The FlightGear Manual

Michael Basler and Martin Spott

including contributions by
Stuart Buchanan, Jon Berndt,
Bernhard Buckel, Cameron Moore,
Curt Olson, Dave Perry,
Michael Selig, Darrell Walisser,
and others

The FlightGear Manual Version 0.9
November 19, 2006
For ***FlightGear*** version 0.9.10.

Contents

Preface

FlightGear is a free Flight Simulator developed cooperatively over the Internet by a group of flight simulation and programming enthusiasts. "The FlightGear Manual" is meant to give beginners a guide in getting ***FlightGear*** up and running, and themselves into the air. It is not intended to provide complete documentation of all the features and add-ons of ***FlightGear*** but, instead, aims to give a new user the best start to exploring what ***FlightGear*** has to offer.

This guide is split into three parts and is structured as follows.

Part I: Installation

Chapter 1, *Want to have a free flight? Take **FlightGear***, introduces ***FlightGear***, provides background on the philosophy behind it and describes the system requirements.

In Chapter 2, *Preflight: Installing **FlightGear***, you will find instructions for installing the binaries and additional scenery and aircraft.

Part II: Flying with *FlightGear*

The following Chapter 3, *Takeoff: How to start the program*, describes how to actually start the installed program. It includes an overview on the numerous command line options as well as configuration files.

Chapter 4, *In-flight: All about instruments, keystrokes and menus*, describes how to operate the program, i. e. how to actually fly with ***FlightGear***. This includes a (hopefully) complete list of pre-defined keyboard commands, an overview on the menu entries, detailed descriptions on the instrument panel and HUD (head up display), as well as hints on using the mouse functions.

Chapter 5, *Features*: Illustration of special features that ***FlightGear*** offers to the advanced user.

Part III: Tutorials

Chapter 6, *Tutorials*, provides information on the many tutorials available for new pilots.

Chapter 8, *A Cross Country Flight Tutorial*, describes a simple cross-country flight in the San Fransisco area that can be run with the default installation.

Appendices

In Appendix A, *Missed approach: If anything refuses to work*, we try to help you work through some common problems faced when using ***FlightGear***.

Appendix C, *OpenGL graphics drivers*, describes some special problems you may encounter in case your system lacks support for the OpenGL graphics API OpenGL which ***FlightGear*** is based on.

Appendix B, *Building the plane: Compiling the program*, explains how to build (compile and link) the simulator. Depending on your platform this may or may not be required.

In the final Appendix D, *Landing: Some further thoughts before leaving the plane*, we would like to give credit to those who deserve it, sketch an overview on the development of ***FlightGear***and point out what remains to be done.

Accordingly, we suggest reading the Chapters as follows:

Installation	
Users of binary distributions (notably under Windows):	2
Installation under Linux/UNIX:	B, 2
Installation under Macintosh:	2
Operation	
Program start (all users):	3
Keycodes, Panel, Mouse. . . (all users):	4
Troubleshooting	
General issues:	A
Graphics problems:	C
Optionally	1, D

While this introductory guide is meant to be self contained, we strongly suggest having a look into further documentation, especially in case of trouble:

- For additional hints on troubleshooting and more, **please read the FAQ**

  http://www.flightgear.org/Docs/FlightGear-FAQ.html,

  The FAQ contains a host of valuable information, especially on rapidly changing flaws and additional reading, thus we strongly suggest consulting it in conjunction with our guide.

- A handy **leaflet** on operation for printout is available at

  http://www.flightgear.org/Docs/InstallGuide/FGShortRef.html,

- Additional user documentation on special aspects is available within the base package under the directory `/FlightGear/Docs`.

Finally:

We know most people hate reading manuals. If you are sure the graphics driver for your card supports OpenGL (check documentation; for instance all NVIDIA Windows and Linux drivers for TNT/TNT2/Geforce/Geforce2/Geforce3 do) and if you are using one of the following operating systems:

- Windows 95/98/ME/NT/2000/XP,
- Macintosh Mac OSX
- Linux
- SGI Irix

you can possibly skip at least Part I of this manual and exploit the pre-compiled binaries. These as well as instructions on how to set them up, can be found at

http://www.flightgear.org/Downloads/.

In case you are running ***FlightGear*** on Linux, you may also be able to get binaries bundled with your distribution. Several vendors already include ***FlightGear*** binaries into their distributions.

Just download them, install them according to the description and run them via the installed FlightGear icon (on Windows), or textttfgfs having set the environmental variables described in Chapter 3 (on Linux).

There is no guarantee for this approach to work, though. If it doesn't, don't give up! Have a closer look through this guide notably Section 2 and be sure to check out the FAQ.

Part I

Installation

Chapter 1

Want to have a free flight? Take *FlightGear*!

1.1 Yet Another Flight Simulator?

Did you ever want to fly a plane yourself, but lacked the money or ability to do so? Are you a real pilot looking to improve your skills without having to take off? Do you want to try some dangerous maneuvers without risking your life? Or do you just want to have fun with a more serious game without any violence? If any of these questions apply to you, PC flight simulators are just for you.

You may already have some experience using Microsoft's © Flight Simulator or any other of the commercially available PC flight simulators. As the price tag of those is usually within the $50 range, buying one of them should not be a serious problem given that running any serious PC flight simulator requires PC hardware within the $1500 range, despite dropping prices.

With so many commercially available flight simulators, why would we spend thousands of hours of programming and design work to build a free flight simulator? Well, there are many reasons, but here are the major ones:

- All of the commercial simulators have a serious drawback: they are made by a small group of developers defining their properties according to what is important to them and providing limited interfaces to end users. Anyone who has ever tried to contact a commercial developer would agree that getting your voice heard in that environment is a major challenge. In contrast, ***FlightGear*** is designed by the people and for the people with everything out in the open.

- Commercial simulators are usually a compromise of features and usability. Most commercial developers want to be able to serve a broad segment of the population, including serious pilots, beginners, and even casual gamers. In reality the result is always a compromise due to deadlines and funding. As ***FlightGear*** is free and open, there is no need for such a compromise.

We have no publisher breathing down our necks, and we're all volunteers that make our own deadlines. We are also at liberty to support markets that no commercial developer would consider viable, like the scientific research community.

- Due to their closed-source nature, commercial simulators keep developers with excellent ideas and skills from contributing to the products. With ***FlightGear***, developers of all skill levels and ideas have the potential to make a huge impact on the project. Contributing to a project as large and complex as ***FlightGear*** is very rewarding and provides the developers with a great deal of pride in knowing that we are shaping the future of a great simulator.

- Beyond everything else, it's just plain fun! I suppose you could compare us to real pilots that build kit-planes or scratch-builts. Sure, we can go out a buy a pre-built aircraft, but there's just something special about building one yourself.

The points mentioned above form the basis of why we created ***FlightGear***. With those motivations in mind, we have set out to create a high-quality flight simulator that aims to be a civilian, multi-platform, open, user-supported, and user-extensible platform. Let us take a closer look at each of these characteristics:

- **Civilian:** The project is primarily aimed at civilian flight simulation. It should be appropriate for simulating general aviation as well as civilian aircraft. Our long-term goal is to have ***FlightGear*** FAA-approved as a flight training device. To the disappointment of some users, it is currently not a combat simulator; however, these features are not explicitly excluded. We just have not had a developer that was seriously interested in systems necessary for combat simulation.

- **Multi-platform:** The developers are attempting to keep the code as platform-independent as possible. This is based on their observation that people interested in flight simulations run quite a variety of computer hardware and operating systems. The present code supports the following Operating Systems:

  - Linux (any distribution and platform),
  - Windows NT/2000/XP (Intel/AMD platform),
  - Windows 95/98/ME,
  - BSD UNIX,
  - SGI IRIX,
  - Sun-OS,
  - Macintosh.

At present, there is no known flight simulator – commercial or free – supporting such a broad range of platforms.

- **Open:** The project is not restricted to a static or elite cadre of developers. Anyone who feels they are able to contribute is most welcome. The code (including documentation) is copyrighted under the terms of the GNU General Public License (GPL).

  The GPL is often misunderstood. In simple terms it states that you can copy and freely distribute the program(s) so licensed. You can modify them if you like and even charge as much money as want to for the distribution of the modified or original program. However, you must freely provide the entire source code to anyone who wants it, and it must retain the original copyrights. In short:

  "You can do anything with the software except make it non-free".

  The full text of the GPL can be obtained from the ***FlightGear*** source code or from

  http://www.gnu.org/copyleft/gpl.html.

- **User-supported and user-extensible:** Unlike most commercial simulators, ***FlightGear***'s scenery and aircraft formats, internal variables, APIs, and everything else are user accessible and documented from the beginning. Even without any explicit development documentation (which naturally has to be written at some point), one can always go to the source code to see how something works. It is the goal of the developers to build a basic engine to which scenery designers, panel engineers, maybe adventure or ATC routine writers, sound artists, and others can build upon. It is our hope that the project, including the developers and end users, will benefit from the creativity and ideas of the hundreds of talented "simmers" around the world.

Without doubt, the success of the Linux project, initiated by Linus Torvalds, inspired several of the developers. Not only has Linux shown that distributed development of highly sophisticated software projects over the Internet is possible, it has also proven that such an effort can surpass the level of quality of competing commercial products.

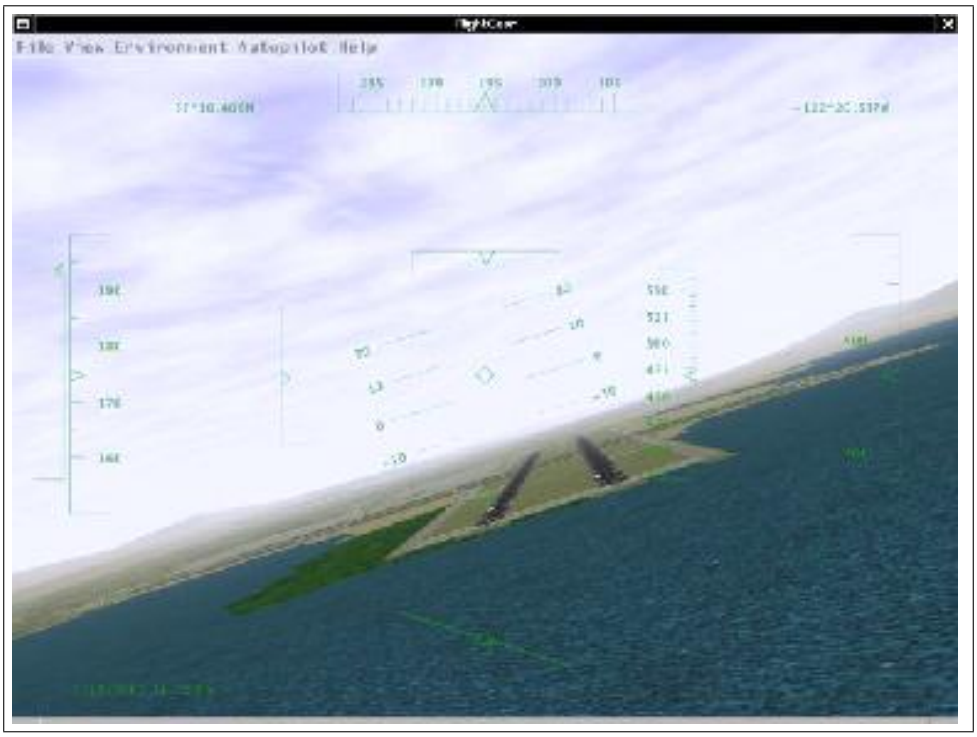


Fig. 1: ***FlightGear*** *under UNIX: Bad approach to San Francisco International - by one of the authors of this manual...*

1.2 System Requirements

In comparison to other recent flight simulators, the system requirements for ***FlightGear*** are not extravagant. A decent PIII/800, or something in that range, should be sufficient given you have a proper 3-D graphics card. Additionally, any modern UNIX-type workstation with a 3-D graphics card will handle ***FlightGear*** as well.

One important prerequisite for running ***FlightGear*** is a graphics card whose driver supports OpenGL. If you don't know what OpenGL is, the overview given at the OpenGL website

http://www.opengl.org

says it best: "Since its introduction in 1992, OpenGL has become the industry's most widely used and supported 2-D and 3-D graphics application programming interface (API)...".

FlightGear does not run (and will never run) on a graphics board which only supports Direct3D. Contrary to OpenGL, Direct3D is a proprietary interface, being restricted to the Windows operating system.

You may be able to run ***FlightGear*** on a computer that features a 3-D video card not supporting hardware accelerated OpenGL – and even on systems without 3-D graphics hardware at all. However, the absence of hardware accelerated

OpenGL support can bring even the fastest machine to its knees. The typical signal for missing hardware acceleration are frame rates below 1 frame per second.

Any modern 3-D graphics featuring OpenGL support will do. For Windows video card drivers that support OpenGL, visit the home page of your video card manufacturer. You should note that sometimes OpenGL drivers are provided by the manufacturers of the graphics chip instead of by the makers of the board. If you are going to buy a graphics card for running ***FlightGear***, one based on a NVIDIA chip (TNT X/Geforce X) might be a good choice.

To install the executable and basic scenery, you will need around 50 MB of free disk space. In case you want/have to compile the program yourself you will need about an additional 500 MB for the source code and for temporary files created during compilation. This does not include the development environment, which will vary in size depending on the operating system and environment being used. Windows users can expect to need approximately 300 MB of additional disk space for the development environment. Linux and other UNIX machines should have most of the development tools already installed, so there is likely to be little additional space needed on those platforms.

For the sound effects, any capable sound card should suffice. Due to its flexible design, ***FlightGear*** supports a wide range of joysticks and yokes as well as rudder pedals under Linux and Windows. ***FlightGear*** can also provide interfaces to full-motion flight chairs.

FlightGear is being developed primarily under Linux, a free UNIX clone (together with lots of GNU utilities) developed cooperatively over the Internet in much the same spirit as ***FlightGear*** itself. ***FlightGear*** also runs and is partly developed under several flavors of Windows. Building ***FlightGear*** is also possible on a Macintosh OSX and several different UNIX/X11 workstations. Given you have a proper compiler installed, ***FlightGear*** can be built under all of these platforms. The primary compiler for all platforms is the free GNU C++ compiler (the Cygnus Cygwin compiler under Win32).

If you want to run ***FlightGear*** under Mac OSX we suggest a Power PC G3 300 MHz or better. As a graphics card we would suggest an ATI Rage 128 based card as a minimum. Joysticks are supported under Mac OS 9.x only; there is no joystick support under Max OSX at this time.

1.3 Choosing A Version

Previously the ***FlightGear*** source code existed in two branches, a stable branch and a developmental branch. Even version numbers like 0.6, 0.8, and (someday hopefully) 1.0 refer to stable releases, while odd numbers like 0.7, 0.9, and so on refer to developmental releases. This policy has been obsoleted by practical reasons - mostly because stable releases got out-ouf-date and behind in features very fast and the so called development releases had been proven to be of comparable stability.

You are invited to fetch the "latest official release" which the pre-compiled binaries are based on. It is available from

http://www.flightgear.org/Downloads/

If you really want to get the very latest and greatest (and, at times, buggiest) code, you can use a tool called anonymous cvs to get the recent code. A detailed description of how to set this up for ***FlightGear*** can be found at

http://www.flightgear.org/cvsResources/.

Given that the recent developmental versions on the other hands may contain bugs (...undocumented features), we recommend using the "latest official (unstable) release" for the average user. This is the latest version named at
http://www.flightgear.org/News/.

1.4 Flight Dynamics Models

Historically, ***FlightGear*** has been based on a flight model it inherited (together with the Navion airplane) from LaRCsim. As this had several limitations (most important, many characteristics were hard wired in contrast to using configuration files), there were several attempts to develop or include alternative flightmodels. As a result, ***FlightGear*** supports several different flight models, to be chosen from at runtime.

The most important one is the JSB flight model developed by Jon Berndt. Actually, the JSB flight model is part of a stand-alone project called ***JSBSim***, having its home at

http://jsbsim.sourceforge.net/.

Concerning airplanes, the JSB flight model at present provides support for a Cessna 172, a Cessna 182, a Cessna 310, and for an experimental plane called X15. Jon and his group are gearing towards a very accurate flight model, and the JSB model has become ***FlightGear***'s default flight model.

As an interesting alternative, Christian Mayer developed a flight model of a hot air balloon. Moreover, Curt Olson integrated a special "UFO" slew mode, which helps you to quickly fly from point A to point B.

Recently, Andrew Ross contributed another flight model called ***YASim*** for *Yet Another Simulator*. At present, it sports another Cessna 172, a Turbo 310, a fairly good DC-3 model, along with a Boeing 747, Harrier, and A4. ***YASim*** takes a fundamentally different approach since it's based on geometry information rather than aerodynamic coefficients. Where ***JSBSim*** will be exact for every situation that is known and flight tested, but may have odd and/or unrealistic behavior outside normal flight, ***YASim*** will be sensible and consistent in almost every flight situation, but is likely to differ in performance numbers.

As a further alternative, there is the UIUC flight model, developed by a team at the University of Illinois at Urbana-Champaign. This work was initially geared

toward modeling aircraft in icing conditions together with a smart icing system to better enable pilots to fly safely in an icing encounter. While this research continues, the project has expanded to include modeling "nonlinear" aerodynamics, which result in more realism in extreme attitudes, such as stall and high angle of attack flight. Two good examples that illustrate this capability are the Airwave Xtreme 150 hang glider and the 1903 Wright Flyer. For the hang glider, throttle can be use to fly to gliding altitude or Ctrl-U can be used to jump up in 1000-ft increments. Try your hand at the unstable Wright Flyer and don't stall the canard! Considerable up elevator trim will be required for level flight. In general, the aerodynamics are probably very close to the original Wright Flyer as they are partly based on experimental data taken on a replica tested recently at the NASA Ames Research Center. Also included are two more models, a Beech 99 and Marchetti S-211 jet trainer, which are older generation UIUC/FGFS models and based on simpler "linear" aerodynamics. More details of the UIUC flight model and a list of aircraft soon to be upgraded can be found on their website at

http://amber.aae.uiuc.edu/~m-selig/apasim.html

Note that the 3D models of the UIUC airplanes can be downloaded from a site maintained by Wolfram Kuss

http://home.t-online.de/home/Wolfram.Kuss/

It is even possible to drive FlightGear's scene display using an external FDM running on a different computer or via named pipe on the local machine – although this might not be a setup recommended to people just getting in touch with ***FlightGear***.

1.5 About This Guide

There is little, if any, material in this Guide that is presented here exclusively. You could even say with Montaigne that we "merely gathered here a big bunch of other men's flowers, having furnished nothing of my own but the strip to hold them together". Most (but fortunately not all) of the information herein can also be obtained from the ***FlightGear*** web site located at

http://www.flightgear.org/

Please, keep in mind that there are several mirrors of the ***FlightGear*** web sites, all of which are linked to from the ***FlightGear*** homepage listed above. You may prefer to download ***FlightGear*** from a mirror closer to you than from the main site.

*The **FlightGear** Manual* is intended to be a first step towards a complete ***FlightGear*** documentation. The target audience is the end-user who is not interested in the internal workings of OpenGL or in building his or her own scenery. It is our hope, that someday there will be an accompanying ***FlightGear*** *Programmer's Guide* (which could be based on some of the documentation found at

http://www.flightgear.org/Docs;

a ***FlightGear*** *Scenery Design Guide*, describing the Scenery tools now packaged as ***TerraGear***; and a ***FlightGear*** *Flight School* package.

As a supplement, we recommend reading the ***FlightGear*** FAQ to be found at
http://www.flightgear.org/Docs/FlightGear-FAQ.html
which has a lot of supplementary information that may not be included in this manual.

We kindly ask you to help us refine this document by submitting corrections, improvements, and suggestions. All users are invited to contribute descriptions of alternative setups (graphics cards, operating systems etc.). We will be more than happy to include those into future versions of *The FlightGear Manual* (of course not without giving credit to the authors).

While we intend to continuously update this document, we may not be able to produce a new version for every single release of ***FlightGear***. To do so would require more manpower that we have now, so please feel free to jump in and help out. We hope to produce documentation that measures up to the quality of ***FlightGear*** itself.

Chapter 2

Preflight: Installing *FlightGear*

To run ***FlightGear*** you need to install the binaries. Once you've done this you may install additional scenery and aircraft if you wish.

Pre-compiled binaries for the latest release are available for

- Windows - any flavor,
- Macintosh OSX,
- Linux,
- SGI Irix.

To download them go to

http://www.flightgear.org/Downloads/binary.shtml

and follow the instructions provided on the page.

If you are running on another OS, or wish to compile for yourself, see Appendix B, *Building the plane: Compiling the program*.

2.1 Installing scenery

The ***FlightGear*** base package contains scenery for a small area around San Fransisco, but the entire world is available at a high level of detail, so you will almost certainly wish to install extra scenery at some point.

The scenery is based on SRTM elevation data (accurate to 30m in the USA, and 90m elsewhere and) VMap0 land use data. Additionally, various people have created buildings, bridges and other features to enrich the environment.

You can download scenery in 10 degree by 10 degree chunks from a clickable map at

http://www.flightgear.org/Downloads/scenery.html

Curt Olson also provides the USA or the entire world along with the latest FlightGear release on DVD from here:

http://cdrom.flightgear.org/

If you are interested in generating your own scenerey, have a look at TerraGear - the tools that generate the scenery for ***FlightGear***:

http://www.terragear.org/

Finally, an alternative data set was produced by William Riley and is available from here:

http://www.randdtechnologies.com/fgfs/newScenery/world-scenery.html.

Whatever scenery you shoose to download, it should be kept in a separate directory from the scenery delivered with the binaries.

To do this, create a `WorldScenery` directory in the ***FlightGear*** data directory, usually

`c:\Program Files\FlightGear\data`

on Windows or

`/usr/local/share/FlightGear/data`

on *nix.

Underneath this directory create `Terrain` and `Objects` subdirectories. These are used for terrain information and buildings/bridges/structures respectively.

Unpack the downloaded scenery into the `WorldScenery/Terrain`. Do not de-compress the numbered scenery files like 958402.gz! This will be done by ***FlightGear*** on the fly.

As an example, consider installation of the scenery package w120n30 containing the Grand Canyon Scenery into an installation located at

`/usr/local/share/FlightGear`.

Once your installation is complete, you'll have the following directories

```
/usr/local/FlightGear/data/WorldScenery/Objects/
/usr/local/FlightGear/data/WorldScenery/Terrain/w120n30/w112n30
/usr/local/FlightGear/data/WorldScenery/Terrain/w120n30/w112n31
...
/usr/local/FlightGear/data/WorldScenery/Terrain/w120n30/w120n39
```

As well as the scenery itself, objects such as bridges, skyscrapers, radio masts can be downloaded from http://fgfsdb.stockill.org. See the website for more information. You can exploit `FG_SCENERY` environmental variable or the `--fg-scenery=`*path* command line option if you want to install different scenery sets in parallel or want to have scenery sitting in another place. These are more fully described in Chapter 3.

2.2 Installing aircraft

The base ***FlightGear*** package contains only a small subset of the aircraft that are available for ***FlightGear***. Developers have created a wide range of aircraft, from WWII fighters like the Spitfire, to passenger planes like the Boeing 747.

You can download aircraft from

http://www.flightgear.org/Downloads/aircraft/index.shtml

Simply download the file and unpack it into the `FlightGear/data/Aircraft` subdirectory of your installation. Next time you run ***FlightGear***, the new aircraft will be available.

2.3 Installing documentation

Most of the packages named above include the complete ***FlightGear*** documentation including a ṗdf version of *The FlightGear Manual* intended for pretty printing using Adobe's Acrobat Reader being available from

http://www.adobe.com/acrobat

Moreover, if properly installed, the ḣtml version can be accessed via ***FlightGear***'s `help` menu entry.

Besides, the source code contains a directory `docs-mini` containing numerous ideas on and solutions to special problems. This is also a good place for further reading.

Part II

Flying with *FlightGear*

Chapter 3

Takeoff: How to start the program

3.1 Launching the simulator under Unix/Linux

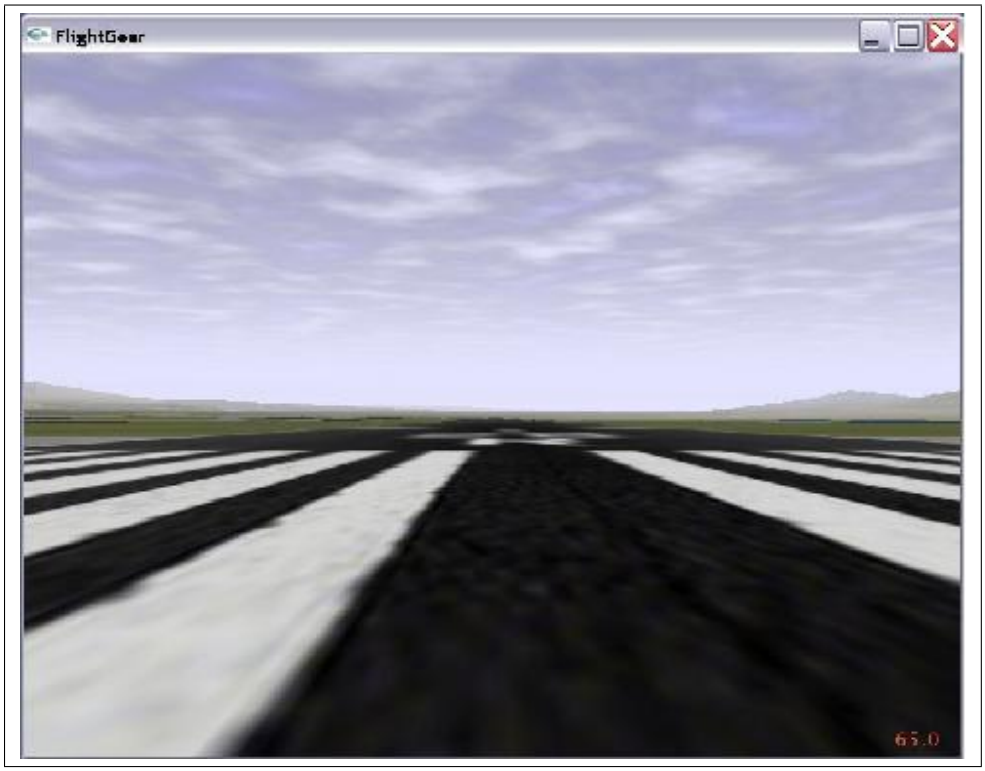


Fig. 3: *Ready for takeoff. Waiting at the default startup position at San Francisco Itl., KSFO.*

Before you can run ***FlightGear***, you need to have a couple of environmental variables set.

