tinyurl.com/olandroidbeta
- OpenAero, a valuable tool for building tools with OLAN and getting Aresti diagrams: http://openaero.net

Technical detail

As an Android application, this project is written in a combination of Java and XML, with a small amount of GLSL for the shaders.

Manoeuvres are represented internally using a series of components. These components are small pieces of flight matching the way an aircraft moves, with elements of forward motion and rotation on the three axes of pitch, yaw and roll. The bounds of these components' variables (how much they change direction etc.) is derived from analysing the manoeuvre catalogue and using the highest common factor of movement – about 15 degrees. These components can also easily be marshalled into/from XML, making building an extensible catalogue a fairly simple task, and maintaining a separation between application and data. The manoeuvre catalogue in the application parses this XML into an associative array, making them addressable by their OLAN figure.

An example flight encoded in OLAN:

d 6% ++```id+ 3 v``

A rough description of the system:

{number% (1% is scale factor 1) full scaling (optional)} {pluses, entry length}{backticks, first group scaling (inverse)}{OLAN figure}{backticks, second group scaling (inverse)}{pluses, exit length}

The flight descriptions are interpreted using regular expressions and groups. Most can be fairly simply captured by splitting the flight description by spaces and dealing with each figure separately, but some are more complicated, such as full figure scaling. A flight may be valid OLAN, but be impossible to fly – the flight goes below the horizon. These can corrected by analysing the components of a flight and adding more vertical movement if its lowest point is below the horizon.

The visualisation aspect of the project is built with OpenGL ES 2.0. Due to the simple nature of the shapes being drawn, and the relatively low performance of devices it will be ran on, the visualisation is kept simple. Each component of a manoeuvre is also a drawable quadrilateral (consisting of two triangles), built in 3D space with points calculated from the pitch, yaw, roll and length of the segment. It also contains a 4x4 matrix defining the transformation from the beginning of the component to the end of it. Using this, components can be drawn sequentially by transforming the origin by this cumulative matrix.

Continued work

While the project has more or less fully satisfied the brief, there are a number of elements which could do with some more polish/features which could be implemented soon:

- Improve the test coverage – something that is difficult to do completely with code which has a visual aspect. As much as can be tested should be tested to ensure stability and uncover potential problems.
- Completing the “realistic” visualisation mode still needs to be done. The basis for this, with the ground texture, is built. A model plane needs to be constructed and support for animating it through the flight needs to be implemented.

Further extension

There are a number of different ways this project can be extended and built upon.

To make the application of greater value to users more familiar with the Aresti system, providing a manoeuvre catalogue and input method which supports the system would be a useful.

Alongside the visualisation, providing some reverse kinematics, demonstrating the controller movements (of the two sticks) necessary to fly the manoeuvre, would be a valuable feature. This could be related to the components, due to their inspiration stemming from the movement and change of direction at a point.

Reinventing the user interface to adhere to Google's latest stylistic Android guidelines – material design – would be nice. It was shied away from due to lack of official library support yet.