The FlightGear Flight Simulator Project

Presented by David Findlay

Abstract:

The FlightGear Flight Simulator Project is an open source project to develop a free flight simulator with an emphasis on realism for training purposes. The project has been running for around 5 years. currently features highly detailed terrain, accurate flight dynamics, instrument panels and visual models for aircraft and a structure that enables users to customise and add their own models to the simulator. Future areas of work will include further improvements to the flight dynamics systems anc visuals of FlightGear

Overview

The FlightGear Flight Simulator project began in 1996, after a discussion by a group for netizens. David Murr wrote the first proposal for the simulator, and work began shortly afterwards. David Murr then left the project, and handed it on to Eric Korpela. Since the middle of 1997 the project has been coordinated by Curt Olson, who works at the University of Minnesota. The project has continued to gain momentum since then, and now there are around 10 programmers who regularly contribute code, as well as a group of many others who make contributions from time to time. The simulator is licensed under the GNU General Public License.

The initial flight dynamics model was based on the NASA Larcsim, but more recently Jon Berndt has been developing a new model which is far more versatile and extendable. It also has many features previously not available in a consumer flight simulator. The current list of features include:

- High resolution terrain
- Textures in a variety of formats including PNG
- Accurate positions of astronomical objects such as stars, the moon and the planets
- Completely customisable head up display, instrument panel, aircraft visual models, flight dynamics models and scenery
- Portable to any platform that supports OpenGL.
- Networking IO system, allowing multiple displays across multiple machines as well as multiplayer on local area networks
- A rudimentary weather model
- Audio subsystem
- Worldwide scenery including support for rivers, roads and night lighting.
- Property manager allowing control of various options
- Autopilot and instrument flight support

More work is needed in several areas of the simulator to improve the "finish" of the program. Many improvements could be made to the GUI and documentation that would make the simulator much easier to use for beginners. The structure of the simulator allows easy addition of new components, and many things can be achieved without requiring a line of code. The property manager provides one place for accessing all the simulator's variables that may be of use to panel and addon designers.

Requirements and Installation

FlightGear Flight Simulator requires a fast video card to run. It relies heavily on 3D accelerated OpenGL. Mesa GL can be used with XFree86 3.3.6, or 4.x, or if using a Nvidia card the Nvidia GLX drivers work with XFree86 4.x. FlightGear works best with at least 32 megabytes of video RAM, however the more you have the better it can run, especially with photorealistic scenery. A joystick is recommended. Rudder pedals are also very useful if you can afford them.

Installation with most distributions is easy. Mandrake, Redhat, Suse and Debian all include FlightGear as part of their standard distributions. If FlightGear is not included in your distribution, you can download the source code from the the FGFS web site and compile it yourself. The code is also portable to a number of other operating systems, including Microsoft Windows and Solaris.
FlightGear has the following dependencies:

- zlib
- GLX and OpenGL headers
- glut
- Plib (plib.sf.net)
- SimGear (www.simgear.org)
- Flightgear base package

Plib is a portable game development library which includes a graphics engine, geometry library, font rendering, user interface and an input library. FlightGear makes extensive use of this library. This approach allows us to concentrate on writing the simulator itself, instead of having to spend a lot of time developing our own libraries. The SimGear package is a collection of various components that can be used for simulator development. It is being developed by the FlightGear team, but is a separate package so as to allow other projects to use it as a "simulation kernel" in the future.

Basic Usage

Once installed running FlightGear Flight Simulator running the program is easy. Provided the executable is in your path, running "runfgfs" should start the simulator in the default situation. This places you on a runway at San Francisco International Airport in a Cessna 172R with the engine shut down. This may soon change to reflect better realism, i.e. not many people start their plane up on the runway. The following procedure allows you to start the engine and take off:

1. Raise mixture(red knob) to full(pushed in).
2. Raise throttle(black knob) to 25% in.
3. Click on the right hand side of the magneto key three times, then middle click to start engine. Whe the engine starts middle click again to turn off the starter.

Figure 1. Cessna 172 Panel

Taking off requires the release of the brakes by pressing the B key, dropping the flaps to 10 degrees and raising the throttle to full. Once in the air various unique features of FlightGear come into play. Firstly FlightGear's JSBSim flight dynamics model simulates ground effect and the P-Factor(caused by rotating airflow of the propeller). This will cause your aircraft to climb quickly and also have the tendency to rotate to one side. It is easiest to fly using a joystick, however it is possible to fly using only the keyboard. Rudder pedals are recommended for training. Trimming the airplane so that it will fly level is achieved using the 7 and 4 keys on the numpad.