- You must add `/usr/local/share/FlightGear/lib` to your `LD_LIBRARY_PATH`
- `FG_HOME` must be set to the root of your ***FlightGear*** installation. e.g. `/usr/local/share/FlightGear`.
- `FG_ROOT` must be set to the date directory of your ***FlightGear*** installation. e.g. `/usr/local/share/FlightGear/data`.
- `FG_SCENERY` should be a list of scenery directories, separated by ":". This works like `PATH` when searching for scenery.

  e.g. `$FG_ROOT/Scenery:$FG_ROOT/WorldScenery`.

To add these in the Bourne shell (and compatibles):

```
LD_LIBRARY_PATH=/usr/local/share/FlightGear/lib:$LD_LIBRARY_PATH
export LD_LIBRARY_PATH
FG_HOME=/usr/local/share/FlightGear
export FG_HOME
FG_ROOT=/usr/local/share/FlightGear/data
export FG_ROOT
FG_SCENERY=$FG_ROOT/Scenery:$FG_ROOT/WorldScenery
export FG_SCENERY
```

or in C shell (and compatibles):

```
setenv LD_LIBRARY_PATH=\
  /usr/local/share/FlightGear/lib:$LD_LIBRARY_PATH
setenv FG_HOME=/usr/local/share/FlightGear
setenv FG_ROOT=/usr/local/share/FlightGear/data
setenv FG_SCENERY=\
  $FG_HOME/Scenery:$FG_ROOT/Scenery:$FG_ROOT/WorldScenery
```

Once you have these environmental variables set up, simply start ***FlightGear*** by running `fgfs --option1 --option2...`. Command-line options are described in Chapter 3.4.

3.2 Launching the simulator under Windows

The pre-built windows binaries come complete with a graphical wizard to start ***FlightGear***. Simply double-click on the `FlightGear Launcher` Start Menu item, or the icon on the Desktop. The launcher allows you to select

- your aircraft
- the start airport and runway
- time of day

- current weather
- ... and whole lot of other environmental settings

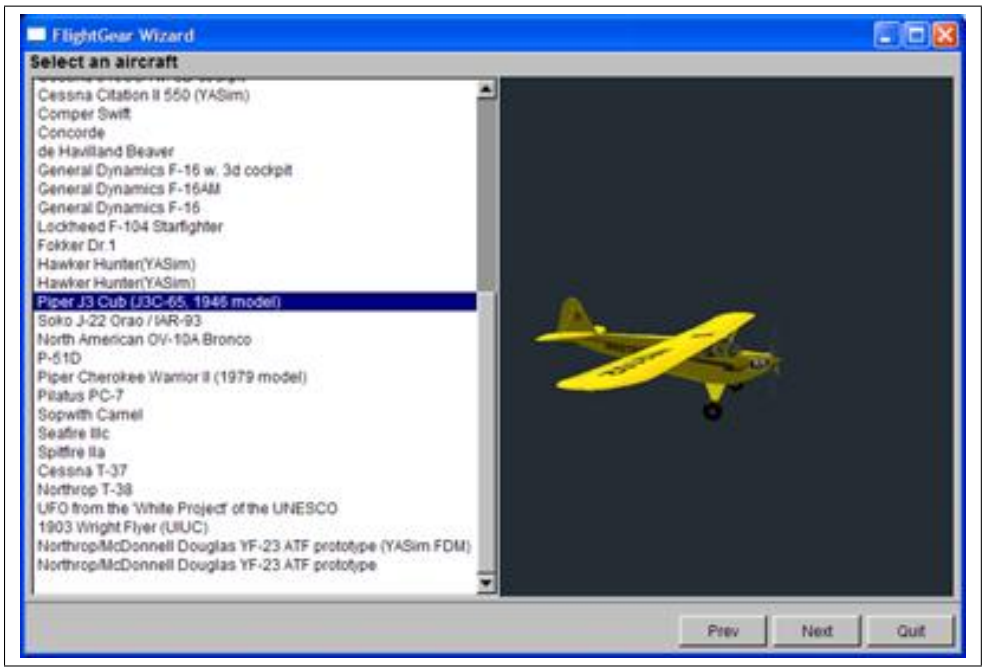


Fig. 4: *The FlightGear Launcher*

The first time your run it, you will be asked to set your `FG_ROOT` variable (normally `c:\Program Files\FlightGear\data`) and `FG_SCENERY`. This should be a list of the directories where you have installed scenery, typically

`c:\Program Files\FlightGear\data\scenery`

and

`c:\Program Files\FlightGear\data\WorldScenery`.

Alternatively, you can run FlightGear from the command line. Open a command shell, change to the directory where your binary resides (typically something like `c:\Program Files\FlightGear\Win32\bin`), set the environment variables by typing

```
SET FG_HOME="c:\Program Files\FlightGear"
SET FG_ROOT="c:\Program Files\FlightGear\data"
SET FG_SCENERY="c:\Program Files\FlightGear\data\Scenery";\
 "c:\Program Files\FlightGear\data\WorldScenery"
```

and invoke ***FlightGear*** (within the same Command shell, as environment settings are only valid locally within the same shell) via

```
fgfs --option1 --option2...
```

Of course, you can create a batch file with a Windows text editor (like notepad) using the lines above.

For getting maximum performance it is recommended to minimize (iconize) the text output window while running ***FlightGear***.

3.3 Launching the simulator under Mac OS X

Say, you downloaded the base package and binary to your home directory. Then you can open `Terminal.app` and execute the following sequence:

```
setenv FG_ROOT ~/fgfs-base-X.X.X
./fgfs-X.X.X.-date --option1 --option 2
```

or

```
./fgfs-X.X.X-version-date --fg-root=\~/fgfs-base-X.X.X --option1
```

3.4 Command line parameters

Following is a complete list and short description of the numerous command line options available for ***FlightGear***.Most of these options are exposed through the FlightGear launcher delivered with the Windows binaries.

If you have options you re-use continually, you can include them in a preferences file. As it depends on your preferences, it is not delivered with ***FlightGear***, but can be created with any text editor (notepad, emacs, vi, if you like).

- On Unix systems, create a `.fgfsrc` file in your home directory.
- On Windows, create a `system.fgfsrc`, in the `FG_ROOT` directory (e.g. `c:\Program Files\FlightGear\data`).

3.4.1 General Options

- `--help`: Shows the most relevant command line options only.
- `--help -verbose`: Shows all command line options.
- `--fg-root=`*path*: Tells ***FlightGear*** where to look for its root data files if you didn't compile it with the default settings.
- `--fg-scenery=`*path*: Allows specification of a path to the base scenery path , in case scenery is not at the default position under `$FG_ROOT/Scenery`; this might be especially useful in case you have scenery on a CD-ROM.

- `--disable-game-mode`: Disables full screen display.
- `--enable-game-mode`: Enables full screen display.
- `--disable-splash-screen`: Turns off the rotating 3DFX logo when the accelerator board gets initialized (3DFX only).
- `--enable-splash-screen`: If you like advertising, set this!
- `--disable-intro-music`: No audio sample is being played when ***FlightGear*** starts up. Suggested in case of trouble with playing the intro.
- `--enable-intro-music`: If your machine is powerful enough, enjoy this setting.
- `--disable-mouse-pointer`: Disables extra mouse pointer.
- `--enable-mouse-pointer`: Enables extra mouse pointer. Useful in full screen mode for old Voodoo based cards.
- `--enable-random-objects`: Include random scenery objects (buildings/trees). This is the default.
- `--disable-random-objects`: Exclude random scenery objects (buildings/trees).
- `--disable-freeze`: This will put you into ***FlightGear*** with the engine running, ready for Take-Off.
- `--enable-freeze`: Starts ***FlightGear*** in frozen state.
- `--disable-fuel-freeze`: Fuel is consumed normally.
- `--enable-fuel-freeze`: Fuel tank quantity is forced to remain constant.
- `--disable-clock-freeze`: Time of day advances normally.
- `--enable-clock-freeze`: Do not advance time of day.
- `--control-mode`: Specify your control device (joystick, keyboard, mouse) Defaults to joystick (yoke).
- `--disable-auto-coordination`: Switches auto coordination between aileron/rudder off (default).
- `--enable-auto-coordination`: Switches auto coordination between aileron/rudder on (recommended without pedals).

- `--browser-app=/path/to/app`: specify location of your web browser. Example: `--browser-app= "C:\Program Files\Internet Explorer\iexplore.exe"` (Note the " " because of the spaces!).
- `--prop:name=value:` set property `name` to `value` Example: `--prop:/engines/engine0/running=true` for starting with running engines. Another example:
  `--aircraft=c172`
  `--prop:/consumables/fuels/tank[0]/level-gal=10`
  `--prop:/consumables/fuels/tank[1]/level-gal=10`
  filles the Cessna for a short flight.
- `--config=path:` Load additional properties from the given path. Example: `fgfs --config=./Aircraft/X15-set.xml`
- `--units-feet`: Use feet for distances.
- `--units-meters`: Use meters for distances.

3.4.2 Features

- `--disable-hud`: Switches off the HUD (**H**ead **U**p **D**isplay).
- `--enable-hud`: Turns the HUD on.
- `--enable-anti-aliased-hud`: Turns on anti-aliaseded HUD lines for better quality, if hardware supports this.
- `--disable-anti-aliased-hud`: Turns off anti-aliaseded HUD lines.
- `--enable-panel`: Turns the instrument panel on (default).
- `--disable-panel`: Turns the instrument panel off.
- `--disable-sound`: Self explaining.
- `--enable-sound`: See above.

3.4.3 Aircraft

- `--aircraft=`*name of aircraft definition file* Example: `--aircraft=c310`. For possible choices check the directory `/FlightGear/Aircraft`. Do not include the extension `"-set.xml"` into the aircraft name but use the remaining beginning of the respective file names for choosing an aircraft. This way flight model, panel etcare all loaded in a consistent way. For a full list, see Sec. 3.6 below.
- `--show-aircraft`: Print a sorted list of the currently available aircraft types.

3.4.4 Flight model

- `--fdm=`*abcd* Select the core flight model. Options are `jsb, larcsim, yasim, magic, balloon, external, pipe, ada, null`. Default value is `jsb` (***JSBSim***). larcsim is the flight model which ***FlightGear*** inherited from the LaRCSim simulatorẏasim is Any Ross' Yet Another Flight Dynamics Simulator. Magic is a slew mode (which drives the UFO aircraft). Balloon is a hot air balloon. External refers to remote control of the simulator via TCP socket, pipe is for local control via named pipe. Null selects no flight dynamics model at all. The UIUC flight model is not chosen this way but via the next option! For further information on flight models cf. Section 1.4 and below.

- `--aero=`*abcd* Specifies the aircraft model to load. Default is a Cessna c172. Alternatives available depend on the flight model chosen.

- `--model-hz=`*n* Run the Flight Dynamics Model with this rate (iterations per second).

- `--speed=`*n* Run the Flight Dynamics Model this much faster than real time.

- `--notrim` Do NOT attempt to trim the model when initializing JSBSim.

- `--on-ground`: Start up at ground level (default).

- `--in-air`: Start up in the air. Naturally, you have to specify an initial altitude as below for this to make sense. This is a must for the X15.

- `--wind=`*DIR@SPEED*: Specify wind coming from the direction DIR (in degrees) at speed SPEED (knots). Values may be specified as a range by using a clon separator; e.g. 180:220@10:15

- `--random-wind`: Adds random wind to make flying more challenging

3.4.5 Initial Position and Orientation

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

- `--offset-distance=`*nm*: Here you can specify the distance to threshold in nm.

- `--offset-azimuth=`*deg*: Here you can specify the heading to threshold in degrees.

- `--lon=`*degrees*: This is the startup longitude in degrees (west = -).
- `--lat=`*degrees*: This is the startup latitude in degrees (south = -).
- `--altitude=`*feet*: This is useful if you want to start in free flight in connection with `--in-air`. Altitude specified in feet unless you choose `--units-meters`.
- `--heading=`*degrees*: Sets the initial heading (yaw angle) in degrees.
- `--roll=`*degrees*: Sets the startup roll angle (roll angle) in degrees.
- `--pitch=`*degrees*: Sets the startup pitch angle (pitch angle) in degrees.
- `--uBody=`*feet per second*: Speed along the body X axis in feet per second, unless you choose `--units-meters`.
- `--vBody=`*feet per second*: Speed along the body Y axis in feet per second, unless you choose `--units-meters`.
- `--wBody=`*feet per second*: Speed along the body Z axis in feet per second, unless you choose `--units-meters`.
- `--vc=`*knots*: Allows specifying the initial airspeed in knots (only in connection with `--fdm=jsb`).
- `--mach=`*num*: Allows specifying the initial airspeed as Mach number (only in connection with `--fdm=jsb`).
- `--glideslope=`*degrees*: Allows specifying the flight path angle (can be positive).
- `--roc=`*fpm*: Allows specifying the initial climb rate (can be negative).

3.4.6 Rendering Options

- `--bpp=`*depth*: Specify the bits per pixel.
- `--fog-disable`: To cut down the rendering efforts, distant regions are vanishing in fog by default. If you disable fogging, you'll see farther but your frame rates will drop.
- `--fog-fastest`: The scenery will not look very nice but frame rate will increase.
- `--fog-nicest`: This option will give you a fairly realistic view of flying on a hazy day.
- `--enable-clouds`: Enable cloud layer (default).

- `--disable-clouds`: Disable cloud layer.
- `--fov=`*degrees*: Sets the field of view in degrees. Default is 55.0.
- `--disable-fullscreen`: Disable full screen mode (default).
- `--enable-fullscreen`: Enable full screen mode.
- `--shading-flat`: This is the fastest mode but the terrain will look ugly! This option might help if your video processor is really slow.
- `--shading-smooth`: This is the recommended (and default) setting - things will look really nice.
- `--disable-skyblend`: No fogging or haze, sky will be displayed using just one color. Fast but ugly!
- `--enable-skyblend`: Fogging/haze is enabled, sky and terrain look realistic. This is the default and recommended setting.
- `--disable-textures`: Terrain details will be disabled. Looks ugly, but might help if your video board is slow.
- `--enable-textures`: Default and recommended.
- `--enable-wireframe`: If you want to know how the world of ***FlightGear*** looks like internally, try this!
- `--disable-wireframe`: No wireframe. Default.
- `--geometry=`*WWWxHHH*: Defines the size of the window used, i.e. `WWWxHHH` can be `640x480`, `800x600`, or `1024x768`.
- `--view-offset=`*xxx*: Allows setting the default forward view direction as an offset from straight ahead. Possible values are `LEFT, RIGHT, CENTER`, or a specific number of degrees. Useful for multi-window display.
- `--visibility=`*meters*: You can specify the initial visibility in meters here.
- `--visibility-miles=`*miles*: You can specify the initial visibility in miles here.

3.4.7 HUD Options

- `--hud-tris`: HUD displays the number of triangles rendered.
- `--hud-culled`: HUD displays percentage of triangles culled.

3.4.8 Time Options

- `--time-offset=`*[+-]hh:mm:ss*: Offset local time by this amount.
- `--time-match-real`: Synchronize real-world and ***FlightGear*** time.
- `--time-match-local`: Synchronize local real-world and ***FlightGear*** time.
- `--start-date-sys=`*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.

3.4.9 Network Options

- `--httpd=`*port*: Enable http server on the specified port.
- `--telnet=`*port*: Enable telnet server on the specified port.
- `--jpg-httpd=`*port*: Enable screen shot http server on the specified port.
- `--enable-network-olk`: Enables Oliver Delises's multi-pilot mode.
- `--disable-network-olk`: Disables Oliver Delises's multi-pilot mode (default).
- `--net-hud`: HUD displays network info.
- `--net-id=`*name*: Specify your own callsign

3.4.10 Route/Waypoint Options

- `--wp=`*ID[@alt]*: Allows specifying a waypoint for the GC autopilot; it is possible to specify multiple waypoints (i.eả route) via multiple instances of this command.
- `--flight-plan=`*[file]*: This is more comfortable if you have several waypoints. You can specify a file to read them from.

Note: These options are rather geared to the advanced user who knows what he is doing.

3.4.11 IO Options

- `--garmin=`*params*: Open connection using the Garmin GPS protocol.
- `--joyclient=`*params*: Open connection to an Agwagon joystick.
- `--native-ctrls=`*params*: Open connection using the FG native Controls protocol.
- `--native-fdm=`*params*: Open connection using the FG Native FDM protocol.
- `--native=`*params*: Open connection using the FG Native protocol.
- `--nmea=`*params*: Open connection using the NMEA protocol.
- `--opengc=`*params*: Open connection using the OpenGC protocol.
- `--props=`*params*: Open connection using the interactive property manager.
- `--pve=`*params*: Open connection using the PVE protocol.
- `--ray=`*params*: Open connection using the RayWoodworth motion chair protocol.
- `--rul=`*params*: Open connection using the RUL protocol.
- `--atc610x`: Enable atc610x interface.

3.4.12 Debugging options

- `--trace-read=`*params*: Trace the reads for a property; multiple instances are allowed.
- `--trace-write=`*params*: Trace the writes for a property; multiple instances are allowed.

3.5 Joystick support

Could you imagine a pilot in his or her Cessna controlling the machine with a keyboard alone? For getting the proper feeling of flight you will need a joystick/yoke plus rudder pedals, right? However, the combination of numerous types of joysticks, flightsticks, yokes, pedals etcon the market with the several target operating systems, makes joystick support a nontrivial task in ***FlightGear***.

Beginning with version 0.8.0, ***FlightGear*** has a reworked integrated joystick support, which automatically detects any joystick, yoke, or pedals attached. Just try it! If this does work for you, lean back and be happy!

Unfortunately, given the several combinations of operating systems supported by ***FlightGear*** (possibly in foreign languages) and joysticks available, chances are your joystick does not work out of the box. Basically, there are two alternative approaches to get it going, with the first one being preferred.

3.5.1 Built-in joystick support

General remarks

In order for joystick auto-detection to work, a joystick bindings xml file must exist for each joystick. This file describes what axes and buttons are to be used to control which functions in ***FlightGear***. The associations between functions and axes or buttons are called "bindings". This bindings file can have any name as long as a corresponding entry exists in the joysticks description file

```
/FlightGear/joysticks.xml
```

which tells ***FlightGear*** where to look for all the bindings files. We will look at examples later.

FlightGear includes several such bindings files for several joystick manufacturers in folders named for each manufacturer. For example, if you have a CH Products joystick, look in the folder

```
/FlightGear/Input/Joysticks/CH
```

for a file that might work for your joystick. If such a file exists and your joystick is working with other applications, then it should work with ***FlightGear*** the first time you run it. If such a file does not exist, then we will discuss in a later section how to create such a file by cutting and pasting bindings from the examples that are included with ***FlightGear***.

Verifying your joystick is working

Does your computer see your joystick? One way to answer this question under Linux is to reboot your system and immediately enter on the command line

```
dmesg | grep Joystick
```

which pipes the boot message to grep which then prints every line in the boot message that contains the string "Joystick". When you do this with a Saitek joystick attached, you will see a line similar to this one:

```
input0:  USB HID v1.00 Joystick [SAITEK CYBORG 3D USB] on
usb2:3.0
```

This line tells us that a joystick has identified itself as SAITEK CYBORG 3D USB to the operating system. It does not tell us that the joystick driver sees your joystick. If you are working under Windows, the method above does not work, but you can still go on with the next paragraph.

Confirming that the driver recognizes your joystick

FlightGear ships with a utility called js_demo. It will report the number of joysticks attached to a system, their respective "names", and their capabilities. Under Linux, you can run js_demo from the folder `/FlightGear/bin` as follows:

```
$ cd /usr/local/FlightGear/bin
$ ./js_demo
```

Under Windows, open a command shell (Start|All Programs|Accessories), go to the ***FlightGear*** binary folder and start the program as follows (given ***FlightGear*** is installed under `c:\Flightgear`)

```
cd \FlightGear\bin
js_demo.exe
```

On our system, the first few lines of output are (stop the program with ^C if it is quickly scrolling past your window!) as follows:

```
Joystick test program.
Joystick 0:  "CH PRODUCTS CH FLIGHT SIM YOKE USB "
Joystick 1:  "CH PRODUCTS CH PRO PEDALS USB"
Joystick 2 not detected
Joystick 3 not detected
Joystick 4 not detected
Joystick 5 not detected
Joystick 6 not detected
Joystick 7 not detected
+-------------JS.0---------------+-------------JS.1---------------+
| Btns Ax:0 Ax:1 Ax:2 Ax:3 Ax:4 Ax:5 Ax:6 | Btns Ax:0 Ax:1 Ax:2 |
+------------------------------+------------------------------+
| 0000 +0.0 +0.0 +1.0 -1.0 -1.0 +0.0 +0.0 .  | 0000 -1.0 -1.0 -1.0 .  .  .  .  .  |
```

First note that js_demo reports which number is assigned to each joystick recognized by the driver. Also, note that the "name" each joystick reports is also included between quotes. We will need the names for each bindings file when we begin writing the binding xml files for each joystick.

Identifying the numbering of axes and buttons

Axis and button numbers can be identified using js_demo as follows. By observing the output of js_demo while working your joystick axes and buttons you can determine what axis and button numbers are assigned to each joystick axis and button. It should be noted that numbering generally starts with zero.

The buttons are handled internally as a binary number in which bit 0 (the least significant bit) represents button 0, bit 1 represents button 1, etc., but this number is displayed on the screen in hexadecimal notation, so:

0001 ⇒ button 0 pressed
0002 ⇒ button 1 pressed
0004 ⇒ button 2 pressed
0008 ⇒ button 3 pressed
0010 ⇒ button 4 pressed
0020 ⇒ button 5 pressed
0040 ⇒ button 6 pressed
... etcu̇p to ...
8000 ⇒ button 15 pressed
... and ...
0014 ⇒ buttons 2 and 4 pressed simultaneously
... etc.

For Linux users, there is another option for identifying the "name" and the numbers assigned to each axis and button. Most Linux distributions include a very handy program, "jstest". With a CH Product Yoke plugged into the system, the following output lines are displayed by jstest:

```
jstest /dev/js3
Joystick (CH PRODUCTS CH FLIGHT SIM YOKE USB ) has 7 axes and 12 buttons.  Driver version is 2.1.0
Testing...(interrupt to exit)
Axes:  0:  0 1:  0 2:  0 3:  0 4:  0 5:  0 6:  0 Buttons:  0:off 1:off 2:off 3:on 4:off 5:off 6:off 7:off
8:off 9:off 10:off 11:off
```

Note the "name" between parentheses. This is the name the system associates with your joystick.

When you move any control, the numbers change after the axis number corresponding to that moving control and when you depress any button, the "off" after the button number corresponding to the button pressed changes to "on". In this way, you can quickly write down the axes numbers and button numbers for each function without messing with binary.

Writing or editing joystick binding xml files

At this point, you have confirmed that the operating system and the joystick driver both recognize your joystick(s). You also know of several ways to identify the joystick "name" your joystick reports to the driver and operating system. You will need a written list of what control functions you wish to have assigned to which axis and button and the corresponding numbers.

Make the following table from what you learned from js_demo or jstest above (pencil and paper is fine). Here we assume there are 5 axes including 2 axes associated with the hat.

Axis	Button
elevator = 0	view cycle = 0
rudder = 1	all brakes = 1
aileron = 2	up trim = 2
throttle = 3	down trim = 3
leftright hat = 4	extend flaps = 4
foreaft hat = 5	retract flaps = 5
	decrease RPM = 6
	increase RPM = 7

We will assume that our hypothetical joystick supplies the "name" QUICK STICK 3D USB to the system and driver. With all the examples included with ***FlightGear***, the easiest way to get a so far unsupported joystick to be auto detected, is to edit an existing binding xml file. Look at the xml files in the sub-folders of `/FlightGear/Input/Joysticks/`. After evaluating several of the xml binding files supplied with ***FlightGear***, we decide to edit the file `/FlightGear/Input/Joysticks/Saitek/Cyborg-Gold-3d-USB.xml`. This file has all the axes functions above assigned to axes and all the button functions above assigned to buttons. This makes our editing almost trivial.

Before we begin to edit, we need to choose a name for our bindings xml file, create the folder for the QS joysticks, and copy the original xml file into this directory with this name.

```
$ cd /usr/local/FlightGear/Input/Joysticks
$ mkdir QS
$ cd QS
$ cp /usr/local/FlightGear/Input/Joysticks/Saitek/
Cyborg-Gold-3d-USB.xml QuickStick.xml
```

Here, we obviously have supposed a Linux/UNIX system with ***FlightGear*** being installed under `/usr/local/FlightGear`. For a similar procedure under Windows with ***FlightGear*** being installed under `c:FlightGear`, open a command shell and type

```
c:
cd /FlightGear/Input/Joysticks
mkdir QS
cd QS
copy /FlightGear/Input/Joysticks/Saitek/
Cyborg-Gold-3d-USB.xml QuickStick.xml
```

Next, open `QuickStick.xml` with your favorite editor. Before we forget to change the joystick name, search for the line containing <name>. You should find the line

```
<name>SAITEK CYBORG 3D USB</name>
```

and change it to

<name>QUICK STICK 3D USB</name>.

This line illustrates a key feature of xml statements. They begin with a <tag> and end with a </tag>.

You can now compare your table to the comment table at the top of your file copy. Note that the comments tell us that the Saitek elevator was assigned to axis 1. Search for the string

<axis n="1">

and change this to

<axis n="0">.

Next, note that the Saitek rudder was assigned to axis 2. Search for the string

<axis n="2">

and change this to

<axis n="1">.

Continue comparing your table with the comment table for the Saitek and changing the axis numbers and button numbers accordingly. Since QUICKSTICK USB and the Saitek have the same number of axes but different number of buttons, you must delete the buttons left over. Just remember to double check that you have a closing tag for each opening tag or you will get an error using the file.

Finally, be good to yourself (and others when you submit your new binding file to a ***FlightGear*** developers or users archive!), take the time to change the comment table in the edited file to match your changed axis and button assignments. The new comments should match the table you made from the js_demo output. Save your edits.

Several users have reported that the numbers of axes and buttons assigned to functions may be different with the same joystick under Windows and Linux. The above procedure should allow one to easily change a binding xml file created for a different operating system for use by their operating system.

Telling *FlightGear* about your new bindings xml file

Before ***FlightGear*** can use your new xml file, you need to edit the file `/FlightGear/joysticks.xml`,
adding a line that will include your new file if the “name” you entered between the name tags matches the name supplied to the driver by your joystick. Add the following line to `joysticks.xml`.

```
<js-named include="Input/Joysticks/QS/QuickStick.xml"/>
```

Some hints for Windows users

Basically, the procedures described above should work for Windows as well. If your joystick/yoke/pedals work out of the box or if you get it to work using the methods above, fine. Unfortunately there may be a few problems.

The first one concerns users of non-US Windows versions. As stated above, you can get the name of the joystick from the program js_demo. If you have a non-US version of Windows and the joystick .xml files named above do not contain that special name, just add it on top of the appropriate file in the style of

```
<name>Microsoft-PC-Joysticktreiber </name>
```

No new entry in the base `joysticks.xml` file is required.

Unfortunately, there is one more loophole with Windows joystick support. In case you have two USB devices attached (for instance a yoke plus pedals), there may be cases, where the same driver name is reported twice. In this case, you can get at least the yoke to work by assigning it number 0 (out of 0 and 1). For this purpose, rotate the yoke (aileron control) and observe the output of js_demo. If figures in the first group of colons (for device 0) change, assignment is correct. If figures in the second group of colons (for device 1) change, you have to make the yoke the preferred device first. For doing so, enter the Windows "Control panel", open "Game controllers" and select the "Advanced" button. Here you can select the yoke as the "Preferred" device. Afterward you can check that assignment by running js_demo again. The yoke should now control the first group of figures.

Unfortunately, we did not find a way to get the pedals to work, too, that way. Thus, in cases like this one (and others) you may want to try an alternative method of assigning joystick controls.

3.5.2 Joystick support via .fgfsrc entries

Fortunately, there is a tool available now, which takes most of the burden from the average user who, maybe, is not that experienced with XML, the language which these files are written in.

For configuring your joystick using this approach, open a command shell (command prompt under windows, to be found under Start|All programs|Accessories). Change to the directory `/FlightGear/bin` via e.g. (modify to your path)

```
cd c:\FlightGear\bin
```

and invoke the tool fgjs via

```
./fgjs
```

on a UNIX/Linux machine, or via

```
fgjs
```

on a Windows machine. The program will tell you which joysticks, if any, were detected. Now follow the commands given on screen, i.e. move the axis and press the buttons as required. Be careful, a minor touch already "counts" as a movement. Check the reports on screen. If you feel something went wrong, just re-start the program.

After you are done with all the axis and switches, the directory above will hold a file called `fgfsrc.js`. If the ***FlightGear*** base directory `FlightGear` does not already contain an options file `.fgfsrc` (under UNIX)/`system.fgfsrc` (under Windows) mentioned above, just copy

`fgfsrc.js` into `.fgfsrc` (UNIX)/`system.fgfsrc` (Windows)

and place it into the directory ***FlightGear*** base directory `FlightGear`. In case you already wrote an options file, just open it as well as `fgfsrc.js` with an editor and copy the entries from `fgfsrc.js` into `.fgfsrc`/`system.fgfsrc`. One hint: The output of `fgjs` is UNIX formatted. As a result, Windows Editor may not display it the proper way. I suggest getting an editor being able to handle UNIX files as well (and oldie but goldie in this respect is PFE, just make a web search for it). My favorite freeware file editor for that purpose, although somewhat dated, is still PFE, to be obtained from

http://www.lancs.ac.uk/people/cpaap/pfe/.

The the axis/button assignment of `fgjs` should, at least, get the axis assignments right, its output may need some tweaking. There may be axes moving the opposite way they should, the dead zones may be too small etc. For instance, I had to change

`-prop:/input/joysticks/js[1]/axis[1]/binding/factor=-1.0`

into

`-prop:/input/joysticks/js[1]/axis[1]/binding/factor=1.0`

(USB CH Flightsim Yoke under Windows XP). Thus, here is a short introduction into the assignments of joystick properties.

Basically, all axes settings are specified via lines having the following structure:

`--prop:/input/joysticks/js[`*n*`]/axis[`*m*`]/binding`
`/command=property-scale` (one line)
`--prop:/input/joysticks/js[`*n*`]/axis[`*m*`]/binding`
`/property=/controls/`*steering option* (one line)
`--prop:/input/joysticks/js[`*n*`]/axis[`*m*`]/binding`
`/dead-band=`*db* (one line)
`--prop:/input/joysticks/js[`*n*`]/axis[`*m*`]/binding`
`/offset=`*os* (one line)
`--prop:/input/joysticks/js[`*n*`]/axis[`*m*`]/binding`
`/factor=`*fa* (one line)

where

n	=	number of device (usually starting with 0)
m	=	number of axis (usually starting with 0)
steering option	=	elevator, aileron, rudder, throttle, mixture, pitch
dead-band	=	range, within which signals are discarded; useful to avoid jittering for minor yoke movements
offset	=	specifies, if device not centered in its neutral position
factor	=	controls sensitivity of that axis; defaults to +1, with a value of -1 reversing the behavior

You should be able to at least get your joystick working along these lines. Concerning all the finer points, for instance, getting the joystick buttons working, John Check has written a very useful README being included in the base package to be found under `FlightGear/Docs/Readme/Joystick.html`. In case of any trouble with your input device, it is highly recommended to have a look into this document.

3.6 A glance over our hangar

The following is a Table 1 of all the aircraft presently available for use with ***FlightGear***. In the first column, you will find the name of the aircraft, the second one tells the start option, the third one names the FDM (flight dynamics management model, see Sec. 1.4), and the last column includes some remarks. Here, “no exterior model” means, that there is no aircraft specific external model provided with the base package. As a result, you will see the default blue-yellow glider, when you change to the external view. However, you can download external views for these models from Wolfram Kuss’ site at

http://home.t-online.de/home/Wolfram.Kuss/.

Moreover, this list is complete insofar as it covers all aircraft available via the `--aircraft=` option.

Tab. 1: *Presently available aircraft in **FlightGear**.*

Aircraft type	Start option	FDM	Remarks
Boeing 747	`--aircraft=747-yasim`	YASim	
BA A4 Hawk	`--aircraft=a4-yasim`	YASim	
North American X-15	`--aircraft=X15`	JSBSim	experimental supersonic plane
Airwave Xtreme 150	`--aircraft=airwaveXtreme150-v1-nl-uiuc`	UIUC	hang glider!
Beech 99	`--aircraft=beech99-v1-uiuc`	UIUC	no exterior model
Cessna 172	`--aircraft=c172-3d`	JSBSim	sports a 3D cockpit
Cessna 172	`--aircraft=c172-3d-yasim`	YASim	sports a 3D cockpit
Cessna 172	`--aircraft=c172-ifr`	JSBSim	with IFR panel
Cessna 172	`--aircraft=c172-larcsim`	LaRCsim	
Cessna 172	`--aircraft=c172`	JSBSim	default
Cessna 172	`--aircraft=c172-yasim`	YASim	
Cessna 172p	`--aircraft=c172p-3d`	JSBSim	sports a 3D cockpit
Cessna 172p	`--aircraft=c172p`	JSBSim	
Cessna 172	`--aircraft=c172x`	JSBSim	flight dynamics testbed
Cessna 182	`--aircraft=c182`	JSBSim	
Cessna 310	`--aircraft=c310`	JSBSim	
Cessna 310	`--aircraft=c310-yasim`	YASim	twin-prop machine
Cessna 310U3A	`--aircraft=c310u3a-3d`	JSBSim	twin-prop machine, 3D cockpit
Cessna 310U3A	`--aircraft=c310u3a`	JSBSim	twin-prop machine
Douglas DC-3	`--aircraft=dc3-yasim`	YASim	
BA Harrier	`--aircraft=harrier-yasim`	YASim	no exterior model
Piper Cub J3 Trainer	`--aircraft=j3cub-yasim`	YASim	
Siai Marchetti S.211	`--aircraft=marchetti-v1-uiuc`	UIUC	no exterior model
Space Shuttle	`--aircraft=shuttle`	JSBSim	no exterior model
UFO	`--aircraft=ufo`	JSBSim	‘White Project’ (UNESCO)
1903 Wright Flyer	`--aircraft=wrightFlyer1903-v1-nl-uiuc`	UIUC	historical model
X-24B	`--aircraft=x24b`	JSBSim	USAF/NACA reentry testbed
Cessna 172	`--aircraft=c172-610x`	JSBSim	full screen, hi-res panel (IFR)

Chapter 4

In-flight: All about instruments, keystrokes and menus

The following is a description of the main systems for controlling the program and piloting the plane: Historically, keyboard controls were developed first, and you can still control most of the simulator via the keyboard alone. Later on, they were supplemented by several menu entries, making the interface more accessible, particularly for beginners, and providing additional functionality.

For getting a real feeling of flight, you should definitely consider getting a joystick or – preferred – a yoke plus rudder pedals. In any case, you can specify your device of choice for control via the `--control-mode` option, i.eṡelect joystick, keyboard, mouse. The default setting is joystick.

A short leaflet showing the standard keys can be found at

http://www.flightgear.org/Docs/InstallGuide/FGShortRef.html.

A version of this leaflet can also be opened via ***FlightGear***'s help menu.

4.1 Starting the engine

Depending on your situation, when you start the simulator the engines may be on or off. When they are on you just can go on with the start. When they are off, you have to start them first. The ignition switch for starting the engine is situated in the lower left corner of the panel. It is shown in Fig. 4.

Fig. 4: *The ignition switch.*

It has five positions: “OFF”, “L”, “R”, “BOTH”, and “START”. The extreme right position is for starting the engine. For starting the engine, put it onto the position “BOTH” using the mouse first.

Keep in mind that the mixture lever has to be at 100 % (all the way in) for starting the engine – otherwise you will fail. In addition, advance the throttle to about 25 %.

Operate the starter using the SPACE key now. When pressing the SPACE key you will observe the ignition switch to change to the position “START” and the engine to start after a few seconds. Afterwards you can bring the throttle back to idle (all the way out).

In addition, have a look if the parking brakes are on (red field lit). If so, press the “B” button to release them.

4.2 Keyboard controls

While joysticks or yokes are supported as are rudder pedals, you can fly ***FlightGear*** using the keyboard alone. For proper control of the plane during flight via the keyboard (i) the `NumLock` key must be switched on (ii) the ***FlightGear*** window must have focus (if not, click with the mouse onto the graphics window). Several of the keyboard controls might be helpful even in case you use a joystick or yoke.

After activating `NumLock` the following main keyboard controls for driving the plane should work:

Tab.,2: *Main keyboard controls for **FlightGear** on the numeric keypad with activated* `NumLock`*. [U.S. keyboard uses "." instead of ","]*

Key	Action
9/3	Throttle
4/6	Aileron
8/2	Elevator
0/,	Rudder
5	Center aileron/elevator/rudder
7/1	Elevator trim

For changing views you have to de-activate `NumLock`. Now `Shift` + <`Numeric Keypad Key`> changes the view as follows:

Tab. 3: *View directions accessible after de-activating* `NumLock` *on the numeric keypad.*

Numeric Key	View direction
Shift-8	Forward
Shift-7	Left/forward
Shift-4	Left
Shift-1	Left/back
Shift-2	Back
Shift-3	Right/back
Shift-6	Right
Shift-9	Right/forward

Besides, there are several more options for adapting display on screen:

Tab. 4: *Display options*

Key	Action
P	Toggle instrument panel on/off
c	Toggle3D/2D cockpit (if both are available)
s	Cycle panel style full/mini
Shift-F5/F6	Shift the panel in y direction
Shift-F7/F8	Shift the panel in x direction
Shift-F3	Read a panel from a property list
i/I	Minimize/maximize HUD
h/H	Change color of HUD/toggle HUD off forward/backward
x/X	Zoom in/out
v/V	Cycle view modes forth an back
Ctrl-c	Set view modes to pilot's view
W	Toggle full screen mode on/off (3dfx only)
z/Z	Change visibility (fog) forward/backward
F8	Toggle fog on/off
F2	Refresh Scenery tile cache
F4	Force Lighting update
F9	Toggle texturing on/off
F10	Toggle menu on/off

The autopilot is controlled via the following keys:

Tab. 5: *Autopilot and related controls.*

Key	Action
Ctrl + A	Altitude hold toggle on/off
Ctrl + G	Follow glide slope 1 toggle on/off
Ctrl + H	Heading hold toggle on/off
Ctrl + N	Follow NAV 1 radial toggle on/off
Ctrl + S	Autothrottle toggle on/off
Ctrl + T	Terrain follow toggle on/off
Ctrl + U	Add 1000 ftto your altitude (emergency)
Enter	Increase autopilot heading
F6	Toggle autopilot target: current heading/waypoint
F11	Autopilot altitude dialog
F12	Autopilot heading dialog

Ctrl + T is especially interesting as it makes your little Cessna behave like a cruise missile. Ctrl + U might be handy in case you feel you're just about to crash. (Shouldn't real planes sport such a key, too?)

In case the autopilot is enabled, some of the numeric keypad keys get a special meaning:

Tab. 6: *Special action of keys, if autopilot is enabled. [U.S. keyboard uses "." instead of ","]*

Key	Action
8 / 2	Altitude adjust
0 / ,	Heading adjust
9 / 3	Autothrottle adjust

There are several keys for starting and controlling the engine :
Tab. 7: *Engine control keys*

Key	Action
SPACE	Fire starter on selected engine(s)
!	Select 1st engine
@	Select 2nd engine
#	Select 3rd engine
$	Select 4th engine
{	Decrease Magneto on Selected Engine
}	Increase Magneto on Selected Engine
∼	Select all Engines

Beside these basic keys there are miscellaneous keys for special actions; some of these you'll probably not want to try during your first flight:

Tab. 8: *Miscellaneous keyboard controls.*

Key	Action
B	Toggle parking brake on/off
b	Apply/release all brakes
g/G	Toggle landing gear up/down
,	Left gear brake (useful for differential braking)
.	Right gear brake (useful for differential braking)
l	Toggle tail-wheel lock)
]/[	Extend/Retract flaps
p	Toggle pause on/off
a/A	Speed up/slow down (time acceleration)
t/T	Time speed up/slow down
m/M	Change time offset (warp) used by t/T forward/backward
Shift-F2	Save current flight to `fgfs.sav`
Shift-F1	Restore flight from `fgfs.sav`
F3	Save screen shot under `fgfs-screen.ppm`
Shift-F4	Re-read global preferences from `preferences.xml`
Shift-F9	Toggle data logging of FDM on/off
ESC	Exit program

Note: If you have difficulty processing the screenshot `fgfs-screen.ppm` on a windows machine, just recall that simply pressing the "Print" key copies the screen to the clipboard, from which you can paste it into any graphics program.

These key bindings are not hard coded, but user-adjustable. You can check and change these setting via the file `keyboard.xml` to be found in the main ***FlightGear*** directory. This is a human readable plain ASCII file. Although it's perhaps not the best idea for beginners to start just with modifying this file, more advanced users will find it useful to change key bindings according to what they like (or, perhaps, know from other simulators).

4.3 Menu entries

By default, the menu is disabled after starting the simulator (you don't see a menu in a real plane, do you?). You can turn it on using the F10 key. To hide the menu, just hit F10 again.

The menu provides the following sub-menus and options.

- **File**
  - **Save flight** Saves the current flight, by default to `fgfs.sav`.
  - **Load flight** Loads the current flight, by default from `fgfs.sav`. You should start ***FlightGear*** using the same options (aircraft, airport...) as when you saved the flight.

  - **Reset** Resets you to the selected starting position. Comes handy in case you got lost or something went wrong.
  - **Hires Snap Shot** Saves a high resolution Screen Shot as `fgfs-screen-XXX.ppm` under the directory you started the program from.
  - **Snap Shot** Saves a normal resolution Screen Shot as `fgfs-screen-XXX.ppm` under the directory you started the program from.
  - **Print** Prints screen shot (Linux only).
  - **Sound Configuration** Allows you to mute sound, set the volume and enable/disable ATC Chatter. ATC Chatter plays a number of real-life ATC communications to add realism.
  - **Browse Internal Properties** Displays a tree view of all the properties within the system. You can navigate through the tree like a graphical directory listing and set properties by clicking on them
  - **Logging** Allows you to log various pieces of flight information to a file. You can set the file to log to, the properties to be logged and the interval between logs.
  - **Quit** Exits the program.

- **View**
  - **Toggle 2D Panel** Switches between a 2D panel and a 3D cockpit (if available).
  - **Rendering Options** Displays a dialog allowing you to toggle various advanced graphical options. This allows you to trade eye-candy such as shadows, 3D clouds and specular reflections for frame-rate. To help you achieve a good balance, enable the "Show Frame Rate" option. This displays the current frame-rate in frames-per-second in the bottom right of the screen. Most people find a frame-rate of around 20fps adequate for flying. The frame-rate is affected by the graphical options you have enabled, the current visibility (set by Z/z), the number of objects in view and their level of detail (LOD).
  - **Adjust View Distance** Displays a dialog showing the current view offset. You can adjust this by dragging the dials. Alternatively you can make small adjustments to your view-point using the mouse (see below).
  - **Adjust HUD Transparency** Displays a dialog allowing you to set the transparency of the HUD and whether anti-aliasing is used to display it.
  - **Instant Replay** Displays a dialog to control the instant replay feature. A good tool for checking your landings! Press “p” to end the replay and pause the flight.

  - **Adjust LOD Ranges** Displays a dialog allowing you to set the range at which different levels of detail are displayed. This affects the textures and objects displayed in the simulator.

- **Location**

  - **Position Aircraft (on ground)** Displays a dialog allowing you to position the aircraft on the runway of any installed airport. You need to know the ICAO code for the airport you wish to start from (e.g. KSFO for San Fransisco International).
  - **Position Aircraft (in air)** Displays a dialog allowing you to position the aircraft at an arbitary point in the air. You must select a known ground point, e.g. an airport, VOR, long/lat coordinates, and a position relative to that point, e.g distance, direction, altitude. You can also set your initial speed and heading. This is useful for practising approaches.
  - **Select Airport from List** This allows you to select an airport without knowing its ICAO code. You can search amongst all the airports that you have installed. Clicking Apply will place you at that airport on a runway appropriate for the current wind.
  - **Random Attitude** Sets the aircraft with a random heading, speed and attitude. Useful for practising recovery from unusual attitudes.
  - **Tower position** Displays a dialog allowing you to change the airport tower used for the Tower View and Tower View Look From.

- **Autopilot** This menu is only available for aircraft that have the default autopilot configured. Other aircraft may have their own autopilot which is configured through the panel.

  - **Autopilot Settings** Displays a dialog allowing you to set the aircraft autopilot. You can set the autopilot up in a variety of different ways - from simply keeping the wings level, to following an ILS.
  - **Add Waypoint** Adds waypoint to waypoint list. The waypoint can be an airport of a fix. The distance and time to the waypoint is displayed in the HUD. Additionally, the heading to the current waypoint is also displayed
  - **Pop Waypoint** Pops the top waypoint from the list.
  - **Clear Route** Clears current route.
  - **Set Lat/Lon Format** Toggles the HUD Latitude/Longitude format between decimal minutes and seconds.

- **Weather**

  - **Weather Scenario** Displays a dialog showing the current weather reported by the closest weather station (usually an airport) as a METAR.

You can change the weather scenario between Fair Weather (clear skies, few clouds, little wind), a Thunderstorm (clouds, rain, lightning), the current METAR, or none.

- **Weather Conditions** Displays a dialog allowing you to set the wind direction, wind speed, turbulence, visibility, temperature, dew point, barometer setting at various altitudes.
- **Clouds** Displays a dialog allowing you to set the cloud types, elevations and thicknesses. Note that this affects 2D clouds only. 3D clouds (if enabled) are configured through the Rendering Options menu.
- **Time of Day** Displays a dialog allowing you to set the current time in the simulator to system time, dawn, morning, night etc. Also displays UTC and local time.

- **Equipment**
  - **Fuel and Payload** For aircraft that support it, allows you to set the fuel and levels and current payload within the aircraft.
  - **Radio Settings** Displays a dialog allowing you to set the frequencies and radials being used by the radios and navigational equipment.
  - **GPS Settings** Displays a dialog allowing you to set waypoints and view course information for the GPS.
  - **Instrument Settings** Displays a dialog allowing you to set the altimeter pressure and Heading Indicator offset.
  - **System Failures** Displays a dialog allowing you to fail various aircraft systems, such as the vacuum.
  - **Instrument Failures** Displays a dialog allowing you to fail specific aircraft instruments.

- **ATC/AI**
  - **Frequencies** Displays a dialog allowing you enter the ICAO code for an airport (or simply click on one of the buttons listing the local airports) and retrieve the radio frequencies for ATIS, and Tower communications.
  - **Options** Displays a dialog allowing you to enable Air Traffic Control (ATC) and computer-generated traffic. You may also set the AI traffic density from 1 (few aircraft) to 3 (busy skies!). This menu also allows you to control the aircraft carriers in the game (see below for details).

- **Debug** The debug menu contains various options outside the scope of this guide.

- **Help**

- **Help** Opens the help system in a browser window.
- **Basic Keys** Lists the basic keys for the controlling the simulator
- **Common Aircraft Keys** Lists the basic keys for controlling the aircraft
- **Aircraft Help** Displays information specific to the aircraft.
- **Start Tutorial** Displays a dialog allowing the user to select a tutorial on the current aircraft. This is only available on some aircraft. See Tutorials below for details.
- **End Tutorial** Ends the current tutorial.

4.4 The Instrument Panel

The Cessna instrument panel is activated by default when you start ***FlightGear***, but can be de-activated by pressing the "P" key. While a complete description of all the functions of the instrument panel of a Cessna is beyond the scope of this guide, we will at least try to outline the main flight instruments or gauges.

All panel levers and knobs can be operated with the mouse To change a control, just click with the left/middle mouse button on the corresponding knob/lever.

Let us start with the most important instruments any simulator pilot must know. In the center of the instrument panel (Fig. 5), in the upper row, you will find the artificial horizon (attitude indicator) displaying pitch and bank of your plane. It has pitch marks as well as bank marks at 10, 20, 30, 60, and 90 degrees.

Left to the artificial horizon, you'll see the airspeed indicator. Not only does it provide a speed indication in knots but also several arcs showing characteristic velocity rages you have to consider. At first, there is a green arc indicating the normal operating range of speed with the flaps fully retracted. The white arc indicates the range of speed with flaps in action. The yellow arc shows a range, which should only be used in smooth air. The upper end of it has a red radial indicating the speed you must never exceeded - at least as long as you won't brake your plane.

Below the airspeed indicator you can find the turn indicator. The airplane in the middle indicates the roll of your plane. If the left or right wing of the plane is aligned with one of the marks, this would indicate a standard turn, i.eȧ turn of 360 degrees in exactly two minutes.

Below the plane, still in the turn indicator, is the inclinometer. It indicates if rudder and ailerons are coordinated. During turns, you always have to operate aileron and rudder in such a way that the ball in the tube remains centered; otherwise the plane is skidding. A simple rule says: "Step onto the ball", i.eṡtep onto the left rudder pedal in case the ball is on the l.h.s.

Fig. 5: *The panel.*

If you don't have pedals or lack the experience to handle the proper ratio between aileron/rudder automatically, you can start ***FlightGear*** with the option `--enable-auto-coordination`.

To the r.h.s of the artificial horizon you will find the altimeter showing the height above sea level (not ground!) in hundreds of feet. Below the altimeter is the vertical speed indicator indicating the rate of climbing or sinking of your plane in hundreds of feet per minute. While you may find it more convenient to use then the altimeter in cases, keep in mind that its display usually has a certain lag in time.

Further below the vertical speed indicator is the RPM (rotations per minute) indicator, which displays the rotations per minute in 100 RPMs. The green arc marks the optimum region for long-time flight.

The group of the main instruments further includes the gyro compass being situated below the artificial horizon. Besides this one, there is a magnetic compass sitting on top of the panel.