Currently only a few options are available in the menu, such as setting the autopilot, or taking screenshots. This should improve in future versions however. Most options require use of the command line. Commonly used commandline options include:

```
- -aircraft=abcd Specifies the aircraft model to load. Default is c172. Alternatives available depen
the flight model chosen.

- -fdm=abcd Select the core flight model . Options are jsb, larcsim, magic, external, balloon. Defau
value is larcsim. Magic is a slew mode. Balloon is a hot air balloon.

- -airport-id=ABCD: If you want to start directly at an airport , enter its international code, i.e.
for JFK airport in New York etc. A long/short list of the IDs of the airports being implemented can
found in /Flight Gear/Airports. You only have to unpack one of the files with gnuzip. Keep in mind,
need the terrain data for the relevant region, though!
```

- -start-date-gmt=yyyy:mm:dd:hh:mm:ss: Specify a starting time and date. Uses your system time.

- -start-date-gmt=yyyy:mm:dd:hh:mm:ss: Specify a starting time and date. Time is Greenwich Mean Time

- -start-date-lat=yyyy:mm:dd:hh:mm:ss: Specify a starting time and date. Uses local aircraft time.

- -protocol= medium,direction,hz,medium_options,...: I/O options, useful for running FlightGear over network or in interaction with external programs (preliminary). protocol can be native, nmea, garmin rul, pve..., medium has the values serial, socket, file..., direction can be in, out, bi, hz is the of times to process channel per second. Should be tried by experts only.

Options in the properties manager can be set using the following option:

```
--prop=/sim/myoption
```

In flight a large number of shortcut keys is available. An exhaustive list of these is available on the FlightGear website, but the most commonly used ones are:

g - Gear up/down
Page Up - Throttle Up
Page Down - Throttle Down
p - Pause
h - hud on/off
v - toggle external view mode

Landings can be difficult at first, but are a very important part of the simulation experience. As you approach the ground the ground effect comes into play and makes it very easy to drift to far down the runway. Another common error is overcorrection when lining up with the runway. During crosscountry flights the highly detailed terrain allows practice of all elements of Visual Flight Rules(VFR) navigation, including pilotage and dead reckoning. The worldwide navigation beacon database allows instrument flight(IFR) to any location in the world and is ideal for training.

Internal Structure

Flight Dynamics

The flight dynamics system in FlightGear is designed to allow easy addition of new modules. Currently there are 4 primary flight dynamics modules, JSBSim, Larcsim, YASim and Balloon. Larcsim is a product of the NASA Langley Research Centre and was the original FDM used by FlightGear. One of the disadvantages of Larcsim was that it only simulated one aircraft, the Navion. This problem was somewhat overcome by a team at the University of Illinois who are researching aircraft icing. They added code to allow Larcsim to simulate a variety of aircraft, but unfortunately the model can become unrealistic and unstable with large aircraft.

The JSBSim project was started by Jon Berndt with the goal of overcoming these problems and is now the default FDM in FlightGear. It has been designed with a purpose of being useful to students of aerospace engineering in learning concepts such as flight dynamics, simulation, propulsion and flight control. The design allows almost any aircraft to be added without writing any code. It uses XML configuration files to achieve this. Currently JSBSim comes with models for the Cessna 172, Cessna 310 and the X-15 rocket plane.

YASim is a recently developed FDM being coordinated by Andy Ross. It has been developed to counter one of the major difficulties in creating aircraft models for JSBSim, the lack of enough data. YASim allows the use of values from aircraft manuals, rather than requiring the input of complex stability and momentum data. This means a realistic model can be created very quickly, using data available in standard aircraft specifications.

The Balloon model is a specially designed module to allow the simulation of hot air balloons. Unfortunately at this time wind is not factored into this, so it isn't much use apart from going up and down. Other modules include a magic carpet mode that allows the user to move around freely as if on a magic carpet that can hover anywhere.

The external module allows FlightGear to be connected other simulation models as a visual system alone.

Visuals

The Plib library Simple Scene Graph is used as FlightGear's graphics engine. The engine includes such features as frustum culling and level of detail. The use of Plib also allows FlightGear to load almost any 3D model or texture format. Unlike most consumer flight sims, the FlightGear earth is a truly spherical model made up of 648 trapezoidal shaped tiles. This approach allows flight all the way to the north and south poles. The sky consists of a 12 sided sky dome that is coloured to reflect the time of day. At night FlightGear renders several hundred bright stars, planets and the moon on th sky in their correct positions. This provides the advantage of being able to use celestial navigation if your instruments fail.

The HUD(head up display) and panel are both fully configurable using XML. Work is also underway to automatically render taxiway signage and other objects. It is also possible to add static models o buildings or other aircraft to scenery using a text configuration file.

Networking

Network IO is a key feature of FlightGear. One machine can run the flight dynamics model, then export the data to any number of other machines running FlightGear. These machines can then display other views of the simulation. This is commonly used in homebrew cockpit simulations. FlightGear is the only consumer flight sim that supports this feature. The interface also allows any add on program to access all of the simulator variables and allows them to be controlled. So far the only application that uses this feature is the Atlas program, which displays a moving map showing the aircraft's location on a synthetic chart.

The networking IO ability of FlightGear also extends to multiplayer flight. Using the FlightGear daemon it is possible to have multiplayer LAN parties with FlightGear.