Four of these gauges being arranged in the from of a "T" are of special importance: The air speed indicator, the artificial horizon, the altimeter, and the compass should be scanned regularly during flight.

Besides these, there are several supplementary instruments. To the very left you will find the clock, obviously being an important tool for instance for determining turn rates.Below the clock there are several smaller gauges displaying the technical state of your engine. Certainly the most important of them is the fuel indicator - as

any pilot should know.

The ignition switch is situated in the lower left corner of the panel (cf. Fig. 4). It has five positions: “OFF”, “L”, “R”, “BOTH”, and “START”. The first one is obvious. “L” and “R” do not refer to two engines (actually the Cessna does only have one) but to two magnetos being present for safety purposes. The two switch positions can be used for test puposes during preflight. During normal flight the switch should point on “BOTH”. The extreme right position is for using a battery-powered starter (to be operated with the SPACE key in flight gear).

Like in most flight simulators, you actually get a bit more than in a real plane. The red field directly below the gyro compass displays the state of the brakes, i.e., it is lit in case of the brakes being engaged. The instruments below indicate the position of youryoke. This serves as kind of a compensation for the missing forces you feel while pushing a real yoke. Three of the arrows correspond to the three axes of your yoke/pedal controlling nose up/down, bank left/right, rudder left/right, and throttle. (Keep in mind: They do **not** reflect the actual position of the plane!) The left vertical arrow indicates elevator trim.

The right hand side of the panel is occupied by the radio stack. Here you find two VOR receivers (NAV), an NDB receiver (ADF) and two communication radios (COMM1/2) as well as the autopilot.

The communication radio is used for communication with air traffic facilities; it is just a usual radio transceiver working in a special frequency range. The frequency is displayed in the “COMM” field. Usually there are two COM transceivers; this way you can dial in the frequency of the next controller to contact while still being in contact with the previous one.

The COM radio can be used to display ATIS messages as well. For this purpose, just to dial in the ATIS frequency of the relevant airport.

The VOR (Very High Frequency Omni-Directional Range) receiver is used for course guidance during flight. The frequency of the sender is displayed in the ”NAV” field. In a sense, a VOR acts similarly to a light house permitting to display the position of the aircraft on a radial around the sender. It transmits one omni-directional ray of radio waves plus a second ray, the phase of which differs from the first one depending on its direction (which may be envisaged as kind of a “rotating” signal). The phase difference between the two signals allows evaluating the angle of the aircraft on a 360 degrees circle around the VOR sender, the so-called radial. This radial is then displayed on the gauges NAV1 and NAV2, resp., left to frequency field. This way it should be clear that the VOR display, while indicating the position of the aircraft relative to the VOR sender, does not say anything about the orientation of the plane.

Below the two COM/NAV devices is an NDB receiver called ADF (automatic direction finder). Again there is a field displaying the frequency of the facility. The ADF can be used for navigation, too, but contrary to the VOR does not show the position of the plane in a radial relative to the sender but the direct heading from the aircraft to the sender. This is displayed on the gauge below the two NAV gauges.

Above the COMM1 display you will see three LEDs in the colors blue, amber,

and white indicating the outer, middle, and, inner, respṁarker beacon. These show the distance to the runway threshold during landing. They to not require the input of a frequency.

Below the radios you will find the autopilot. It has five keys for WL = "Wing-Leveler", "HDG" = "Heading", NAV, APR = "Glide-Slope", and ALT = "Altitude". These keys when engaged hold the corresponding property.

You can change the numbers for the radios using the mouse. For this purpose, click left/right to the circular knob below the corresponding number. The corresponding switch left to this knob can be used for toggling between the active/standby frequency.

A detailed description of the workings of these instruments and their use for navigation lies beyond this Guide; if you are interested in this exciting topic, we suggest consulting a book on instrument flight (simulation). Besides, this would be material for a yet to be written ***FlightGear*** Flight School.

It should be noted, that you can neglect these radio instruments as long as you are strictly flying according to VFR (visual flight rules). For those wanting to do IFR (instrument flight rules) flights, it should be mentioned that ***FlightGear*** includes a huge database of navaids worldwide.

Finally, you find the throttle, mixture, and flap control in the lower right of the panel (recall, flaps can be set via [and] or just using the mouse).

As with the keyboard, the panel can be re-configured using configuration files. As these have to be plane specific, they can be found under the directory of the corresponding plane. As an example, the configuration file for the default Cessna C172 can be found at `FlightGear/Aircraft/c172/Panels` as `c172-panel.xml`. The accompanying documentation for customizing it (i.eṡhifting, replacing etcġauges and more) is contained in the file `README.xmlpanel` written by John Check, to be found in the source code in the directory `docs-mini`.

Most aircraft also have a 3D cockpit as an alternative to the 2D panel mentioned above (see Fig. 6). Its functionality is the same as that of the 2D panel mentioned above, but it gives a much more realistic view, while instruments may be better readable in the 2D cockpit.

Fig. 6: *The 3D cockpit of the Cessna 172.*

4.5 The Head Up Display

At current, there are two options for reading off the main flight parameters of the plane: One is the instrument panel already mentioned, while the other one is the HUD (**H**ead **U**p **D**isplay) . Neither are HUDs used in usual general aviation planes nor in civilian ones. Rather they belong to the equipment of modern military jets. However, some might find it easier to fly using the HUD even with general aviation aircraft. Several Cessna pilots might actually love to have one, but technology is simply too expensive for implementing HUDs in general aviation aircraft. Besides, the HUD displays several useful figures characterizing simulator performance, not to be read off from the panel.

The HUD shown in Fig. 7 displays all main flight parameters of the plane. In the center you find the pitch indicator (in degrees) with the aileron indicator above and the rudder indicator below. A corresponding scale for the elevation can be found to the left of the pitch scale. On the bottom there is a simple turn indicator.

There are two scales at the extreme left: The inner one displays the speed (in kts) while the outer one indicates position of the throttle. The Cessna 172 takes off at around 55 kts. The two scales on the extreme r.h.s display your height, i. ėhe left one shows the height above ground while the right of it gives that above zero, both being displayed in feet.

Besides this, the HUD delivers some additions information. On the upper left you will find date and time. Besides, latitude and longitude, resp., of your current position are shown on top.

You can change color of the **HUD** using the “H” or “’h” key. Pressing the toggle “i/I” minimizes/maximizes the HUD.

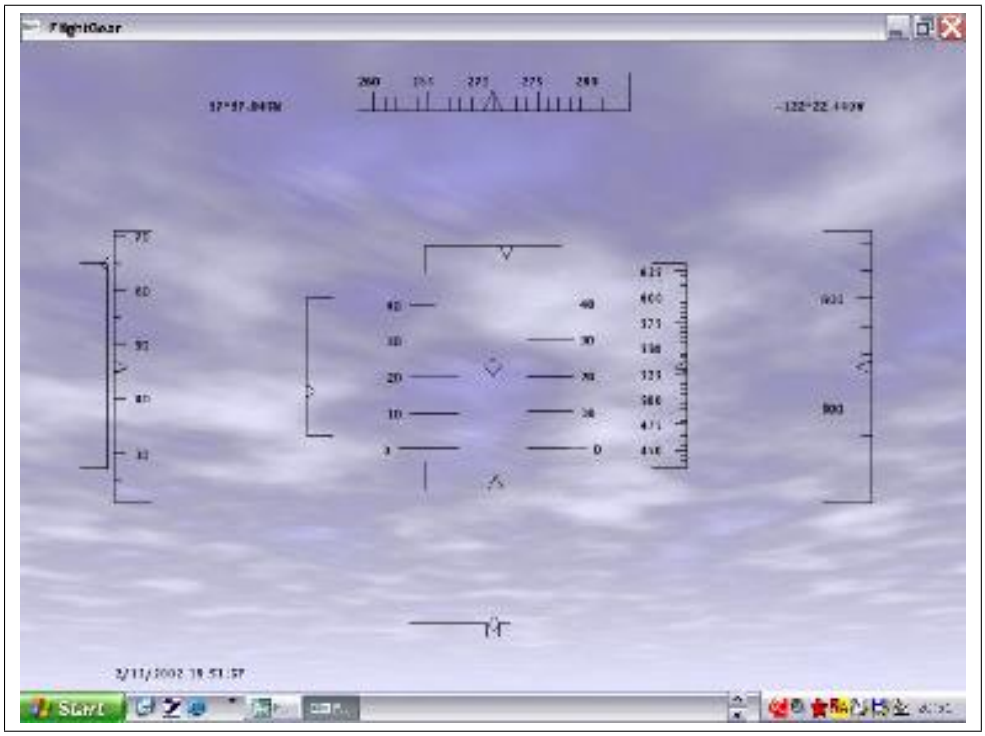


Fig. 7: *The HUD, or Head Up Display.*

4.6 Mouse controlled actions

Besides just clicking the menues, your mouse has got certain valuable functions in ***FlightGear***.

There are three mouse modes. In the normal mode (pointer cursor) panel’s controls can be operated with the mouse. To change a control, click with the left/middle mouse button on the corresponding knob/lever. While the left mouse button leads to small increments/decrements, the middle one makes greater ones. Clicking on the left hand side of the knob/lever decreases the value, while clicking on the right hand side increases it.

Right clicking the mouse activates the simulator control mode (cross hair cursor). This allows control of aileron/elevator via the mouse in absence of a joystick/yoke (enable `--enable-auto-coordination` in this case). If you have a joystick you certainly will not make use of this mode

Right clicking the mouse another time activates the view control mode (arrow cursor). This allows changing direction of view, i.e. pan and tilt the view, via the mouse. Clicking the left mouse button resets the view. Dragging using the middle mouse button moves the viewpoint itself.

Right clicking the mouse once more resets it into the initial state.

If you are looking for some interesting places to discover with ***FlightGear*** (which may or may not require downloading additional scenery) you may want to check

http://www.flightgear.org/Places/.

There is now a menu entry for entering directly the airport code of the airport you want to start from.

Finally, if you're done and are about to leave the plane, just hit the ESC key or use the corresponding menu entry to exit the program. It is not suggested that you simply "kill" the simulator using Ctrl-C on the text window.

Chapter 5

Features

FlightGear contains many special features, some of which are not obvious to the new user. This section describes how to enable and make use of some of the more advanced features.

Many of the features are under constant development, so the information here may not be completely up-to-date. For the very latest information (and new features), see the ***FlightGear*** Wiki, available from
http://www.seedwiki.com/wiki/flight_gear/flight_gear.cfm

5.1 Aircraft Carrier

FlightGear supports carrier operations on the Nimitz, (located near San Fransisco) and Eisenhower. The carriers are equipped with working catapult, arrester wires, elevators, TACAN and FLOLS and are currently available for aircraft using the YASim FDM (in particular the Seahawk, Seafire and A4F.)

To enable the carrier, you must edit your preferences.xml file in $FG_ROOT using a text editor (e.g. Notepad under Windows). Search for the word “nimitz”. You ought to find something that looks like this;

```
<!--<scenario>nimitz_demo</scenario>-->
```

You should remove the “comment” marks so that it looks like this;

```
<scenario>nimitz_demo</scenario>
```

Also ensure that the line above that referring to ai being enabled is set to "true"
Save the file and quit the text editor.

5.1.1 Starting on the Carrier

You are now ready to start FlightGear. To position your aircraft on the carrier at startup, use the following command line options (noting the upper-case "N");

```
--carrier=Nimitz --aircraft=seahawk
```

Note that several FG aircraft are carrier capable, but the seahawk is possibly the easiest to fly to begin with.

If you are using the Windows or OSX launcher to run FG, you should find a text entry box in the gui that allows you to specify command line options, add the above options there. Linux or Cygwin users can just add them to their usual startup command:

```
fgfs --carrier=Nimitz --aircraft=seahawk
```

Please note the uppercase “N’ in “Nimitz’.

5.1.2 Launching from the Catapult

Once FlightGear has started, you should ensure that the parking brakes are off and press “L” to engage the launchbar (this might be best done from an external view initially.) You should notice the aircraft being pulled into alignment with the catapult and see the strops appear and hold down the aircraft. This will only happen if your aircraft is close enough to the correct spot on the catapult; as a rough guide, for the default parking position the seahawk’s nose should be rougly level with the deck observation bubble.

To get the carrier into as good a position as possible for launch, select the “ATC/AI” menu, then check the "Turn into wind" box under the “AI Carrier” section. You should now notice the carrier begin to pick up speed and turn into the wind, and naturally the deck may tilt somewhat as it turns. You should wait for this maneuver to finish and the deck to return to level before moving on to the next stage.

Being engaged to the catapult, you should spool up the engines to full power, ensure the brakes are off and that all flight controls are in a suitable position for launch (stick held right back with the seahawk.) When ready, press “C” to release the catapult. Your aircraft will be hurled forward off the deck, and you should be able to raise the undercarriage and climb slowly away, being careful to avoid stalling.

5.1.3 Finding the Carrier - TACAN

Actually finding the carrier in a vast expanse of open water can be very difficult, especially if visibility is poor. To assist with this task, Nimitz is equipped with TACAN, which allows a suitably-equipped aircraft (Seahawk at present) to obtain a range and bearing to the carrier. First, you must set the appropriate TACAN channel, 029Y in this case, in the radios dialogue (ctrl-r or choose Equipment/Radio Settings from the FG menubar). You should, if within range, notice the DME instrument show your distance from the carrier, and the ADF instrument (next to the DME in the seahawk) should indicate a bearing to the carrier. Turn to the indicated heading and you should see the DME dial indicate your closing in on the carrier.

5.1.4 Landing on the Carrier

This is the most difficult part of the operation, as in real life. You might well find Andy Ross' tutorial on operating the A4 Skyhawk useful here. It is available from here:

http://www.seedwiki.com/wiki/flight_gear/a4_skyhawk.cfm?wpid=209330

Basically you should use the TACAN to locate the carrier, and line up with the rear of the deck. As this part of the deck is at an angle to the course of the vessel, you may need to correct your alignment often. Ensure that the aircraft is in the correct configuration for approach (the Help/Aircraft Help menu should contain useful data for your aircraft) and that the gear and the arrestor hook are down.

As you approach you should see, on the left hand side of the deck, a set of brightly coloured lights - called the Fresnel Lens Optical landing System (FLOLS). This indicates your position on the landing glideslope. You will see a horizontal row of green lights, and when approximately on the glideslope, an orange light (known in some circles as the "meatball") approximately in line with the green lights. When approaching correctly, the meatball appears in line with the green lights. If you are high it is above, and when low it is below. If you are very low the meatball turns red. If you fly to keep the meatball aligned you should catch number 3 wire.

Carrier landings are often described as "controlled crashes" and you shouldn't waste your time attempting to flare and place the aircraft gently on the deck like you would with a conventional landing - ensuring that you catch the wires is the main thing.

Immediately your wheels touch the deck, you should open the throttles to full power, in case you have missed the wires and need to "go around" again; the wires will hold the aircraft if you have caught them, even at full power.

If you wish, you can then (with 0.9.10 and later) raise the elevators from the ATC/AI menu, taxy onto one of the elevators, lower it (uncheck the box on the menu) and taxy off into the hangar.

Don't be discouraged if you don't succeed at first - it's not an easy maneouver to master. If after a little practice you find the Seahawk too easy, you could move on to the Seafire for more of a challenge!

5.2 Atlas

Atlas is a "moving map" application for FlightGear. It displays the aircraft in relation to the terrain below, along with airports, navigation aids and radio frequencies.

Further details can be found on the Atlas website:

http://atlas.sourceforge.net

5.3 Multiplayer

FlightGear supports a multiplayer environment, allowing you to share the air with other flight-simmers. For server details and to see who is online (and where they are flying), have a look at the excellent multiplayer map, available from http://mpmap01.flightgear.org

Click on the 'server' tab to see a list of multiplayer servers. At time of writing there are two sets - one for official ***FlightGear*** releases, and one for the current development stream (CVS). The servers within each set are connected.

5.3.1 Quick Start

To connect to a server, note down the server name (usually mpserver0?.flightgear,org) and port number (usually 5000) for the appropriate server closest to you, and start ***FlightGear*** as follows.

Using the FlightGear Launcher

The final screen of the FlightGear Launcher has a section for Multiplayer. Simply select the checkbox, enter the hostname and port number you noted above and choose a callsign to identify yourself. Your callsign can be up to 7 characters in length.

Using the Command Line

The basic arguments to pass to fgfs for multiplayer are these:

```
--multiplay=out,10,<server>,<portnumber>
--multiplay=in,10,<client>,<portnumber>
--callsign=<anything>
```

Where

1. <portnumber> is the port number of the server e.g. 5000.
2. <server> is the name of the multiplayer server e.g. mpserver01.flightgear.org.
3. <client> is the name of your computer, or the IP address ip address of the network interface being used by FG to connect to the server - even if that's a local 192.168 type address. e.g. 192.168.0.1
4. <callsign> is the call sign to identify yourself, up to 7 characters, e.g. N-FGFS.

Once the simulator has loaded, you should see yourself displayed on the map. If you don't, check the console for error messages and see the Troubleshooting section below.

5.3.2 Troubleshooting

To get multiplayer to work, we need information about the IP address of our computer and the ability to communicate with the server. How to get this information depends on your configuration and is described below.

Those using a USB modem to connect to the Internet

First of all, you need to know the IP address of the network interface you're going to be running FG multiplayer over. If your Internet connection is via an ADSL modem that plugs directly into your computer with a USB connection, you should be able to find your IP address by visiting http://www.whatismyip.com . Please note that this address may very well change every now and again - if MP stops working, check this first.

Those using some kind of Ethernet router to connect to the Internet

Otherwise, your connection is likely via some kind of router that connects to your computer via an RJ-45, or "Ethernet" connector (similar shape to most Western telephone plugs), or by a wireless link. You need to find the IP address of that network interface.

Under linux, this can be found by logging in as root and typing "ifconfig". You may find more than one interface listed, beginning with "lo" - ignore that one. You should have something like "eth0" or "wlan0" also listed - look through this block of text for "inet addr". This will be followed directly by the number you're looking for, e.g. "inet addr:192.168.0.150"

Under Windows XP, click start, run, and type "cmd". In the terminal window which appears, type "ipconfig" This should show you your IP address - note it down.

With Windows 98, click start, run, and type "winipcfg" to get information about your IP address.

Configuring your router

This section ought to be unnecessary now with recent versions of the FG server. If you have problems though, it won't hurt to follow through.

Now, all(!) that remains is to configure your router to forward UDP port 5000 or 5002 to the IP address you've just found. This is not something that can be described in step-by-step detail, because each manufacturer's configuration interfaces differ greatly. Some tips are given here - if you get stuck, ask nicely on the FlightGear IRC channel for help (details on the flightgear website).

You should know how to log on to your router's configuration page, usually via a web browser. You are looking for settings pertaining to "port forwarding" "virtual server" "Forwarding Rules" or similar. When you have found the relevant settings, you need to add a rule that forwards port 5000 or 5002 (depending on which server

you wish to join - add both if you like) to the IP address you discovered earlier. If there is a choice given, ensure it is UDP ports that are forwarded. If there is no choice, you may assume that both TCP and UDP are being forwarded. Save your configuration, and most routers will probably then need to be rebooted to apply the changes.

Note: (for BSD users) If you are using a ADSL modem, you might have to put the port forward command into the ppp.conf file rather than firewall. This is because the firewall script will only run each time the machine is booted rather than the ppp line coming back online.

Starting Multiplayer FlightGear

Finally, start FG using the command line given right at the start (if you're using the windows launcher you will find entry boxes for Multiplayer arguments - insert the relevant details there). You will end up with something like this;

```
fgfs --callsign=MyName
     --multiplay=in,10,192.168.0.2,5000
     --multiplay=out,10,202.83.200.172,5000
     --airport=KSFO
     --runway=28R
     --aircraft=hunter
```

The current server IP address (in the "out" section) can be found by asking on the IRC channel, and likewise the relevant port number; 5000 is the default. Choose your own callsign - this is currently limited to seven characters.

Once you have started FG, you should, if others are flying, see messages in the terminal from which FG was started, similar to the following;

```
Initialising john51a using 'Aircraft/ufo/Models/ufo.xml'
FGMultiplayRxMgr::ProcessRxData - Add new player. IP: 10.0.0.36,
Call: john51a,model: Aircraft/ufo/Models/ufo.xml
```

The MultiPlayer Map is available at http://mpmap01.flightgear.org - this should show you if anyone else is currently flying, and where.

If It Still Doesn't Work

You MUST give your local, behind-the-router IP address for MultiPlayer to work. Trust me on this one!

You should check that your firewall is not causing problems - either turn it off temporarily or add an exception to allow incoming connections on port 5000 and 5002.

If it's still just not working for you, ask nicely on the FlightGear IRC channel and someone should be able to assist.

5.4 Multiple Displays

FlightGear allows you to connect multiple instances of the program together to display different views of the simulation though a highly flexible I/O subsystem.

For example, you may want to have the aircraft panel displayed on a screen right infront of you, while the view forward is displayed on a separate screen or using a projector. Using multiple displays can vastly improve the realism of the simulation.

Given enough hardware, you can create sophisticated simulation environments with mock-up cockpits, panels, multiple views, and even a separate control station allowing an instructor to fail instruments, change the weather etc. An example of this is the 747 cockpit project.
http://www.flightgear.org/Projects/747-JW/

5.4.1 Hardware

Each instance of ***FlightGear*** can support a single display. Due to the complexity of the FDM and graphics, ***FlightGear*** is very processor-intensive, so running multiple instances of ***FlightGear*** on a single machine is not recommended.

You will therefore need a computer for each view of the simulation you wish to display, including the panel. The computers obviously must be networked and for simplicity should be on the same subnet.

One computer is designated as the master. This computer will run the FDM and be connected to controls such as yokes, joysticks and pedals. As the machine is running the FDM, it usually only displays a simple view, typically the main panel, to maximize performance.

All other computers are designated as slaves. They are purely used for display purposes and receive FDM information from the master.

5.4.2 Basic Configuration

Creating a basic configuration for multiple displays is straightforward. The master computer needs to broadcast the FDM and control information to the slaves. This is done using the following command line options:

```
--native-fdm=socket,out,60,,5505,udp
--native-ctrls=socket,out,60,,5506,udp
```

The slave computers need to listen for the information, and also need to have their own FDMs switched off:

```
--native-fdm=socket,in,60,,5505,udp
--native-ctrls=socket,in,60,,5506,udp
--fdm=null
```

5.4.3 Advanced Configuration

The options listed above will simply show the same view on both machines. You will probably also want to set the following command-line options on both master and slave computers.

```
--enable-game-mode  (full screen for glut systems)
--enable-full-screen (full screen for sdl or windows)
--prop:/sim/menubar/visibility=false (hide menu bar)
--prop:/sim/ai/enabled=false (disable AI ATC)
--prop:/sim/ai-traffic/enabled=false (disable AI planes)
--prop:/sim/rendering/bump-mapping=false
```

If using the master computer to display a panel only, you may wish to create a full-screen panel for the aircraft you wish to fly (one is already available for the Cessna 172), and use the following options.

```
--prop:/sim/rendering/draw-otw=false (only render the panel)
--enable-panel
```

For slave computers displaying side-views, use the following options.

```
--fov=35
--prop:/sim/view/config/heading-offset-deg=-35
--prop:/sim/view/config/pitch-offset-deg=3
```

5.5 Recording and Playback

Another feature of the I/O system is the ability to record your flight for later analysis or playback. Technical details of how to record specific FDM information can be found in the $FG_ROOT/protocol/README.protocol file.

To record a flight, use the following command line options:

```
--generic=file,out,20,flight.out,playback.xml
```

This will record the FDM state at 20Hz (20 times per second), using the playback protocol and write it to a file flight.out.

To play it back later, use the following command line options:

```
--generic=file,in,20,flight.out,playback.xml
--fdm=external
```

The playback.xml protocol file does not include information such as plane type, time of day, so you should use the same set of command line options as you did when recording.

5.6 Text to Speech with Festival

FlightGear supports Text To Speech (TTS) for ATC and tutorial messages through the festival TTS engine (http://www.cstr.ed.ac.uk/projects/festival/). This is available on many Linux distros, and can also be installed easily on a Cygwin Windows system. At time of writing, support on other platforms is unknown.

5.6.1 Installing the Festival system

1. Install festival from http://www.cstr.ed.ac.uk/projects/festival/

2. Check if Festival works. Festival provides a direct console interface. Only the relevant lines are shown here. Note the parentheses!

```
$ festival
festival> (SayText "FlightGear")
festival> (quit)
```

3. Check if MBROLA is installed, or download it from here:

   http://tcts.fpms.ac.be/synthesis/mbrola/

   See under "Downloads"m "MBROLA binary and voices" (link at the bottom; hard to find). Choose the binary for your platform. Unfortunately, there's no source code available. If you don't like that, then you can skip the whole MBROLA setup. But then you can't use the more realistic voices. See below for details of more voices. Run MBROLA and marvel at the help screen. That's just to check if it's in the path and executable.

```
$ mbrola -h
```

5.6.2 Running FlightGear with Voice Support

First start the festival server:

```
$ festival --server
```

Now, start ***FlightGear*** with voice support enabled. This is set through the /sim/sound/voices/enabled property. You can do this through the command line as follows.

```
$ fgfs --aircraft=j3cub \
       --airport=KSQL \
       --prop:/sim/sound/voices/enabled=true
```

Of course, you can put this option into your personal configuration file. This doesn't mean that you then always have to use FlightGear together with Festival. You'll just get a few error messages in the terminal window, but that's it. You cannot enable the voice subsystem when FlightGear is running.

To test it is all working, contact the KSFO ATC using the ' key. You should hear "your" voice first (and see the text in yellow color on top of the screen), then you should hear ATC answer with a different voice (and see it in light-green color).

ou can edit the voice parameters in the preferences.xml file, and select different screen colors and voice assignments in $FG_ROOT/Nasal/voice.nas. The messages aren't written to the respective /sim/sound/voices/voice[*]/text properties directly, but rather to aliases /sim/sound/voices/atc,approach,ground,pilot,ai-plane.

5.6.3 Troubleshooting

On some Linux distros, festival access is restricted, and you will get message like the following.

```
client(1) Tue Feb 21 13:29:46 2006 : \
  rejected from localhost.localdomain
not in access list
```

Details on this can be found from:
http://www.cstr.ed.ac.uk/projects/festival/manual/festival_28.html#SEC130.

You can disable access restrictions from localhost and localhost.localdomain by adding the following to a .festivalrc file in $HOME:

```
(set! server_access_list '("localhost"))
(set! server_access_list '("localhost.localdomain"))
```

Or, you can just disable the access list altogether:

```
(set! server_access_list nil)
```

This will allow connections from anywhere, but should be OK if your machine is behind a firewall.

5.6.4 Installing more voices

I'm afraid this is a bit tedious. You can skip it if you are happy with the default voice. First find the Festival data directory. All Festival data goes to a common file tree, like in FlightGear. This can be /usr/local/share/festival/ on Unices. We'll call that directory $FESTIVAL for now.

1. Check which voices are available. You can test them by prepending "voice_":

```
$ festival
festival> (print (mapcar (lambda (pair) (car pair)) \
                                        voice-locations))
(kal_diphone rab_diphone don_diphone us1_mbrola \
                    us2_mbrola us3_mbrola en1_mbrola)
nil
festival> (voice_us3_mbrola)
festival> (SayText "I've got a nice voice.")
festival> (quit)
```

2. Festival voices and MBROLA wrappers can be downloaded here:

   http://festvox.org/packed/festival/1.95/

   The "don_diphone" voice isn't the best, but it's comparatively small and well suited for "ai-planes". If you install it, it should end up as directory $FESTIVAL/voices/english/don_diphone/. You also need to install "festlex_OALD.tar.gz" for it as $FESTIVAL/dicts/oald/ and run the Makefile in this directory. (You may have to add "–heap 10000000" to the festival command arguments in the Makefile.)

3. Quite good voices are "us2_mbrola", "us3_mbrola", and "en1_mbrola". For these you need to install MBROLA (see above) as well as these wrappers: festvox_us2.tar.gz, festvox_us3.tar.gz, and festvox_en1.tar.gz. They create directories $FESTIVAL/voices/english/us2_mbrola/ etc. The voice data, however, has to be downloaded separately from another site:

4. MBROLA voices can be downloaded from the MBROLA download page (see above). You want the voices labeled "us2" and "us3". Unpack them in the directories that the wrappers have created: $FESTIVAL/voices/english/us2_mbrola/ and likewise for "us3" and "en1".

5.7 Air-Air Refuelling (AAR; feature available past 0.9.10)

5.7.1 What's possible

At present, there are two tanker aircraft (KC135-E and KA6-D) and three receiving aircraft (A4F, Lightning and T38) capable of in-air refuelling. When flying one of these aircraft in the default scenery area, one can locate the tanker aircraft using air-air TACAN and/or radar and then receive a full or partial load of fuel by flying in close formation behind the tanker. Refuelling is also possible between aircraft in a MultiPlayer session. The KC135 is a boom refueller, while the KA6 has a hose. The A4F and Lightning are both fitted with a probe for hose refuelling while the T38 is fitted with a boom receiver. At the moment, either type can refuel from any tanker, but in the future it is likely that the correct type will have to be used.

5.7.2 Necessary preparations

Like the aircraft carriers, AAR is implemented as an "AI scenario". Selecting these normally requires editing the "preferences.xml" file in the flightgear data directory.

There is a shortcut in this case though; simply selecting the Lightning, A4F or T38 should automatically load a scenario containing a tanker, assuming you haven't changed anything in your preferences.xml file.

Assuming this is the case, choose one of the aforementioned aircraft, make sure that "AI models" are enabled and start at KSFO (the default airport.)

Depending on the scenario, you might see the tanker crossing overhead when the sim starts; if not, don't worry.

5.7.3 In the cockpit

Perhaps the first thing to do after starting the engines if necessary is to select the appropriate TACAN channel if your aircraft is so equipped (the A4F and Lightning both are). For the KC135 (by default used by the Lightning and T37) this is currently "040X", and for the KA6D (used by the A4F) it is "050X". Enter this channel using the relevant dropdown boxes in the "radios" dialogue (from the menus, "equipment/radios" or press control-r).

You should now see the current bearing to the tanker indicated in the nav display of the A4 or the TACAN indicator (green needle) in the Lightning. If the tanker is within range, it will also appear on the radar display of the T38 or Lightning. Take off...

5.7.4 In the Air

Turn to an appropriate heading, guided by the TACAN bearing (you should try a "leading" approach to close in on the tanker) and look for the tanker on the radar or nav. screen. Around 5nm away, you should reduce your speed to around 20kts faster than the tanker (these fly at 280 kts TAS) - a "slow overtake". The KC135 will be visible from about 10nm, the KA6-D, being smaller, just over 1 nm. You should use airbrakes as necessary to keep control of your speed should you find yourself overshooting.

Close to within 50ft of the tanker (don't get too close, or visual artifacts might hide the boom from view). You should see indication in the cockpit that you are receiving fuel - there is a green light in the A4 fuel gauge, and you should see the indicated tank load increase.

Getting to this stage is not necessarily easy - it can take a lot of practice. As with carrier landings, this is not an easy manoeuver in real life either and there are additional complications in the sim; the tanker, being an AI model, is unaffected by the wind and flies TAS (True Air Speed), while you are flying IAS (Indicated Air Speed) and are affected by the environment. As in real life, your aircraft will also steadily increase in weight as the tanks fill which will affect the trim of the aircraft.

(You might find it helpful to use the autothrottle to help control your speed - ctrl-a then Page Up/Down to increase and decrease the set speed.)

Once your tanks are full, or you have taken as much fuel as you wish, close the throttle a little, back away from the tanker and continue your intended flight.

5.7.5 More advanced topics

1. Multiplayer Refuelling

   Refuelling is possible within a MultiPlayer session given certain conditions. A basic flyable KC135 model is available - the pilot of this aircraft should use the callsign "MOBIL1", "MOBIL2" or "MOBIL3". Other numbers are acceptable, but only these three have A-A TACAN channels assigned. These are 060X, 061X and 062X respectively.

   If the receiving aircraft uses a YASim FDM, there are no further complications. Should the receiving aircraft be JSBSim based, the user must make sure that there are no AI tankers in their configuration. This means disabling (commenting out) all refuelling "scenarios" in the relevant aircraft-set.xml and in preferences.xml.

   MP refuelling works in exactly the same way as AI refuelling and is a fun challenge. It is best to ensure that your network connection is as free from interruptions as possible; the MP code does a degree of prediction if there is a "blip" in the stream of packets and this can make close formation flight very difficult or even impossible.

2. Selecting Different Scenarios

   There are several AAR scenarios available in the AI directory:

   (a) refueling_demo.xml has a KC135 circling near KSFO at 3000ft,

   (b) refueling_demo_1.xml the KC135 on a North/South towline at 8000ft and

   (c) refueling_demo_2.xml the KA6D on a similar N/S path but at 8500ft.

   These can be selected by editing preferences.xml (use your operating system's search facility to locate this if you don't know where it is). Open preferences.xml in a text editor (e.g. notepad if on windows) and search for the <ai> </ai> tags. Place a line like

   ```
   <scenario>refueling\_demo</scenario>
   ```

   somewhere within the <ai> tags; you should see other scenarios already there too, perhaps commented out (i.e. with <!– –>).

Part III

Tutorials

Chapter 6

Tutorials

6.1 In-flight Tutorials

FlightGear contains an in-flight tutorial system, where a simulated instructor provides a virtual 'lesson'. To access tutorials, Select Start Tutorial from the Help menu. Tutorials are only available on some aircraft.

The tutorial system works particularly well with the Festival TTS system (described above).

6.1.1 Cessna 172P tutorials

A number of flight tutorials exist for the Cessna 172p, based around Half-Moon Bay airport (KHAF) near San Francisco, provided in the base package. To start the tutorials, select the Cessna 172P aircraft, and a starting airport of KHAF, using the wizard, or the command line:

```
$ fgfs --aircraft=c172p --airport=KHAF
```

When the simulator has loaded, select Start Tutorial from the Help menu. You will then be presented with a list of the tutorials available. Select a tutorial and press Next. A description of the tutorial is displayed. Press Start to start the tutorial.

Each tutorial consists of your instructor giving you a number of directions and observing how you perform them, If you fail to follow the directions, the instructor will provide information on how to correct your deviation. At the end of the tutorial, the number of deviations is shown. The fewer deviations you make, the better you have performed in the tutotial.

For simplicity, run tutorials with AI aircraft turned off from the Options item on the AI/ATC menu. Otherwise, ATC messages may make it difficult to hear your instructor.

To ask your instructor to repeat any instructions., press '+'. You can pause the tutorial at any time using the 'p' key. To stop the tutorial select Stop Tutorial from the Help menu.

6.2 FlightGear Tutorials

A range of ***FlightGear*** tutorials are available from various sources targetted at different users.

Eric Brasseur has written a very good tutorial for people completely new to ***FlightGear***, and flying in general. It also describes many basic principles of flight. Accessible here:

http://www.4p8.com/eric.brasseur/flight_simulator_tutorial.html.

Secondly, there is an excellent tutorial written by David Megginson – being one of the main developers of ***FlightGear*** – on flying a basic airport circuit specifically using ***FlightGear***. This document includes a lot of screen shots, numerical material etc., and is available from

http://www.flightgear.org/Docs/Tutorials/circuit.

A tutorial describing cross-country flight in FlightGear can be found in the following section.

6.3 Other Tutorials

There are many non-***FlightGear*** specific tutorials, many of which are applicable. First, a quite comprehensive manual of this type is the Aeronautical Information Manual, published by the FAA, and being online available at

http://www.faa.gov/ATPubs/AIM/.

This is the Official Guide to Basic Flight Information and ATC Procedures by the FAA. It contains a lot of information on flight rules, flight safety, navigation, and more. If you find this a bit too hard reading, you may prefer the FAA Training Book,

http://avstop.com/AC/FlightTraingHandbook/,

which covers all aspects of flight, beginning with the theory of flight and the working of airplanes, via procedures like takeoff and landing up to emergency situations. This is an ideal reading for those who want to learn some basics on flight but don't (yet) want to spend bucks on getting a costly paper pilot's handbook.

While the handbook mentioned above is an excellent introduction on VFR (visual flight rules), it does not include flying according to IFR (instrument flight rules). However, an excellent introduction into navigation and flight according to Instrument Flight Rules written by Charles Wood can be found at

http://www.navfltsm.addr.com/.

Another comprehensive but yet readable text is John Denker's "See how it flies", available at

http://www.monmouth.com/ jsd/how/htm/title.html.

This is a real online text book, beginning with Bernoulli's principle, drag and power, and the like, with the later chapters covering even advanced aspects of VFR as well as IFR flying

Chapter 7

A Basic Flight Simulator Tutorial

7.1 Foreword

Aviation is about extremes:

- An airplane is quite fragile and flies at huge speeds. Yet it is one of the most secure transport devices.
- A pilot constantly follows rules and procedures. Yet an airplane is a symbol of freedom.
- Once you are trained, flying a little airplane is easy. Yet if a problem occurs, you have to sort of solve a rubicube in a few seconds.
- Many flight tutorials are written with a lot of humor. Yet an attempt to make humor or show your skills with a real aircraft will bring you before a court.

This tutorial is based on the Cessna 172p which is the default airplane on lots of flight simulators and a great airplane:

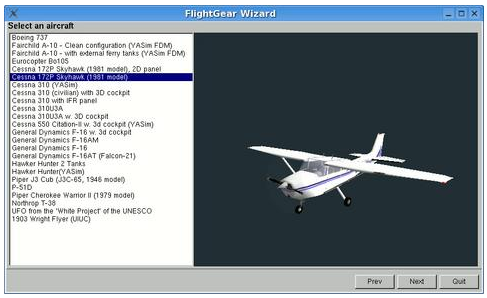
Possibly take a quick look at the following articles. You may feel the need to come back to them later. They contain answers to questions that can arise while reading this tutorial. The first ones show the airplane's main parts and controls:

- http://www.gleim.com/aviation/ltf/howtheyfly.php?PHPSESSID=889ab9792636f430a66e3e5d70f7d346
- http://www.pilotfriend.com/flight_training/new_site/aerodynamics/aircraft%20controls.htm
- http://www.flightgear.org/Docs/getstart/getstart.html
- http://en.wikipedia.org/wiki/Aircraft
- http://en.wikipedia.org/wiki/Flight_controls
- http://en.wikipedia.org/wiki/Airplane_flight_mechanics
- http://en.wikipedia.org/wiki/Aircraft_engine_controls
- http://www.firstflight.com/flt1.html
- http://www.avsim.com/mike/mickey_site/ppilot/ppilot_faq/pp_cessnas.html
- http://www.ig-wilson.com/index.php?f16land
- http://www.navfltsm.addr.com/

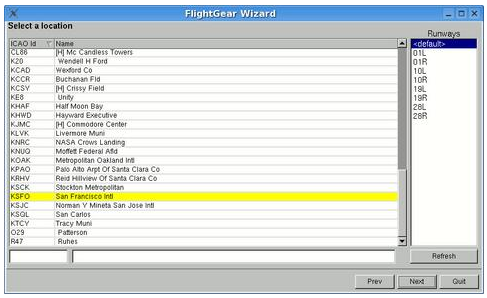
I did my best not to tell too much nonsense. I apologize for the bad habits or reflexes you may get due to this tutorial. It contains for sure some ugly mistakes. Maybe come back in some time, as I make frequent changes to better it.

7.2 Starting Up

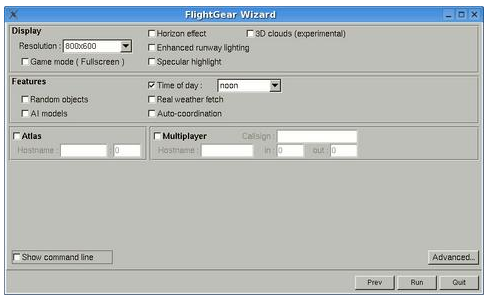
- On Windows, ***FlightGear*** first lets you choose an aircraft and airport. Ask for a Cessna 172p airplane like shown below. To match this tutorial do not choose the 2D panel version. (To be honest, the 2D version is more appropriate to train. Try out both of them and make your choice...) Press the **Next** button to get to the next dialog.



Most airplane airports should fit but in this tutorial I assume you are using FlightGear's traditional airport of San Francisco (KSFO):



In the final dialog maybe best cancel all display options. Ask for a flight at noon (your choice, but best a moment with the Sun still up). (The first time you use ***FlightGear***, pick "noon" in the drop list, even if it is already selected.) Best start with a little display window of 800×600. JLater on you can try to add options and ask for a wider window. Press the **Run** button and the flight simulator window should start:



(If you get problems when you run version 0.9.9 of FlightGear on your Windows system, try installing version 0.9.8. It is available inside the FTP mirrors mentioned at the top of the ***FlightGear*** download page.)

(By a parallel way, volunteers are needed to compile future source code pre-versions of FlightGear and test them on their computer. See

http://www.flightgear.org/Docs/Tutorials/fg_cygwin/fgfs_cygwin.pdf and http://www.flightgear.org/docs.html for documentation.)

(If you get problems under Windows Me; the flight simulator suddenly stuttering, too few images per second... try killing all tasks except Explorer and Systray before you launch FlightGear. (If one of the tasks you kill is an antivirus or such protection software, this is a security risk.) Also, on one Windows Me machine, a ***FlightGear*** of 800×600 yielded good results, while a lower resolution of 640×480 triggered awful FPS drops and stutters (Frames Per Second).)

- On Linux and some other Unix-like systems, it can be ***FlightGear***is installed but you see it nowhere in the menus. Then do one of these:
  - If you have for example the latest ***FlightGear*** installed on a SlackWare Linux system, open a terminal window (also named "console" window) and type the `fgrun` command (and hit the **`Enter`**key). You will get the same cute dialog windows as under Windows (actually I got the snapshots displayed above by using that Slackware version). If you have another version or you get problems configuring the dialog, use next procedure:
  - To start FlightGear open a terminal window and type the `fgfs - timeofday=noon` command (and type the **`Enter`** key). If the FlightGear window you get is too small, close it and maybe restart FlightGear with this command: `fgfs - timeofday=noon -geometry=1024x768`.

If you don't use the `-timeofday=noon` option, it often happens that ***FlightGear*** starts in a night environment. To get a daytime environment, use the **Weather** menu. Choose **Time of day** In the dialog box ask for say **Noon**. Then click **Dismiss**:

(If ***FlightGear*** is available in your KDE or Gnome menu, you can edit the FlightGear launch icon properties and change the simple `fgfs` fgfs command to

something like `-geometry=1024x768` `fgfs - timeofday=noon` or whatever command options you require. You can use any other resolution you want instead of 1024x768. (I try to keep a 4×3 ratio.))

7.3 The basic catastrophe: flying straight

Once ***FlightGear*** is started you see this window content and you hear the sound of an engine:

The airplane engine is on, at low power. The airplane trembles a little, yet it doesn't move.

About the keyboard.

- In this tutorial and in the ***FlightGear*** documentation, a lowercase key letter means you simply hit that key. An uppercase means you shift that key. (The ⇑ **Shift** keys are those two keys with a hollow fat arrow pointing upwards.) In other words: if you are told to type „v", simply hit the **v** key briefly. If you are told to type „V", push a **Shift** key down and keep it pushed down, hit the **v** key, then release the **Shift** key. (In short: V is the same as **Shift-v**.)

- I assume you let the **NumLock**on. That is a little green lamp at the right of your keyboard. Type the **NumLock**key till the lamp is on. Hence I assume you will use the **Home**, **End**, **PgUp** and **PgDown** keys located above the four cursor keys. You can switch **Num Lock** to off and use the keypad for **Home**, **End**, **PageUp**, **PageDown**. Your choice. But this tutorial supposes **Num Lock** is on.

Type key **v**, to see the aircraft from the outside. Type **v** kseveral times, till you get back inside the aircraft. (Typing **V** makes you cycle backwards through the views.):

! Each time before you step inside a real airplane, you have to inspect the airplane all around to check every part of it. You make sure nothing is hampering the moving parts, nothing obstructing the instrument openings...

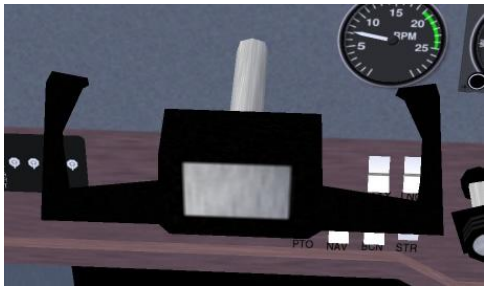
Hold the **Page Up** key down for eight or so lengthy seconds. You hear the engine sound rise.

The airplane starts moving. It drifts to the left, accelerates, rises in the air, banks to the left, falls to the ground, hits it, rises again and crashes.

Maybe you wish to see a replay of this crash: use the **View** menu, choose **Instant Replay**, then click the **Replay** button at the bottom of the dialog window. (Use **v** and **V** to see the airplane from the outside.) The picture below shows the end part of the flight. (Type key **F3** to make a snapshot. Key **F10** to remove the menu bar.)

Close the ***FlightGear*** window and start a new ***FlightGear*** window.

In order to fly straight you need the airplane's control yoke:



Move the yoke by moving the mouse. For this you need to be in mouse yoke mode. Get in that mode by clicking the right mouse button. The mouse cursor becomes a + sign. Move the mouse and see the yoke moving accordingly. Type **v** to see the plane from the outside. Again move the mouse and see the tail elevator moving and the ailerons at booth wings ends. (Type **x** a few times to see the airplane from a closer point of view and better see the ailerons moving up and down. Type **X** to zoom back out. **Ctrl-x** for default zoom. Type **V** to get back inside the plane.)

> ! Each time you start flying a real airplane, you have to visually check that moving the control yoke makes the ailerons and elevator move.

One more click on the right mouse button gets you in mouse view mode. The mouse cursor becomes a ↔. sign. This allows you to move your virtual head all around. Click the left mouse button to center the view back in. A third right-click will bring you again in standard mouse pointer mode.

The right mouse button cycles the mouse through three modes:

- *Normal mode*. This mode allows to click on the menu and on the instrument panel.
- *Yoke mode*. The mouse controls the yoke (+ pointer shape). (The pointer can no longer leave the FlightGear window.)
- *View mode*. The mouse controls the direction you look towards ↔) pointer shape).

Restart the flight simulator, right-click to put the mouse in control yoke mode (+pointer shape) and put the engine throttle on maximum by holding **Page Up** down. Do not try to keep the airplane rolling straight on the runway using the mouse/yoke. Let it drift leftwards. Wait till it rises in the air. Then use the mouse

to try and get the airplane to fly straight. (If you want to control the airplane on the ground see section 7.5.)

You have to prevent the airplane from banking to the left:

Prevent it from banking to the right:

Prevent it from plunging to the ground:

Prevent it from raising its nose in the air (and the stall warning siren from yelling):

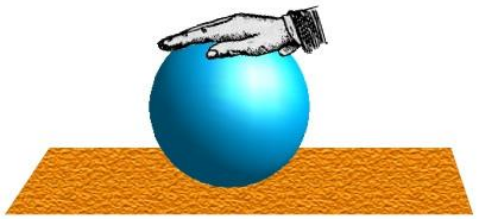
Try to fly more or less straight, with the horizon stable above the airplane nose:

Whatever your skills at video games or maybe even air combat simulators, you won't succeed. The airplane will crash, probably even faster than when you didn't try to control it. This is the moment where most candidates get desperate and abandon trying to fly a simulator or a real aircraft. Just hold tight. Keep trying. Five minutes every day. And read the technical explanations below:

Most awful is this error: when the airplane plunges to the ground, you move the mouse forwards (push the yoke). Because you want to move the airplane's nose upwards. Actually you have to do the opposite: move the mouse backwards (pull the yoke).

Reciprocally, when you want the airplane's nose to dive, you must move the mouse forwards. This can seem odd, but all airplane control yokes are designed that way. You have to get used to it. (Little mouse moves have strong effects on the airplane. Maybe decrease the mouse speed for your first virtual flight attempts.)

A visualization may help: imagine a soccer ball is on your desk and you "glue" your hand on top of it. If you move your hand forwards the ball will roll and your hand will plunge to the desk. If you move your hand backwards the ball will roll back and your hand will now be directed to the ceiling. Your hand is the airplane:



A second error is when you assume the control yoke bank imposes the airplane bank. In other words, you believe if the control yoke is level, the airplane will fly level. This is false. Actually the yoke bank imposes the speed at which the airplane banks. If the airplane is banked 20° to the left and the control yoke is level, the airplane will stay banked at 20° left forever (roughly speaking). If you want the airplane to bank back to level, you have to turn the control yoke slightly to the right (move the mouse slightly rightwards) and keep it slightly to the right for a while. The airplane will turn slowly rightwards. Once it is level with the horizon, put the control yoke level too. Then the airplane will keep level (for a short while).

A third error is: you try to find "the right position" for the yoke/mouse. You try to find the fine tuning that will leave the airplane fly straight. Actually there exists no such ideal yoke position. The airplane is unstable. You constantly have to move the mouse a little bit to correct the airplane's attitude and keep it flying straight. This may seem a stressing nightmare but you will become used to it. Just like with driving a car. After a few months you will even no longer notice you are guiding the airplane to fly straight. (You can use the autopilot to keep the airplane level during long flights.)

An important hint: don't keep your eyes on the airplane instrument panel or on the control yoke drawing. Keep your eyes on the outside scenery and especially the horizon. Check the angle of the horizon and its height above the airplane's white nose. The horizon line and the white airplane engine cover are your main flight instruments. Look at the instrument panel only once in a while.

(While the mouse is in yoke control mode (+ pointer shape), don't move it close to the FlightGear window edges. It's useless and awful things can happen. If you want to get the mouse outside of the window, first go back to standard mouse mode by clicking two times on the right mouse button.)

You can also control the yoke using the four **`keyboard arrow`** keys or the keypad **8**, **2**, **4** and **6** keys.

You may hear beeping sounds while flying around the airport. Those are landing aid signals. Don't pay attention to them, they don't warn for a danger.

You master the thing if, while you are flying straight, the airplane very steadily climbs in the air. Next step is to keep the airplane at more or less constant altitude or make it descend slowly then rise again slowly.

The altimeter instrument is at the middle top of the instrument panel. The long needle shows hundreds of feet, the short needle shows thousands of feet. Hence the altimeter below shows an altitude of 300 feet. That makes roughly 100 meters.

Beware: an altimeter does not automatically show the absolute altitude above sea level. You have to tune that in. See the little black knob on the lower left side of the altimeter. Start ***FlightGear*** and stay on the ground. Click (in normal mouse mode) inside the black knob. A click on the left half makes the altimeter turn back. On the right half the altimeter turns higher. Use that little knob to tune in the altitude. The principle is you use the knob when you are sure about the altitude. If you know you are at 1,100 feet altitude, tune in 1,100 feet on the altimeter. . . (Clicking with the middle mouse button makes the knob turn faster. Type **`Ctrl`**-**`c`** to see the two button halves highlighted.)