To set up a multiview visual system the following commands are used:

On the main box: runfgfs --native=socket,out,60,s1,5500,udp # s1 is the ip address of the client box

On the clients: runfgfs --native=socket,in,60,,5500,udp --fdm=external

There is no arbitrary limit on the number of client machines that can use this arrangement. Just add an extra --native option to the server commandline arguments for each extra box. Multiple video cards in a single machine is slow compared to this option, because the cards would have to compete for bus bandwidth.

Time Management

Many of the tasks FlightGear does don't need to run on every single frame. Typically these tasks car be run every few seconds, minutes or hours as required. So to improve performance, the event manager allows for such tasks to be scheduled to run periodically and spread the processing load out over a longer time. The scheduler consists of two parts. The first part is simply a list of registered events along with any management information associated with that event. The second part is a run queue. When events are triggered, they are placed in the run queue. The system executes only one pending event per iteration in order to balance the load. More recently support has been added for pthreads, allowing such processes as tile loading to run in a separate thread, almost eliminating the pauses as the scenery loads.

Property System

The property database and manager provides an easy method of accessing simulator variables. It uses an object oriented hierarchical system, whereby the current aircraft latitude is located in /sim/position/latitude. The property manager provides setter and getter methods which are used to access these properties. Properties can also be modified using the network interface, which is ideal for Flight Instructors and also allows other programs to interact with the simulator in realtime. The property system makes GUI construction easier as well as allowing all critical simulator variables to be saved to a file at any time.

Conclusion

The FlightGear Flight Simulator provides a viable alternative to commercial flight simulators for many training and research purposes. The value of an open architecture and development style has already been shown by the many research projects that are using FlightGear. For instance, FlightGear is currently being used for icing research, autonomous aerial vehicle simulation, human factors research and as the visual system for a flight simulation facility for piloted evaluation of aircraft carrier operations. It has also been used by home cockpit builders as a high quality visual system. Without the intrinsic flexibility of the design none of this would be possible.

The project has created many challenges that no other open source project has had to face. These challenges are greatly enjoyed by the developers. In the future we hope to improve the accuracy of flight modelling so that it can realistically handle all edges of the flight envelope, add support for immersive technology, as well as adding more data to the scenery. FlightGear already handles many gigabytes of scenery data, and this is set to increase as hardware gets faster.

The FlightGear project is always looking for new developers, but at this time the project particularly needs the assistance of artists. Most coders aren't great artists, so some of our textures and models don't look the best in the industry. Recently we have received some artistic assistance, but as usual the more the better. Programmers are also needed for several sections of the simulator that still require finishing. The project is currently moving toward the 0.79 release, and we hope by version 1.0 to have a simulator that could compete with any of the currently available commercial sims.

Credits and References

Many thanks to the main developers:

Curtis Olson - Project Cordinator
David Megginson - Wrote property manager and works on flight dynamics among other things
John Check - Base package maintainer and panels
Norman Vine - Many improvements and bug fixes
Jon Berndt - JSBSim coordinator
Erik Ross - YASim coordinator
Tony Peden - Flight dynamics contributor
Oliver Delise - Network code
Christian Mayer - Weather subsystem
Alex Perry - VSI, DG, Altitude among other things

Plus a cast of thousands including:

Raul Alonzo
Eric Mitchell
Michele America
Michael Basler
David Findlay
Paul Bleisch
Jim Brennan
Bernhard H. Buckel
Bernie Bright
Gene Buckle
Ralph Carmichael
Didier Chauveau
Reto Koradi
Bob Kuehne
David Luff
Alan Murta
Phil Nelson
Alexei Novikov
Brian Paul
Dave Cornish
Dave Eberly
Jean-Francois Doue
Francine Evans
Oscar Everitt
Bruce Finney
Jean-loup Gailly
Mark Adler
Thomas Gellekum
Neetha Girish
Jeff Goeke-Smith
Michael I. Gold
Mike Hill
Charlie Hotchkiss
Bruce Jackson of NASA
Richard Kaszeta
Tom Knienieder
Mark Vallevand
Gary R. Van Sickle
Robin Peel
Friedemann Reinhard
Petter Reinholdtsen
William Riley
Paul Schlyter
Chris Schoeneman
Phil Schubert
Jonathan R Shewchuk
Gordan Sikic
Michael Smith
U.S.G.S
Carmelo Volpe
Darrell Walisser
Ed Williams
Jean-Claude Wippler
Robert Allan Zeheological
Survey
Durk Talsma
UIUC
Roland Voegtlinienieder

For more information on FlightGear, the following websites may be useful:

http://www.flightgear.org - The Official FlightGear Web Site
http://jsbsim.sf.net - JSBSim
http://www.isis.de/members/~odelise/progs/flightgear - Multiplayer FGFS
http://fgatd.sourceforge.net/ - FAA Certification
http://atlas.sourceforge.net/ - Atlas moving map for FGFS
http://rockfish.net/fg/ - John Check's FGFS pages