Also keep in mind the difference between “altitude above sea level” and “altitude above the ground”. If you fly above Mount Everest at an altitude of 24,000 feet *above sea level*, then you are at 0 feet *above the ground*. (That’s why the HUD displays two altitudes.)

7.4 Basic turning

Once you are able to fly straight, even just approximately, you can begin to learn to turn. The principle is simple:

- When the airplane is banked to the left, it turns to the left.
- When the airplane is banked to the right, it turns to the right.

Don't overbank. 20° is a good bank to get a steady and reliable turn. This it what the turn coordinator is used for. On the picture below the indicator shows the airplane is banked 20° to the right. This is just fine to turn to the right:

Try this out: keep the airplane banked around those 20° for a few minutes and look at the outside. You will see the same ground features appear again and again, every 120 seconds. This shows you need 120 seconds to make a 360° turn (or 60 seconds for a 180°)turn). (This is utterly important when navigating: whatever speed the airplane is flying, if you bank at 20° you always need 60 seconds to make a 180° turn. Whatever the speed or altitude. The bank indicator and the clock are essential navigation instruments.) (Note there seems to be a small error on ***FlightGear***: a 180° takes only 50 seconds instead of 60.)

So, by banking the airplane to the left or to the right, you make it turn to the left or to the right. Keeping the airplane level with the horizon keeps it flying straight.

(The little purple ball in the bottom of the turn indicator shows the sideways forces. If you turn neatly (using the rudder a little bit (see below)), the ball will remain centered. If the ball is pushed say rightwards, this means you the pilot too are pushed rightwards. Like in a car turning to the left. During a neat turn in an airplane, even a strong turn, the passengers never endure a sideways force. They are only pushed a little harder on their seats.)

By experimenting you will notice you easily get fast and spectacular turns by banking the airplane to strong angles and pulling on the yoke. It would be mad to do this with a real airplane if you are a beginner or if you have passengers aboard. Anyway one of the trainings to become a pilot is to make the airplane bank up to 60°.

Each time you start the flight simulator, you have to decide whether you are going to learn flying or just get fun doing mad things. There is nothing bad with the fun. The more fun you make, the better understanding of the aircraft you get. That's one use of a flight simulator and it's good for your security. But you also have to train calm and realistic flying. Either you make a mad flight or you make a serious flight to mimic a real flight. Don't mix these two modes.

7.5 Turning on the ground

The picture below shows the tachometer instrument. It displays how many hundreds of Rotations Per Minute the engine is doing:

Start the flight simulator. Type the **`Page Up`** key a few times, till you get the engine rotation speed to 1,000 RPM (like displayed above). (Typing the **`Page Down`** key decreases the engine speed.)

At roughly 1,000 RPM, the airplane will roll on the runway, but it will not accelerate nor take off.

Type the "**`.`**"key (**`Shift-;`** on Azerty keyboards). The airplane will make a sudden little turn to the right. If you keep the "**`.`**"key down the airplane will halt. When you type the "**`.`**" key, you activate the brake on the right wheel of the airplane. That makes the airplane turn right and halt.

To activate the brake on the left wheel, use the "**`,`**" key.

The "**`,`**" and "**`.`**" keys simulate two brake pedals located at your feet on a real airplane. This way you can control both the speed and turn of the airplane on the ground. (Some airplane, like the Hunter, can turn on the ground only by using this method.)

(For the hackers amongst you who own an Azerty keyboard and want to tune in something more practical than "**`,`**" and "**`Shift-;`**" for the differential brakes: being root, edit file **keyboard.xml** (it is located at `/usr/share/games/FlightGear/data/keyboard.xml` on my computer). Around line number 300 you should find two lines
`<key n="44">` and
`<name>,</name>` and a little below two other lines
`<key n="46">` and
`<name>.</name>`. They are explicitly followed by lines mentioning them as "Left brake" and "Right brake". Change the first two lines to
`<key n="59">` and
`<name>;</name>` and the two further below to
`<key n="58">` and
`<name>:</name>` to get ";" and ":" for the differential brakes. (59 is the ASCII

code of symbol ";" and 58 is the ASCII code of symbol ":".))

The brakes can be very useful when taxiing slowly on the runway. Another (complementary) method exists: you can use the airplane front wheel. In a real airplane this is done by pushing the rudder pedals with your feet. You push with your feet on the side you want to turn towards. In ***FlightGear***two ways exist to control the rudder pedals:

- Using the keypad **0** and **Enter**keys . If you type the keypad **Enter** key say seven times, you will see the airplane firmly turns to the right and stays turning. Type the keypad **0** key seven times to get the airplane back rolling (almost) straight.

- Using the mouse. While the mouse is in yoke control mode (+pointer shape), if you hold the left mouse button down, the mouse controls the rudder instead of the yoke. This feature is absolutely marvelous. Take some time to train it.

Start the simulator, Type **v** or **V** to view the airplane from the outside and keep **x** down a couple of seconds to zoom on the airplane. Look at the front wheel and keep keypad **0** down. Then keep keypad **Enter** down. See the front wheel turn. Click on the right mouse button to get in yoke control mode (+ pointer shape). Keep the left mouse button down to get in rudder control mode and move the mouse to the left and to the right. Note that the rudder, that big vertical control surface at the rear of the plane, moves together with the front wheel.

I tend to control the rudder pedals using the mouse when the front wheel is on the ground and using the keypad **0** and **Enter** keys when the front wheel no more touches the ground. In other words: I keep the left mouse button down when the front wheel touches the ground. This allows for a precise and easy rudder control on the ground. I release the left mouse button when the front wheel no more touches the ground. Then I use the keypad keys **0** and **Enter** to control the rudder.

A drawback of FlightGear is that you don't see the position of the rudder pedals. To see it, two methods are available:

- Ask for the "panel", by typing the **P** key (**Shift**+**p**). Type **P** again to remove the panel. The little white cursor at the bottom of the I-shaped indicator shows the position of the rudder pedals. (That I-shaped indicator is just below the red (**BRAKE** light.) (Note the panel won't appear if the view is not centered.) (You can also choose a plane with a 2D panel when you start the simulator.)

- Ask for the Head-Up Display, by typing the **h** key. The picture below shows the *HUD* rudder indicator. The green arrow is slightly to the right of the green center line. This means the right rudder pedal is slightly pushed in.

(Type **h** several times to toggle between two HUD colors and no HUD. Type **H** to change the HUD color intensity. Type **I** to get a simpler HUD (my favorite) (**i**to get back standard HUD). The sequence of keys I use to get my favorite HUD is **h H I**. The picture below shows this HUD. The uppermost and large scaled green indicator is the compass. Just below it is the horizontal yoke/mouse/ailerons position. The arrow shows the
yoke/ailerons is centered. At the full right of the picture is the engine throttle lever position indicator. The arrow at its bottom shows the throttle is tuned to minimum. At the full left are the trim and vertical yoke/mouse/elevator position indicators (the trim is on the left side, the yoke is on the right side). The short green texts at the top of the picture, left and right from the HUD compass, are the plane GPS position. They are almost unreadable on a standard 800×600 window like below. (Either tune in a black HUD (**H**) or use a larger window. 1200×900 is fine.) The green HUD texts at the bottom of the window, left and right, contain valuable data. (I don't use them in flight. I rather use them during flight replays.))

This is the airspeed indicator, expressed in knots:

A knot is 1.85325 kilometer/hour. So, if you want to have a rough idea of your speed in flight expressed in km/h, multiply the knots displayed by 2. A knot is 1.15115 miles per hour, so very roughly, 1 knot is 1 mph. (Be careful with these roughnesses. Multiplying by 2 instead of 1.85325 makes a difference of 8%. Now, for example: landing at 65 knots instead of 70 knots makes the landing quite different, even when this is only a difference of 8%... And landing at 80 knots, which is only 14% more than 70 knots, can get you into real trouble.) (Note some aircrafts' airspeed indicators display mph instead of knots.)

Note the airspeed indicator displays the speed of the aircraft compared to the surrounding air, not the speed compared to the ground like a car speed indicator does. If the plane is halted on the ground and there is a 10 knot wind blowing towards its face, the airspeed indicator will display 10 knots airspeed, although the plane doesn't move...

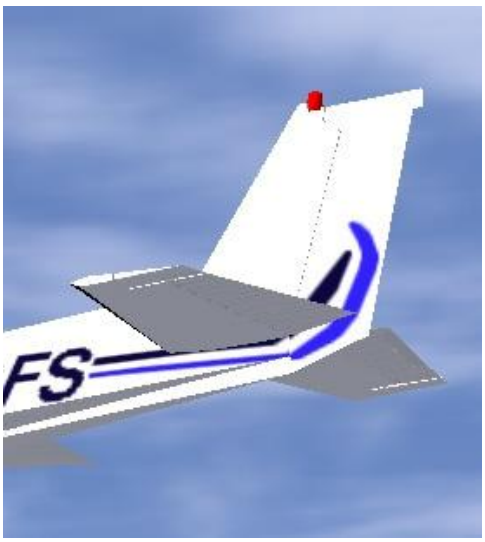
When the airplane rolls over the runway at more than 40 knots, you must prevent the front wheel from touching the ground. During take off, once over 40 knots

you make the front wheel leave the ground by pulling a little bit on the control yoke (on the mouse).

The picture below shows the front wheel slightly lifted. Don't overdo this. Keep the airplane's white nose cover well below the horizon. You just need to lift the plane's nose a little.

The reason why you must raise the front wheel is it is not designed to roll at high speeds. It would shimmy.

Question: if the front wheel no longer touches the runway, how do you steer the airplane? Answer: still using the rudder pedals. Indeed the rudder pedals are linked to the tail rudder, that big vertical moving part at the tail of the plane:



At air speeds above 40 knots, the rudder is adequate to steer the airplane.

The rudder pedals command both the front wheel and the rudder at the airplane's tail. So, just move the rudder pedals...

Note the front wheel and the tail rudder don't make the airplane turn exactly the same way. So when the rudder takes over the front wheel, you must adapt the rudder pedals angle. That means fast typing keypad **`0`** and keypad **`Enter`** (or hold the left mouse button down and tightly control the rudder with the mouse).

Once you trained all this, you are able to keep the airplane straight on the runway when taking off.

An advice: say the airplane is heading too much to the right. You type keypad **0** a few times to make it turn back to the left. Well don't wait till the trajectory is corrected. Type keypad **Enter** a short while before the airplane reaches the direction you wish. Otherwise it will go turning too much to the left. (If you use the mouse, things are much easier and precise.)

So, two methods exist to steer the airplane on the ground: the differential brakes on the side wheels and the rudder pedals. This is essential to aviation: at least two ways to perform each important function. This is called ***redundancy***. If one method fails, you use the other method, even if that second method is not optimal. Sometimes three or even more ways exist to perform a given task.

Don't overdo turning on the ground, especially at high speed. That would make the plane fall sideways and be damaged. Make use of the simulator to try this out (fun mode).

(Why does the airplane drift to the left when it rolls on the ground, making you have to compensate with a little push on the right rudder pedal (about two keypad **Enter** hits)? Main reason is the flow of air produced by the propeller. It blows along the airplane body, but also it turns around the airplane body. The upper part of that slight vortex pushes the vertical tail to the right. That makes the front head to the left.)

You can center all yoke and rudder controls by typing **5** on the keypad. This is a good preflight precaution. Sometimes it can "save your life" in flight. (Note the trim is not centered by keypad key **5** (see below).)

A little problem in flight was (in version 0.9.9 and earlier) the troubles!mouse drifting away from the center of the screen. After a while, you got the yoke centered by placing the mouse quite far from the center of the screen. Two solutions exist:

- The solution I use is to click the right mouse button three times. Cycling through the three mouse modes centers the mouse in yoke mode. Actually the click sequence I use is slightly more complicated: right-click, lift the mouse, left-click, right-click, mouse back down, right-click. This is hectic but it ensures the view keeps centered too. (Note I almost never center the mouse. I don't look at the mouse to know if the yoke is centered. Rather I look at the horizon. That's the way I check the plane is flying OK.)

- Quickly put the mouse in the center of the window, type keypad **5** and get the plane back to stable flight.

Before you type **F3** to make a snapshot, better put the mouse in standard mouse pointer mode. Only then type **F3**, then **Enter** to close the little report window.

Then click the right mouse button to get back to mouse yoke control (+ shaped pointer).

7.6 So, two methods exist to turn in the air?

Indeed. You can use the wing ailerons (steered by the yoke/mouse) or you can use the tail rudder (steered by the rudder pedals / the keypad keys /**0** and **Enter**.

Why these two ways? Because we need redundancy, of course, but especially because they are very complementary:

- When flying close above the ground, you better don't bank the airplane in order to turn. Use the rudder instead. Acting on the rudder pedals allows to make the airplane turn without banking. (If you are close to the ground, this implies you fly at low speed.)
- Reciprocally, when the plane is close above the runway, the two side wheels need to be at the same height above the runway. That means the wings must be level with the horizon. The plane is not allowed to bank. You keep the plane wings level with the horizon by using the yoke/mouse/ailerons. Note this does not need to be perfect. A bank of a few degrees seems harmless.
- In flight, especially at high speed, the rudder is a dirty way to make the airplane turn:

  → It makes the airplane present its flank to the airstream, hence the airplane is braked.

  → The airplane will turn very slowly.

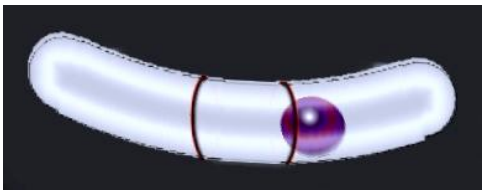
  → You don‘t get a very good control on the turn.

  → At high flight speed the centrifugal force will be disturbing or even dangerous.

  Using the yoke/mouse/ailerons allows for efficient, fast, reliable and comfortable turns.

- When you turn in flight, using the ailerons, you still need the rudder a little bit. You add a little bit of rudder (that is the rudder pedals/the keypad **0** and **Enter**) to the movement. This allows you to perfectly compensate the centrifugal force. You check this visually on the turn coordinator. On the picture below the little ball is pushed rightwards during a strong turn to the right using the ailerons. That means you the pilot endure a rightwards force too. Compensate this by pushing the right rudder pedal (type the keypad **Enter** key a few times). In normal flight you use the rudder to keep the little ball centered. (I don't care for this when flying the simulator. Gentle

turns on the Cessna 172p seem to keep the ball centered without using the rudder.)



- The rudder can be vital when the wings are stalled. Indeed, during a stall the wing ailerons become less effective or even useless. (Note some airplanes can go in a very dangerous stall if you overdo the rudder control at low speed.)

So, you tend to turn by using the ailerons in normal flight and by using the rudder when close above the ground at low speed. Yet one method never completely cancels out the other. You still need the rudder at high altitudes and speeds. Reciprocally you have to use the ailerons a little bit when close to the ground, to keep the wings level with the horizon. (Actually you must use the ailerons even when taxiing slowly on the ground, when there are strong side winds, to prevent the airplane being tilted and blown aside.)

Best never make quick and strong movements with the rudder. On the ground at high speed this can make the airplane tumble aside. In flight at low speed it can cause a very dangerous stall. In flight at high speed it can cause all kinds of aerodynamic and physical discomfort. Try to make slow movements with the rudder. Make slow tunings at a time and take your time to stabilize their consequences. (Only the ailerons allow for nervous movements.)

I recommend you train to turn with the rudder in flight. Fly at a low speed of about 70 knots. Try to keep the altitude stable by increasing and decreasing the engine power. Maybe best a quite low altitude. Use the rudder to get to a target, to maintain a heading, to make turns to a new target... See how the plane yaws. Learn to anticipate rudder control. Don't try to make steep turns. Use the yoke/ailerons to keep the wings level constantly.

7.7 A Bit of Wieheisterology

Wieheisterology, from German "Wie heißt Er" – "What's that name". This section is about gauges, switches and controls of the aircraft. It's like the buttons of a video game control pad: you can play a game with just the arrow buttons but if you want to get the fun out of the game and beat serious opponents you need to learn and train the functions of the other buttons. The same, you can take off and land a basic

airplane with just the engine throttle and the yoke but you need all the controls to make a secure and efficient flight. You need to have at least a basic understanding of the physics behind the controls. In emergency situations you have to understand how the controls work to be able to cope with their deficiencies.

7.7.1 Engine control

An airplane engine is a technological wonder. It is the most powerful, efficient, lightweight and reliable fuel energy plant commonly available.

On the bottom left, below the instrument panel you find the magneto switch/engine starter:

To see the switch, either type **P** to get the schematic instrument panel or type **Shift-x** to zoom out (**x** or **Ctrl-x** to zoom back in).

Move that switch with the **{** and **}** keys (use the **Alt Gr** key on Azerty keyboards).

You probably know the fuel inside a car engine is ignited by electric sparks. A car engine contains an electric magneto to create the electricity for the sparks. An airplane engine contains two such magnetos: the “left” one and the “right” one (redundancy...). When you put the magneto switch on OFF, both magnetos are switched off. Hence the engine can’t run. (Putting the magneto switch on OFF is a way to shut the engine down. Yet you shouldn’t use it because it causes residues to deposit inside the cylinders.) When you put the magneto switch on L you are using the left magneto. On R you are using the right magneto. On BOTH you use both. In flight you have to be on BOTH.

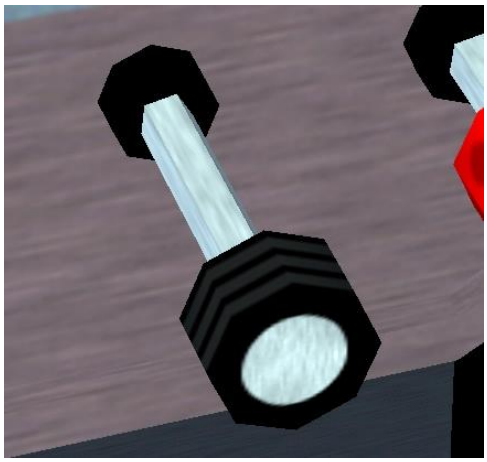
Why do you have the possibility to use the left and right magnetos alone? This can seem useless, since you fly using both. The reason is each time you start the engine in order to fly, you have to verify each magneto *separately*. So you put the magneto switch on L, then on R, slowly. That way you check each of them. If everything is OK, then you put the magneto switch on BOTH. Should one of the two magnetos fail in flight, the other one will keep doing the job. The failure of one magneto is rare, the failure of both together is almost impossible. If during the pre-flight check it appears one of the magnetos fails, ***you have to cancel the flight***.

You surely already started the simulator and typed **{** to shut the engine down. So now you want to start it back on. Type **}** three times in order to put the magneto switch on BOTH. To start the engine press the **Space Bar**. Keep it pressed a few seconds, till the engine is started.

You can also turn the magneto switch and start the engine by clicking left and right of the switch (normal mouse mode). Type **Ctrl-c** to see the two click sides highlighted by yellow rectangles.

If you turn the switch to OFF, the engine noise stops. If you quickly turn the switch back to L, the engine starts again, though you didn't turn the switch to START. The reason is the propeller was still rotating. You should have waited till the propeller came to a halt. Then, placing the switch on L, R or BOTH won't start the engine. (Once the engine is halted, always place the magneto switch to OFF.)

Let's talk about throttle lever:

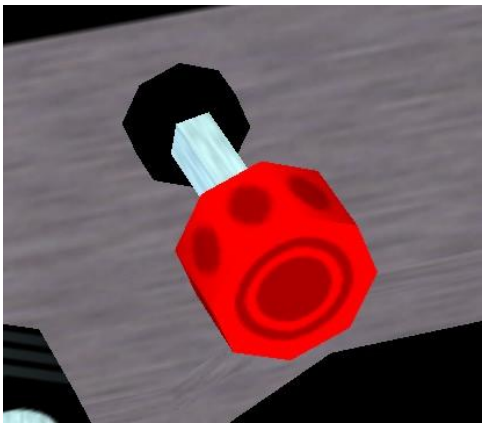


You already know you increase the engine power by pushing that throttle lever in (**Page Up** key). You decrease the power by pulling the lever out (**Page Down** key). You can also click left and right of the lever (middle mouse button for quicker moves, **Ctrl-c** to highlight the left and right halves).

What means "increase the power"? Does it mean you increase the amount of

fuel delivered to the engine? Yes, but this is not enough to fully understand what you are doing. You need to be aware that the engine is also fed with a huge amount of air. The engine's cylinders burn an intimate mixture of fuel and air. Fuel alone wouldn't burn. Only a mixture of fuel and air can detonate and move the engine pistons. So when you push the throttle in, you increase both the fuel and the air fed to the engine.

The amount of air compared to the amount of fuel matters a lot. The proportion of the two has to be tuned closely. This is the purpose of the mixture lever. The picture below displays the mixture lever, far too much pulled out:



When the mixture lever is fully pushed in, you feed the engine with an excess of fuel. When the lever is pulled out completely, there is an excess of air. The correct position is inbetween. Usually quite close to fully pushed in.

When you start the engine and when you take off, you need a fuel-rich mixture. That means the mixture lever pushed in. A fuel-rich mixture allows the engine to start easily. It also makes the engine a little more reliable. The drawback is that a part of the fuel is not burned inside the engine. It is simply spilled away. This makes the engine more polluting, it decreases the energy the engine can deliver and it slowly degrades the engine by causing deposits of residues inside the cylinders.

Once in stable flight, you have to pull the mixture lever a little, to get the optimal mixture. Check this out by doing the following. Start the simulator. Put the parking brakes on with key B (that is **`Shift-b`**). Push the throttle in to its maximum. The engine RPM are now close to the maximum. Slowly pull on the mixture lever (using the mouse in normal pointer mode). You will see the RPM increases a little. You get more power, without increasing the fuel intake. You spill no more fuel in the engine and it pollutes less. If you continue to pull the mixture lever, the RPM will decrease back away, because now there is too much air. The excess of air slows the explosions down inside the cylinders and decreases the explosion temperature, hence the thermodynamic yield decreases. You have

to tune in the optimal mixture. You can check you get the optimal tuning by the fact you get the highest RPM. (Another method is to check the engine exhaust temperature. Roughly, you need to get the highest temperature.)

Question: why a mixture lever? A car contains no mixture lever and drives fine. There are two answers. First is a car is not an optimal device. An airplane is, hence it needs fine tunings. Second and more fundamental answer is a car operates at constant altitude. So the mixture tuning can be tuned in once and forever by a garagist. A plane rises in the air. The higher the altitude, the less dense the air is. Hence the openings or pumps that let the air into the engine have to get wider or pump stronger in order to inject the same weight of air into the cylinders. So when you gain altitude, you have to pull a little on the mixture lever to keep an optimal fuel/air mixture. When you descend back to the ground, you have to push the lever back in. (Actually, if you live at sea level and you move to a new location high in a mountain country, and you take your car with you, you should ask a mechanic to adapt the mixture tuning of your car. Should you drive your car back to sea level, it will drive fine but it will be less powerful and more polluting. . . I suppose modern cars contain some electronics to control this.)

You have to take the mixture lever seriously. It allows you to burn less fuel for the same speed and distance, hence to fly farther away and pollute less. It can also cause serious trouble. Suppose you go flying at high altitude and pull on the mixture lever accordingly. Then you descend back in order to land. But you forget to push the mixture lever back in. The fuel/air mixture will become far too rich in air and the engine will simply halt. You may think the engine is failing and panic, while you only have to push the mixture lever back in. . .

When landing, you have to tune back in a mixture that is a little too rich in fuel. This means pushing the mixture lever in. That way the engine becomes a little more reliable and will be better adapted to a decrease in altitude.

I wrote above that placing the magneto on OFF is not the right way to stop the engine. The right method is to pull the mixture level. First pull the throttle out completely, to get the engine to minimum power and fuel consumption. Then pull the mixture lever, till the engine stops because the mixture contains too much air. This ensures the engine doesn't get poised by spilled fuel residues. Finally, turn the magneto switch to OFF to ensure the engine won't start back accidentally (for example because strong wind makes the propeller turn).

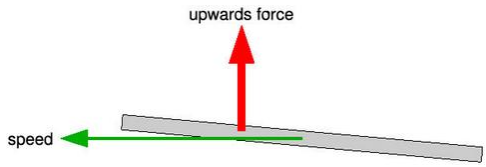
An important warning: you may think the RPM indicator reflects the engine power. Wrong. Two things make the RPM increase: the engine power *and* the airplane speed. To check this, fly to a given altitude then pull the engine power to minimum. Try out diving to the ground then rising back to altitude. You will see the RPM varies strongly. It rises while diving and decreases while rising, together with the plane speed. Though you didn't tune the engine power. One pitfall of this is when you intend to tune the engine power in for landing. Suppose you're flying

fast. You know the ideal RPM for landing is around 1,900 RPM. So you pull the throttle till you get 1,900 RPM. You think you tuned in the appropriate RPM. You think you shouldn't bother any more about it. But now the plane's speed decreases. Hence the RPM decreases. A few minutes later, you get the low flight speed you wanted. You don't see the RPM is now at 1,000. Far too slow. You will either lose altitude or stall. Or both. So, be cautious with the throttle and with the RPM indicator. Either pull on the throttle more steadily or be mentally prepared to push it back in quickly.

7.7.2 Wings and speed

Fly with full engine power. Diving the nose a little makes you lose altitude and raising the nose a little makes you gain altitude. You may think this is quite logical. The plane travels in the direction it is heading; the direction the propeller is heading. This is not the appropriate way to think about it. It would be fine for a rocket, but not for an airplane. A rocket is lifted by its engine, while a plane is lifted by its wings. That's a huge difference.

Get a big rigid square of cardboard, hold it horizontally in your hand with your arm stretched out and make it do fast horizontal movements while rotating your torso. When the cardboard moves flat through the air, it experiences no lift force. If you twist your arm slightly to give the cardboard a slight upward angle, you will feel it tends to lift in the air. There is an upward force acting on the cardboard. That's the way a wing holds a plane in the air. The wings have a slight upward angle and lift the airplane. The more angle you give the cardboard, the more lift force. (Till you give it too steep an angle. Then you will rather feel a brake force. The cardboard is "stalling" (see below).)



- When you pull the yoke, the airplane's nose rises up. Hence the wings travel through the air at a steeper angle. Hence the lift force on the wings is stronger. Hence the plane rises in the air.
- When you push the yoke, the airplane's nose dives. Hence the wings travel through the air with less angle. Hence the lift force on the wings decreases. Hence the plane descends.

What matters is the angle the wings travel through the air. That's the angle of attack.

I wrote above that when the wings travel through the air with no angle, they don't lift. This is false. It would be true if the wings were a flat plate like the cardboard. But they aren't. The wings are a slightly curved airfoil. That makes them create a lift even when traveling through the air at no angle. Actually, even with a little negative angle of attack they still create a lift force. At high speed the airplane flies with the wings slightly angled towards the ground! This is not very important...

The angle at which the wings travel through the air matters. Something else matters too: the speed. Take the cardboard again in your hand. Hold it with a given slight angle and don't change that angle. Check that the faster you move the cardboard, the more upward lift force it experiences.

- When you increase the engine power, the plane increases speed, the lift force on the wings increases and the plane gains altitude.
- When you decrease the engine power, the plane decreases speed, the lift force on the wings decreases and the plane loses altitude.

To make things a little more complicated: when rising in the air, the airplane tends to lose speed. When descending, it tends to gain speed.

That's all a matter of compromises. If you want to fly at a constant altitude and at a given speed, you will have to tune both the engine power and the yoke/elevator (or better: the trim (see below)), till you get what you want. If you want to descend yet keep the same speed, you have to push the yoke a little and decrease the engine power. And so on. You constantly have to act both on the engine power and on the yoke. (During a normal flight one doesn't make things that complicated. Simply tune in a comfortable engine power level then forget about it and rely on the yoke and trim for the altitude.)

A very interesting exercise you can perform with the simulator is to fly straight with full engine power. Get maximum speed while keeping in horizontal flight. Then you decrease the engine power to minimum. And you pull steadily on the yoke to keep the plane at constant altitude. The plane slows down steadily, meanwhile you pull more and more on the yoke. Since the speed decreases the lift of the wings would decrease, but you compensate the loss of speed by increasing the wing angle of attack. (This proves the plane does not necessarily travel in the direction its nose is heading. In this experiment we make the nose rise in order to stay at constant altitude.) Once the plane is severely slowed down, and the nose is strongly heading upwards, you may hear a siren yell. That's the stall warning (see below). The angle of attack of the wings is too strong. The wings are now braking

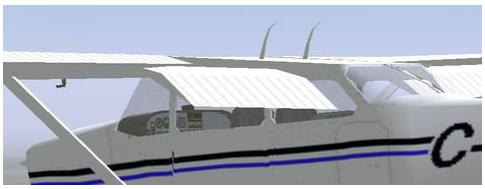
the airplane instead of lifting it. The plane quickly loses altitude. Whatever you pull on the yoke, you're falling. The only thing you can do is push the yoke forwards, make the nose dive, gain speed and glide towards the ground. Possibly push the engine throttle back in to full power.

Question: is it better to control the airplane's speed and altitude with the yoke or with the throttle? Answer: it depends on what exactly you intent to do and on the situation you are in. In normal flight, as said above, you tend to set a comfortable engine power level, forget about it and rely on the yoke and trim. During take off and landing the procedures are quite strict about the use of yoke and throttle. You do the opposite: control the speed with the yoke and trim, control the altitude and descent speed with the engine throttle. That will be discussed in sections below.

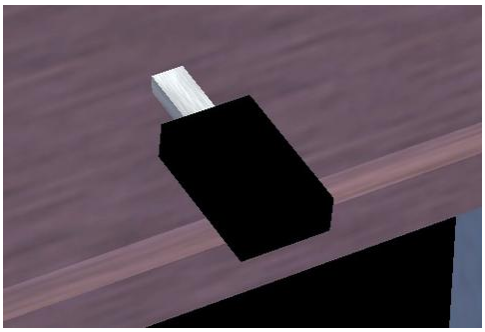
For long range flights, to spare fuel costs or for ecological concerns, one can fly a plane in "economy mode". The you try to tune the wings angle of attack so they brake the less, you try to tune the engine, you try to find the most appropriate speed and altitude... Its all a matter of compromises. You will often fly slower than the usual cruise speed.

7.7.3 The flaps

The flaps are situated at the rear of the wings, aside the plane's body:



Deploy the flaps and pull them back in by using the flaps control lever:



You can either click on it with the mouse or use the **[** and **]** keys. Key **[** to retract the flaps one step, **]** to deploy them one step. Type **v** to view the plane from

the outside and try out **[** and **]**. (On the schematic instrument panel the flaps lever is located at the lower right.)

There are four flaps steps:

- No flaps. For normal flight.
- One flaps step. For take off, when you want to gain altitude while flying slowly. Or during runway approach, while flying at constant altitude.
- Two flaps steps. To brake the plane, in order to lose altitude quickly, for example when you dive towards the runway to land.
- Three flaps steps. To lose altitude even more quickly.

There is a security risk. Do not deploy one flaps step above 110 knots. Do not deploy two or three flaps steps above 85 knots.

The flaps brake the plane at high speed. This is one more reason not to forget to pull the flaps back in once you fly above 85 or 110 knots.

My favorite way to know the flaps position is to type **Shift-right arrow**. Then quickly **Shift-up arrow** to get back to front view. Another method I use is to make sure the flaps are fully retracted by quickly typing [several times. Then type] the exact amount of times needed.

The role of one flap step is to increase the wing lift. The wing lifts more at a given speed. Hence you will get in the air a little sooner during take off. It also has the effect to make the plane fly with the nose a little more downward. This is handy: it allows to keep an eye on the runway while rising in the air. It allows a better view on the runway during landing.

The role of two or three flaps steps is to brake the plane. This is mandatory when landing, because the airplane glides very well. If you cut down the engine power completely, sure the plane will descend, yet too slowly. You need to deploy two or three flaps steps in order to brake and really descend towards the ground.

The fact that the flaps brake during landing makes you need more engine power during the landing. This can seem odd. Why not simply throttle the engine down to minimum and use less flaps steps? The answer is it's better to have a strongly braking plane and lots of engine power, because then the plane reacts faster to your commands. Should the engine fail, then just retract flaps as needed...

Trying to take off with two or three flaps is a bad idea. This can sound fun, but beware: suppose you deployed one flaps to take off. Yet you forgot to pull the flaps back in. Later on you encounter a emergency situation and you need to gain altitude very fast. You deploy one flaps step. Actually you add one flaps step to the flaps step already out. So now you have two flaps steps. Hence the flaps are braking and you fail to gain altitude... Whenever you feel the plane is behaving

really odd and seems unable to rise in the air, or even keeps falling whatever your efforts and the engine power, think maybe you deployed more than one flaps steps.

Redundancy... What can you do if the flaps don't deploy and you really need to brake? Answer: slowly push the rudder pedals on one side. This will make the plane present its flank to the air stream and brake. Compensate the turning by using the ailerons (yoke). This is not a very comfortable way to brake and you should train it before using it close to the ground. (I tried to use both the full flaps, the rudder to an extreme and the throttle to minimum. You really loose altitude very quickly...)

7.7.4 The stall

A stall is an emergency situation, at whatever altitude. It means the plane is flying too slowly hence the wings travel through the air at too strong an angle. The wings suddenly start braking the plane instead of lifting it. It is especially dangerous when close above the ground. It is dangerous even at high altitude because you lose part of your control over the plane.

During a normal flight, a stall should never occur. As a pilot you have to constantly keep the plane well above stall speed. Once the stall siren yells, it means things already have gone very bad.

Some little airplane like the Piper Cub are designed to land using a near stall. Planes like the Cessna 172 are designed to make stalls less likely to occur and less deadly when they occur. That's for example one reason why the wing extremities are square. The Cessna is still controllable during a stall and a simple stall and fast descent to the ground should not kill the passengers. (Wind turbulences or a strong bank can make things go worse...)

A stall can make some airplanes go into a deadly spin. Fly for example the F-16 Falcon to some altitude, throttle the engine down to minimum and pull steadily on the yoke to keep the same altitude while decelerating... One problem with the legendary WWII fighter plane Spitfire was during too tight turns the inside wing would suddenly stall completely but not the outside wing.

What can you do during a stall? The procedure can be very different on different planes. You should not trust this tutorial, especially not for such a serious matter. Anyway:

- On little planes like the Cessna 172p or the Piper Cub you keep all controls on the airplane: elevator, rudder and both ailerons. Keep in mind the ailerons are on the wings and those wings are stalling. Think of using the rudder to turn. If you are very close to the ground, simply let the plane fall on the ground and keep it on the ground. Just try to make the best possible fall. If you are high in the air, push the yoke to dive the nose and gain speed. Think

of retracting the flaps at one step. If possible, immediately add as much engine power as you can. Till you are out of danger.

- On the F-16, the ailerons loose all control during a stall. Actually that's the first sign of a stall: the ailerons act no more and the plane banks loosily. You only keep control with the rudder and the elevator. If the stall just begins, dive the nose and increase engine power. If the stall degenerated in a spin, engine power and dive won't solve the problem. Only the rudder will help to slow the spin down. Push the rudder to the other extreme of the spin direction. Push the yoke to help. Decrease engine power to minimum to get out of the spin more easily. Once the spin is slow enough, a dive and engine power will help. (If you put full engine power during the spin, you will loose less altitude I believe, but the spin's end will be difficult to control. React quickly to center the rudder again once the spin nearly stops, otherwise you will immediately go spinning in the other direction.) At low altitude you probably won't have the time to do all this. Note a spin with the plane belly up can happen too.

Stall-elegant airplanes like the Piper J3 Cub and the Cessna 172 tend to have roughly rectangular wings. While stall-ugly airplanes like the F-16 Falcon and the Cessna Citation II tend to have trapezoidal wings. The advantage of the trapezoidal wings is they have a better aerodynamic yield. They allow to fly more distance with a same quantity of fuel. The ends of the rectangular wings engender strong turbulences. Those turbulences brake the plane but also they keep the air flowing correctly once the plane stalls...

When you learn to fly a virtual plane, making it stall is a very good exercise:

- Fly at constant altitude with no engine power, till the stall begins. Then try to control the plane while it stalls and descends to the ground. Keep the yoke pulled to the maximum. Keep the plane in a steady attitude, the wings parallel with the horizon. Try to change direction. Experiment with the flaps. Note one flaps step decreases the stall speed. Two or more flaps steps don't change the speed much. Then end the stall by pushing the yoke and restoring engine power.

- Raise the nose in the air and bring the plane to a halt like a stone thrown upwards. Then try to get the plane back to normal flight.

Try to perform the exercices above with different airplanes. You will notice how elegant the Cessna 172p behaves. First time I tried a stall with the virtual Cessna Citation II, I was at 1,000 feet altitude, which is supposed to be safe. The plane suddenly fell from the sky like a tumbling stone. I was not able to stabilize the plane and it crashed. I was really frightened by that airplane. On second attempt

I managed to stabilize before the plane hit the ground. Anyway, from now on I won't fly a Cessna 172p and a Cessna Citation II with the same mood.

If you fly an unknown virtual airplane and wish to know the landing speed, a rule of thumb is you find out the stall speed by experimenting. Then you multiply that speed by a factor of 1.2 or more. (A friend who is Aerospace Engineer told me 1.2.) The stall speed of the Cessna 172p is 40 knots yet its imposed landing speed is 70 knots (minimum 65 knots). That makes a factor of 1.75...I made an experiment landing the virtual Cessna 172p at 50 knots. It virtually falls to the ground, at close to -1000 feet/minutes vertical speed. This seems very hard for the landing gear. Next, while approaching at 50 knots with the Cessna 172p, the runway and most of the ground are completely hidden. This obviously tells a higher speed is mandatory. I would recommend following rules to find a correct landing speed. It must be the lowest possible speed that satisfies all these conditions:

- Be more than $1,2\times$ the stall speed.
- Be high enough to allow a vertical speed of no more than -500 feet/minute (all or most flaps steps deployed).
- On most modern planes, the landing speed must be high enough to allow you to see the runway while approaching at low speed and constant altitude.

On big jet airliners the flaps make a lot of difference. Bear that in mind when you try to find the stall speed. Make the experiment with the flaps deployed, as they will be deployed during the landing.

The load of the airplane also changes the stall speed a lot, and therefore the landing speed. You land a fully loaded airplane at a higher speed than an empty one.

(Once you get used to landing different airplanes, you get a feeling for the landing speed of an unknown airplane. You just feel the airplane "wants" to land at that speed. I suppose this is because the airplane was designed to land at that speed.)

> ! In a real airplane the sounds and vibrations tell a lot about the state of the airplane. When all vibrations stop, this means you are going to stall. Then push the yoke to get speed.

7.7.5 The trim

The trim is that dark big vertical wheel with gray dots located at the middle below the instrument panel:

On ***FlightGear***, the keys **Home** and **End** are used for the trim. The key **Home** rolls the wheel upwards while the key **End** rolls the wheel downwards. You can also click on the upper or lower half of the trim wheel (**Ctrl**-**c** for a yellow highlight). Possibly look at the plane from the outside (**v** or **V** and **x**) and move the trim while looking at the elevator.

In first approximation, the trim does the same as the yoke: it acts on the elevator. Turning the trim wheel downwards is the same as pulling on the yoke. Yet there is a key difference between the trim and a real yoke. If you tune the trim, it keeps that tuning. While if you pull or push on the yoke, it goes back to neutral once you release it.

Once in flight, you would keep the mouse/yoke at a given forward (or backward) position. That position is optimal to keep the plane at a roughly steady altitude. In a real airplane, this means you would constantly keep pushing (or pulling) on the yoke. That would be quite uncomfortable. This is where the trim falls in. You tune the trim to impose a default elevator angle. Then you no longer have to push or pull the yoke constantly. In other words: make a global rough tuning with the trim and occasional fast tunings with the yoke/mouse.

The trim is an important control. I tend to forget it, for two reasons. First is the mouse makes the trim virtually useless. This is quite unnatural of course. People with a force-feedback joystick/yoke will feel the need for the trim, as well as people flying real airplanes. Second is the trim didn't operate on the particular version of FlightGear I was using until recently...

During take off the trim must be neutral. You have to check the trim is centered before every take off. Also if you abort a landing and start rising back to altitude, put the trim to neutral. Otherwise the plane may buck.

During landing, while flying at a constant speed of 70 knots and a constant altitude of 500 feet, the same applies as for a steady flight: try to get the yoke/mouse/elevator towards neutral position by tuning the trim. On the Cessna 172p this means trim on neutral (except when the plane is loaded). On the Cherokee Warrior II this means the trim a little "pulled".

During the final dive, some people seem to let the trim as it is and use the yoke, others make the dive using the trim and don't use the yoke/elevator. I don't know which is best. I use the yoke.

To know the trim position, use the HUD (**h**, **H** and **I**) or the I-shaped indicator on the schematic instrument panel (**P**).

The trim movement is very slow. Be patient.

Lots of modern airplanes have a remote control for the trim: a little switch on the yoke, that you can manipulate easily with your fingers. So you don't have to duck to roll the big wheel.

7.7.6 What direction am I flying?

Four basic methods exist to know what direction you are flying:

- Look through the windows. Try to learn and recognize all sorts of ground features, like hills, bridges, cities, forests... The Sun and the Moon are essential features, but clouds can cover them and they move through the sky. Looking through the windows can be quite hectic on a flight simulator. You only have a narrow view on the virtual outside world. Using two more displays, placed left and right of the main one, will help. Yet this is expensive and not mandatory. Several ways exist to allow you to pan your virtual head inside the airplane:

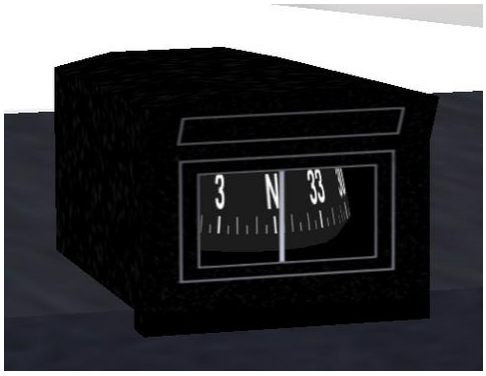
  → Use **Shift** and the four **arrow keys** to look frontwards, backwards, leftwards and rightwards.

  → Use **Shift** and the keypad keys to look in the four directions mentioned above and in four diagonal directions in-between

  → Put the mouse in pan mode (right button, ↔cursor look). This allows you to look in just every direction, including towards the sky and towards the ground. This method is great while the autopilot is on. It is a little dangerous otherwise, because the plane will bank or fall while you're looking all around. Click the left mouse button to quickly get back to the default forwards vision. Hint: if you click the left mouse button to center the vision back, by the time you click the right mouse button to go out of mouse look mode you will already have panned a

few degrees away from the forward view. This is not a serious problem, except for the fact it prevents the instrument panel to appear when typing **P**. A solution is to lift the mouse before you click the left button. Then click the right button. Then let the mouse back down. (While the autopilot is on and you are looking all around, use the **x**, **X** and **Ctrl-x** keys to zoom in and out. Use the **z**, **Z** and **Ctrl-z** keys to dissolve the mist outside.)

- The compass (picture below). This is the indicator located above the instrument panel. The compass is a very simple and classical, yet not perfectly reliable instrument. When flying over some places, magnetic perturbations on the ground can make the compass tell nonsense. Also, the compass never shows the real direction of the North, East or South. Rather it shows a direction a few degrees aside from the real direction (depending on the country you are in). Close to the poles the error of the compass becomes really strong.



- The directional gyro (picture below) or "heading indicator". The gyro is started before take off and keeps its initial heading for hours. It simply tells you how many degrees you turned to the left or to the right. You are supposed to tune in the right direction of the North Pole before you take off, using the knob at the bottom left of the instrument (normal mouse pointer mode, click left or right half of the knob, middle mouse button to move faster, **Ctrl-c** to highlight halves). (The red knob, bottom right, is used to tell the autopilot what direction you wish to fly (**HDG** = "heading").

- The clock. If you make steady turns, at the angle proposed by the turn indicator, a 180° turn takes 60 seconds whatever the flight speed (yet it is 50 seconds on FlightGear...).

> ! During a real flight in a real airplane, you are supposed to cross-check all direction indicators once in a while.

Memorize the directions: North is 0°, East is 90°, South is 180° and West is 270°.

7.8 Let's Fly

7.8.1 A realistic take off

By now I assume you are able to keep the airplane on the runway while taking off (rudder) and you're able to fly straight, descend peacefully, gain altitude steadily, make gentle turns (yoke)... No need you perform this all perfectly. Yet a basic and approximate control of the airplane has been acquired.

Rules during take off:

- You are not allowed to keep the front wheel on the ground above 40 knots. It would shimmy.
- When close to the ground (I don't know the exact limit) you have to keep the two rear wheels at the same height above the runway. The reason is any moment you will or may touch the ground. You need to touch with both two rear wheels together. That means you need to keep the wings level with the horizon. Hence you cannot make use of the yoke/mouse/ailerons to turn. Instead you use the rudder pedals to turn. (Since you fly around 70 knots,

this yields not too much sideways force problems.) The yoke/mouse/ailerons are used to keep the wings level with the horizon.

- You are not allowed to fly lower than 500 feet above the ground. The sole exception is in the axis of the runway, during take off and during landing. (While flying over cities you are not allowed to fly lower than 1,000 feet above the ground.)
- When lower than 500 feet above the ground, you are not allowed to fly *slower* than 70 knots speed. That's because a blow of wind from the rear can occur any moment. You need to fly fast enough so that such wind blows won't make the plane stall and fall to the ground.
- When lower than 500 feet (above the ground), you are not allowed to fly *much faster* than 70 knots. You wouldn't be able to make maneuvers quick enough. You would be more destructive if you hit something. Besides, 70 knots is a nearly optimal speed to gain altitude and your sole acceptable purpose while lower than 500 feet is to gain altitude...
- While taking off, you must stay aligned with the runway. Indeed that's the sole place you are allowed to fly below 500 feet. (If you take off from a long runway like KSFO, this also allows to land back safely and quickly should an emergency occur. (Above a short runway, you cannot simply dive and get back on the runway, because it is too short. You need to turn and circuit to make a regular landing. For this you need to have enough engine power or to be at least at 500 feet above the ground. Otherwise, quickly find a place where you can make an emergency landing.))

So, you need to take off and rise in the air at a steady speed of around 75 knots.

Problem: since the front wheel is slightly lifted and the flaps are one step deployed, the plane will rise from the ground already at 55 knots. That's well below the desired flight speed of 75 knots. What to do then? Answer: as soon the two rear wheels lift from the ground, push the yoke forwards a little. Keep the plane close above the ground. (The aim of this is: should a wind blow from the rear occur, the plane will fall from only a few feet hight.) Keep it close above the ground while accelerating, till a speed of about 70 knots is reached. Then switch to the opposite mode: now you must pull on the yoke to prevent the plane from going above 75 knots. Force the plane to rise in the air, so it doesn't gain speed. Keep in control. If the speed goes below 75 knots, push a little on the yoke. If it rises above 75 knots, pull a little on the yoke. Till you reach 500 feet above the ground.

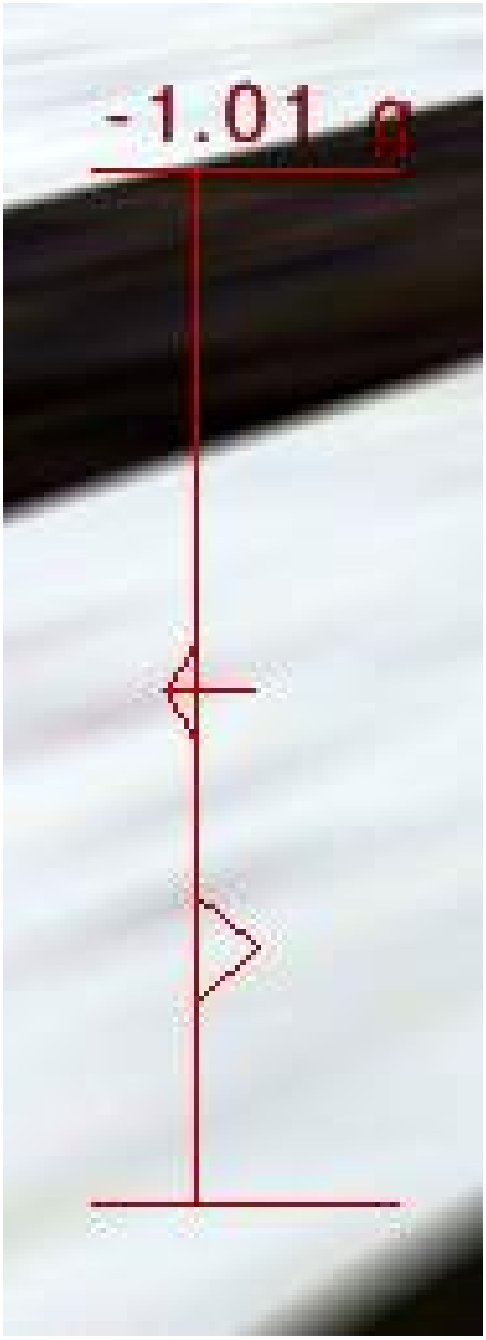
This is the procedure I use to take off. I assume you just started ***FlightGear***; the airplane is at the start of the runway and the engine is turning at minimum power:

1. Get a HUD (**`h`**, **`H`**, **`i`** and **`I`**) or the schematic instrument panel with the I indicator (2D panel aircraft from start or key **`P`**).

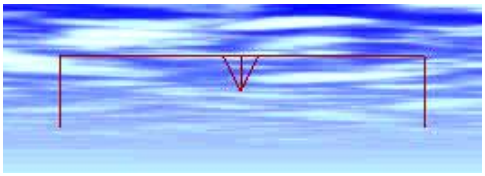
2. Deploy one step of flaps (**]**).

3. Get in mouse yoke mode (+ pointer shape) by clicking on the right mouse button.

4. Pull the yoke/mouse/elevator to 1/2 the total way:



5. Ensure the yoke/mouse/ailerons is centered:



6. Push the left mouse button down and keep it down, so the mouse gets in rudder control mode. (If you don't want to use the mouse to control the rudder, use the keypad **0** and **Enter** keys.) (Before you push the left mouse button, ensure the yoke/elevator is pulled 1/2 like asked above and the yoke/ailerons is centered.)

7. Keep the **Page Up** key down till the engine roars at its maximum power.

8. The airplane is now accelerating. Move the rudder/mouse to the left and to the right to keep aligned with the runway (the left button is pressed). You need to keep in the middle of the runway but this does not need to be very precise. More important is your path is parallel with the runway middle line and stable.

9. Because the yoke/elevator is pulled $\frac{1}{2}$ in, around 40 knots the nose will rise up. Immediately release the left mouse button, to get back in yoke control mode. Immediately push the yoke/mouse a little bit, to keep the engine cover below the horizon. You just need to let the front wheel rise a little bit above the runway. Let the rudder keep its angle (probably slightly turned to the right; two keypad **Enter** hits from the center position). From now on keep the left mouse button released, to stay in yoke control mode. Use the keypad **0** and **Enter** keys to control the rudder. (You can also make short presses on the left mouse button to make little rudder adjustments. I prefer using the keys.)

10. The airplane soon leaves the ground. The two rear wheels no longer touch the runway. Push the mouse a little, to prevent the airplane from rising in the air. Keep it flying close above the runway and aligned with it. (Do not try to stick really close to the ground. This would be dangerous. Let the plane rise a little bit. Just do not favor the rising.)

11. Use the yoke/ailerons/mouse to keep the wings level with the horizon. Use the rudder/keypad **0** and **Enter** to turn (needs training). Optimal rudder position seems to be slightly right from neutral; two keypad **Enter** hits.

12. Once the airspeed reaches 70 knots, pull on the yoke/mouse a little bit. Now the airplane firmly rises in the air. If the speed gets below 75 knots, push the yoke to force the airplane to rise slower and gain airspeed. If the speed rises

above 75, pull the yoke to rise faster and decrease the airspeed. There is no need to be very precise. Try to keep a stable speed. Just avoid to go below 70 knots and above 80 knots.

13. Don't keep your eyes too much on the speed indicator while you are rising above the runway. Rather look at the horizon and at the engine cover. The top of the engine cover should roughly match with the horizon line:

14. If you want to check the position of the runway but you can't see it because it is hidden by the engine cover, push the yoke/mouse a short while to make the nose dive a little bit for a second. This only works for long runways. Another trick is to look for a building, a hill or something far in front of the runway, on the horizon. Keep aiming at that object while rising in the air. Keep the engine cover a little below the horizon line, so the object you aim at stays visible.

15. Once you reach 500 feet, retract the flaps (**[**) and push the yoke a little. Center the rudder (slowly, one step at a time). You are now allowed to gain speed or go on climbing (your choice, or the control tower's). Decrease the engine power a little so the RPM needle gets in the green zone (**Page Down**). Turn calmly towards your intended flight direction. Use your time to optimize the mixture. You're in flight.

500 feet above the ground is the minimum flight altitude above open land. Above a city the minimum altitude is 1,000 feet.

If you take off from KSFO heading to the West, you have city areas in front of you and left of you. So, once you reach 500 feet above the ground, best turn to the right.

Don't forget to center the rudder. If the rudder is pushed to one side, this will brake the plane. It makes the plane move sideways through the air, with its flank aerobraking.

Don't forget to retract the flaps.

! During a real take off you must keep in touch with the control tower. You also have to constantly look in all directions to check no other airplane is coming in your direction.

An aviation classic is the ground effect. It's the fact a wing lifts better when close above to the ground. That too makes the wheels leave the ground at quite a low speed, a speed at which the airplane cannot really fly. While you are accelerating a few feet above the runway, you are in ground effect. If you know about it, ground effect is an advantage because it makes flying close above the ground more secure. The airplane behaves a tiny little bit like a hovercraft. If you are not aware of the ground effect, it can cause problems. For example it can make you think the airplane has enough speed to rise in the air, while it has not.

! During a real take off, if the engine halts below 500 feet, you are not allowed to turn and try to glide and land back on the runway. You only have enough height to try to turn and land back if you are above 500 feet when the engine halts.

! Before a real take off you have to go through check-lists. A checklists makes you verify, tune and tighten a list of items. You have to follow a long checklist before you enter the runway and a short checklist before you accelerate to take off.

This is the checklist I follow when I take of the virtual Cessna 172p on ***FlightGear***. It is very short compared to a real checklist. Anyway I know I can go into (moderate) trouble if I don't follow it. I had to build up the discipline to follow it carefully each time:

- Check the wind direction
- Deploy one step of flaps.
- Click the right mouse button and ensure the mouse is in yoke mode (+).
- Put on a HUD (**h**, **H**, **i**, **I**) or the schematic instrument panel (**P**) in order to know the controls positions.
- Pull the yoke/mouse to $\frac{1}{2}$ the pull path.
- Check the yoke/ailerons are centered.
- Keep the left mouse button down and check the rudder is centered or slightly to the right.
- Keep the **Page Up** button down to start accelerating, till the engine RPM is maximum.

7.8.2 Landing

When I was a boy, I had a simple yet fairly good flight simulator on my Sinclair ZX Spectrum home computer. I could do everything with it, except landing. I always crashed the plane, or reached the end of the runway before stopping. One day a real pilot saw me trying to land. He had never seen a flight simulator, but he had no problem to recognize each flight instrument and ground feature on the screen. He told me what to do. Decrease engine power, increase engine power, push the nose down, pull the nose up, turn a little left, turn a little right, get the flaps out... We made a perfect landing on the second attempt.

Just like for take off, landing is partly a procedure, partly rules you have to stick to. You have to adapt constantly.

Same basic rules apply as for take off, yet in reverse order:

- Stay at 70 knots once below 500 feet. Descend towards the runway while keeping at 70 knots.
- After the final rounding (see below), stay close above the runway while decreasing speed from the 70 knots flight speed down to the roughly 55 knots landing speed.
- Touch the runway with the two main wheels. Keep the front wheel from the ground till the speed is below 40 knots.

(If you know what you are doing you are allowed to use a speed a little below 70 knots: 65 knots.)

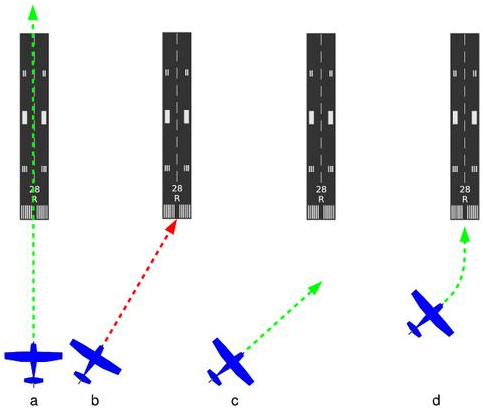
Following rules are essential during the whole procedure of landing:

- Tune the speed using the yoke/mouse/elevator: push the yoke if you are flying below 70 knots, pull the yoke if you are flying above 70 knots. No matter this makes you gain or lose altitude (except when this causes a danger of course).
- Tune the altitude using the engine throttle. Add power if you are too low, retract power if you are too high.
- Once approaching the ground, use the yoke/ailerons to keep the wings level with the horizon. Turn using the rudder.
- Don't shut the engine down. Only shut the engine down when the airplane is completely halted on the ground. There are two reasons for this:
  - → Any moment you may need full engine power to rise back in the air.
  - → Engine thrust enables you to make more precise landings. For example if you land on a very short runway, you need that precision.

The reason why the yoke/elevator is used to tune the speed is this method allows for fast reactions and fine tuning. It is more important to tune the speed closely than the altitude.

If you are both a little too high and a little too slow, simply push the yoke a little and both problems will be solved together. No need to use the throttle. Use your mind...

You have to get aligned with the runway. That means your flight direction has to match the middle line of the runway (drawing (a) below). In order to arrive at this, don't aim at the start of the runway (b). Rather aim at a fictitious point well ahead of the runway (c). And begin to turn gently towards the runway well before you reach that fictitious point (d). Note the turns and bankings you make for these flight corrections are often very soft. You wouldn't even notice them on the turn coordinator. This is one example where you better rely on the outside horizon line than on the inside flight instruments.



Try to get aligned with the runway as soon as possible. Constantly apply the alignment procedure. The closer you come to the runway, the better the alignment should become.

My favorite landing procedure for the Cessna 172p is roughly this one:

1. Far from the runway, yet already heading towards it, start decreasing the speed and let the plane descend towards 500 feet.

2. Check the rudder is neutral. Otherwise the plane will be braking and more engine power is needed. (Type keypad keys **0** and **Enter** to center the rudder if needed.) If you make corrections using the rudder, keep in mind you may need a little more engine power.

3. Once the speed is below 100 knots, deploy one flaps step (**]**).

4. Once an altitude of 500 feet is reached, keep that altitude. Once a flight speed of 70 knots is reached, keep that speed. (If in doubt, keep above 500 feet.) The exact altitude doesn't matter much provided it is stable. But stick to 70 knots.

5. Be firm with the flight speed. Keep a tight and quick control on the yoke/mouse to keep 70 knots. If the speed is lower than 70 knots, push the yoke to gain speed (no matter you lose altitude). If you are above 70 knots, pull the yoke to lose speed (again, no matter this makes you gain a little altitude). Don't panic if the speed rises to 75 knots or decreases to 65 knots. But keep in mind you can really get in trouble if you approach a short runway at 80 knots. I manage to keep the speed between say 68 and 72 knots.

6. Tune the trim to get the average position of the yoke/elevator centered. This is not mandatory on the simulator, yet that way you better mimic piloting a real airplane. On the Cessna 172p (with no load) this means trim on neutral.

7. Be firm with the flight speed. Keep a tight and quick control on the yoke/mouse to keep 70 knots. If the speed is lower than 70 knots, push the yoke to gain speed (no matter you lose altitude). If you are above 70 knots, pull the yoke to lose speed (again, no matter this makes you gain a little altitude). Don't panic if the speed rises to 75 knots or decreases to 65 knots. But keep in mind you can really get in trouble if you approach a short runway at 80 knots. I manage to keep the speed between say 68 and 72 knots.

8. Fly at constant speed and steady altitude towards the runway. 70 and 500. Keep trying to align with the runway. You will **never** be perfectly aligned. You have to go on aligning till the airplane halts on the runway.

9. Now you are at a low flight speed of 70 knots, no more use the ailerons/yoke to turn. Instead use the ailerons to keep the wings horizontal. Turn using the rudder (Keypad keys **`0`** and **`Enter`**). The rudder can seem an odd device for this purpose yet you will get used to it. Move the rudder only a few key hits to the right or to the left. Be patient. Make one key hit at a time and allow the airplane to stabilize before you possibly make another key hit. (When I started making landings, I found the rudder to be hectic and I prefered to use the ailerons to turn. As experience build up, I finally found out that turning with the rudder allowed for more precise and comfortable adjustments.)

10. The airplane may oscillate a little. Don't bother. Just keep in control using the yoke.

11. You're flying at constant altitude and 70 knots speed. Once the beginning of the runway passes under the engine cover, it's time to take things up seriously. This is shown in the picture below. (Whatever altitude you are flying, once the engine cover begins to eat the runway, you are at a correct angle towards the runway start.)

12. Type] two times, to deploy the full three flaps steps.

13. Immediately push the yoke forwards, to make the airplane plunge to the ground. Indeed, the full flaps deployed make the plane brake. You plunge towards the runway to land, of course, but also to keep the speed at 70 knots.

14. Decrease the engine power. $\frac{1}{4}$ the maximum is often fine. The Cessna 172p needs even less. I tend to decrease the engine power throughout the dive, to end with almost no power. The possibility to add engine power is necessary for your safety and for the precision of the landing, but also the possibility to decrease the engine power. So I try to make my dives a way that I keep the engine at some decent power level throughout the dive. If a dive obviously begins with the need to decrease the power to minimum, there is a risk you touch the runway far beyond its start.

15. Watch the speed indicator like it if was your heartbeat. Control the speed using the yoke/elevator.

16. The dive makes you head towards the runway. You will soon become aware that the plane is going towards a point of the runway much further than the start of the runway. There is nothing wrong with that on a long runway. Yet you should train to land on short runways. In order to correct the dive and head towards the start edge of the runway, decrease the engine power. (On the Cessna 172p this often leads to power to minimum, while on most other airplanes you keep some power tuned in.) The picture below is a snapshot from a good dive. (Note the vertical speed indicator shows -500 feet/minute. I never use that indicator. I solely aim at the runway edge and its 12 white strips. Anyway, -500 feet/minute is the right descend speed...)

17. Closely keep the speed at 70 knots by pulling and pushing the yoke. Calmly increase and decrease the engine power in order to head the plane towards the starting line of the runway. Don't bother to aim exactly at the start of the runway. It doesn't matter if you arrive a few feet before the runway start or much further after it. Provided you arrive at 70 knots.

18. Keep aligning with the runway, using the rudder pedals to turn (keypad keys **0** and **Enter**). Keep the wings level with the horizon using the mouse/yoke/ailerons. (Use the ailerons to turn only if an emergency occurs and you need to make fast and steep turns. Then you probably need to abort the landing and get back to altitude (see below).)

19. If you suddenly realize you will arrive really far before the beginning of the runway, possibly retract the flaps to one step (**[**). You can also let the engine roar to maximum power for a few seconds. If you followed the procedure you shouldn't need to do such extreme things... (At any time, if you feel things are going wrong, retract the flaps to one step, throttle to full engine power, put the trim on neutral and gain back altitude (keep the speed above 70 knots). Whatever wrong happens – you arrive aside from the runway, too far before the runway, at a wrong speed, a swarm of birds is passing, whatever – abort the landing. Get back to altitude and retry.)

20. The "rounding" is the most impressive part. You are like going to crash on the runway. Yet you will pull the yoke/mouse before it's too late. Don't pull on the yoke too early. Don't pull on the yoke too firmly. Once you are really close to the runway (for a beginner: once you are convinced it's too late and you are going to smash into the ground), pull the yoke gently and bring the plane in a steady flight above the runway. That's the rounding. (During the rounding, ground effect contributes to your security and ease.)

21. It is often best to reduce the engine power to minimum during the rounding.

22. Go on using the rudder pedals (keypad **0** and keypad **Enter**) to keep aligned with the runway. Use the yoke/ailerons to keep the wings level with the

horizon (so both left and right wheels will touch the runway at the same time).

23. Now you're flying close above the runway (in ground effect). Throttle the engine power to minimum if it wasn't already done (this is mandatory). Deploy full flaps if they weren't already deployed completely (this is not mandatory on a long runway). (Don't shut the engine down. Just throttle to minimum power. It still can happen that you suddenly must take off again and need full power in a few seconds.)

24. Keep the plane flying close above the runway. As the speed decreases from 70 knots down to 50 knots and below, keep pulling more and more on the yoke/mouse, steadily. Keep the plane in the air while ensuring it stays really close to the surface of the runway. Steadily lift the nose, while the plane slows down, up to quite a strong angle. Make sure the plane does not gain back altitude (don't look at the instruments, look at the outside). You really have to avoid the plane rises back in the air. Indeed it would do that at a speed below 70 knots... (You shouldn't need to pull the yoke more than $\frac{1}{2}$ its maximum.)

25. Don't land the plane. Let it land by itself, once the speed is too low and the nose is high up in the air. The plane renounces to fly, it calmly sinks in and the two rear wheels touch the runway. If you don't hear the wheels hit the runway and the wheels nevermore leave the runway, you probably made an optimal landing. This also makes the front wheel stays above the runway while the two rear wheels touch.

26. Once the rear wheels roll on the runway, retract the flaps. That way the wings will lift less and the plane will be more firmly on the ground. (My favorite way to land the airplane is to let the flaps down and keep pulling on the yoke while the airplane is rolling. That way I get maximum braking. I suppose this is an example of the difference between a simulator and reality. Using my way the airplane risks to get back in the air any moment and it is very sensitive to blows of wind. If I made real landings, maybe I wouldn't dare do this...)

27. When the plane is rolling, an optimal position for the yoke/elevator seems to be pulled $\frac{1}{2}$ of the total way.

28. Use the rudder pedals to keep the plane rolling in the middle of the runway and straight while the speed decreases. This most often leads me to two keypad **`Enter`** hits to the right of the center position.

29. Once rolling at a speed below 40 knots, the nose will go down automatically. Help it by pushing the yoke/mouse calmly, back to neutral position. The front wheel now must touch the runway. Beware: check the rudder position

first. If it is too much to the left or to the right, the plane will turn violently once the front wheel touches the runway. The plane may even fall aside and hit the ground with a wing tip. (The rudder slightly to the right; two keypad **Enter** hits, seems an optimal position.)

30. Now the front wheel is on the ground, use the mouse to control the rudder. Keep the left mouse button down and forget the keypad keys. Maybe just check the ailerons and elevator positions are sound before you press the left mouse button. (Actually if everything went correctly and there is no cross-wind, you shouldn't need to steer the plane using the rudder.)

31. Once the front wheel is on the ground, you are allowed to use the brakes. Your choice. Keep the **b** key down. Be prepared to release it should a problem occur. If you forgot to almost center the rudder, braking can go really bad.

Once the plane is halted or at very low speed, you can release the **b** key (if you used it) and add a little engine power to taxi to the parking or hangar.

To shut the engine down:

- Engine throttle to minimum (hold **Page Down** down for a while).
- Pull the mixture lever to halt the engine (mouse in normal pointer mode, click on the left of the red mixture lever to pull it out).
- Rotate the magneto switch to OFF (a few hits on **{**).

To set the parking brakes in, type **B**.

You must be mentally prepared to abort landing anytime. Whatever happens: an order from the control tower, a wrong speed or landing angle, a wrong alignment with the runway, a strong blow of wind, birds flying over the runway... retract the flaps to one, push the engine to maximum, center the trim and get back to high altitude. Then either you restart the landing procedure or you go for another airport. The pride of a pilot is to make only safe landings.

Don't try to find "the ideal distance" to start diving to the runway. The procedure above proposes you start diving when the white engine cover starts eating the runway edge (provided you fly at 70 knots with one flaps step) (the altitude doesn't matter). Best is you train to land while starting the dive earlier and while starting to dive later. You need to be trained to increase or decrease engine power according to what is needed. During a real landing, depending on the airplane's weight, the wind speed and other random things, the "ideal" moment to dive is unpredictable. As experience builds up, you will better feel the right moment.

If you want to make things simple for your first landing trainings, make use of the fact the runway at KSFO is very long. Wait a little more before you begin

the dive: let the nose "eat up" the whole length of the leading part of the runway (let the successive pairs of white strips on the runway disapear below the airplane nose). Then lower the flaps to three steps and decrease the engine to minimum. Dive to keep the speed around 70 knots and try to keep aligned with the runway. You will end the dive quite far beyond the runway start and at a high vertical speed, but who cares. Make the final rounding. Keep aligned with the runway and try to fly close above it. Keep pulling more and more on the yoke/mouse, to keep the airplane flying. Yet avoid it rising in the air. Till the wheels touch the ground. Then just keep the airplane on the runway, using the rudder. Once the speed is below 40 knots, push the yoke/mouse and keep key **b** down to brake.

If you are a newbie, you probably won't succeed to apply the procedure perfectly. My advice: invent your own, more simple procedure. Then regularly come back to the procedure listed here and read it again, to get hints and ideas to better your procedure. Till you get it. Also best read other landing procedures. Send me a mail if you find interesting differences. Analyze your own procedure. If it implies to fly at very low speed, it is dangerous because a blow of wind from the rear will make the plane fall. A probable problem with your procedure is the plane needs a lot of runway length to land. If you look at the runway start you will see there are successive groups of white stripes. I land the Cessna 172 always well before the last group of stripes. If you are a real beginner, your procedure surely will make the plane tilt over or crash once in a while. The procedure listed here is safe. Train your procedure, again and again. The more you train it, the more you will become able to use the one listed here. That's the way I learned to land...

> ! In a real airplane, you must keep in touch with the control tower constantly while landing. You will be contacted by the control tower or you have to contact it in some key parts of the landing. If you don't contact the control tower just after landing, an emergency rescue team is immediately underway. If there is no good reason you didn't contact the tower, you will really be in trouble.

Maybe you'd like to train landing without having to take off and circuit in order to head for the runway and land. Type the command line displayed below in a terminal window to start the simulator in flight and heading for the runway. The airplane is placed 6 miles ahead of the runway, at an altitude of **1000** feet and a speed of about **120** knots.

```
fgfs -offset-distance=6 -altitude=1000 -vc=120
```

Possibly add `-timeofday=noon -geometry=1024x768` jparameters if you need daylight and a bigger window (choose anything you need instead of 1024×768 (I favor 1200×900 an my screen)). FlightGear command line parameters are listed in
http://www.flightgear.org/Docs/InstallGuide/getstartch4.html#x9-330004.4

(Note the parameters above make the airplane have some trim tuned in. Yet you need another trim tuning during the horizontal steady flight towards the runway. See the section 7.7.5 above, about the trim. If in doubt, just center the trim. On the Cessna 172p, a centered trim seems the right position.)

Once you are trained, you no longer need to do a long horizontal flight at 500 feet and 70 knots to get to the runway. Instead you can descend all the way from your flight altitude and at a higher speed. You should be able to get at 500 feet and 70 knots a short while before the final dive.

Landing at 65 knots instead of 70 knots allows to use a much shorter runway length. Yet to benefit from this you better train landing at 65 knots. It is quite different from landing at 70 knots.

The landing speed varies according to the load of the airplane. The more load of petrol, passengers and freight, the higher the optimal landing speed will be.

7.9 “My Friend the Wind”

7.9.1 How to fly when there is wind

Think of a hot air balloon. Think of it as being in the middle of a gigantic cube of air. The cube of air may move at high speed compared to the ground, anyway the balloon itself is completely static in the middle of the cube. Whatever the wind speed, persons aboard a hot air balloon experience not the faintest blow of wind. (To pilot a hot air balloon you bring it at an altitude where the wind blows in a direction that more or less suits your needs.) The same way, an aircraft flies in the middle of a gigantic cube of air and only refers to that cube of air. The motion of the cube of air compared to the ground doesn’t bother the aircraft.

You, the pilot, on the contrary, do bother for the speed of the surrounding air compared to the ground. It can make you drift to the left or to the right. It can make you arrive at your destination much later or much sooner than planed.

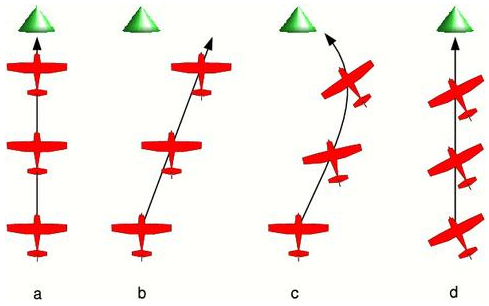
When the wind blows in the same direction as you fly, the speed of the wind adds itself to the airspeed of the plane. Hence you move faster compared to the ground. You will arrive earlier at your destination and have less time to enjoy the landscape. (It sometimes happens that a jet airliner flying with a strong wind from the rear, moves faster than the speed of sound compared to the ground. Though it doesn’t brake the sound barrier.)

When the wind blows in the opposite direction you fly (towards the nose of the plane), the speed of the wind subtracts itself from the airspeed of the plane. Hence you move slower compared to the ground. You will arrive later at your destination and have more time to enjoy the landscape. (Some slow airplane flying against strong wind can even seem to fly backwards, because the speed of the wind

is faster than the flight airspeed of the airplane.)

The two cases above are quite simple. More complex is when the wind blows towards the side of the airplane. Look at the pictures below.

- On picture (a) there is no wind. The pilot wants to reach the green hill situated to the North. He heads for the hill, towards the North, and reaches the hill after a while. When there is no wind, you just head towards your destination and everything's fine.
- On picture (b), the pilot keeps heading to the North. Yet there is wind blowing from the left; from the West. The airplane drifts to the right and misses the hill.
- On picture (c), the pilot keeps heading towards the hill. This time he will arrive at the hill. Yet the plane flies a curved path. This makes the pilot loose time to get to the hill. Such a curved path is awful when you need to make a precise navigation. (Note something: the airplane tends to get into the wind, like a weather vane.)
- Picture (d) shows the optimal way to get to the hill. The plane is directed to the left of the hill, slightly towards the West. That way it compensates the wind and keeps on a straight path towards the hill. It will need more time to reach the hill than if there was no wind, anyway this is the best attitude. (Note something: the solution is to let the airplane head a little bit into the wind, like a weather vane would.)



How much to the left or to the right of the object must you head? At what angle? Serious pilots use tight geometry and trigonometry computations to get near exact and optimal angles. Yet I wouldn't fly a virtual Cessna 172p if I had to do such dry things. You need no computations at all to fly roughly straight. The trick is you must keep your eyes on the object you fly towards. You know you will head the plane in a direction to the left or to the right of the object, but you don't need to know the angle. Just keep your eyes on the object. Get aware you are drifting leftwards or rightwards. Then let your instinct slowly head the plane

to the right or to the left to compensate the obvious drift. When you begin training this, you need to force your instinct a little bit and think of what you are doing. Very soon this will become automatic, just like when you learned to fly straight. You will no more keep the plane headed towards the object. You will rather keep it flying towards the object. The picture below shows a flight towards the top of the little mountain ahead. Wind blows from the right. I just look at he mountain top. And I let my hands head the plane to right of the mountain, without really thinking about it:

The faster the flight airspeed compared to the wind speed, the less the wind will influence.

7.9.2 How to take off when there is wind

Main recommendation to take off is you must find a way to accelerate facing the wind; with the wind blowing towards the nose of the airplane. Before most runways are build, statistics are made about the wind at that location. The runway orientation is chosen so it aligns with the wind most often. Lots of airports have two runways at different orientations because the wind sometimes blows in one of these directions and sometimes in the other direction. The location of an airport is often chosen because at that place the wind often has a stable direction and speed.

Take off with a faint wind blowing towards the rear of the airplane, say 1 knot, for sure is no problem. Yet above a few knots you can get into trouble. With a 10 knot wind blowing from the rear, the front wheel will rise at the usual 40 knots airspeed, but that makes 50 knots compared to the runway. What matters is the speed the front wheel roll over the runway, not the airspeed... If a problem occurs and you are still rolling at 60 knots on the runway, the consequences will be more dramatic. To end with, you will need much more runway length and have less opportunities to abort the landing.

The main way to know the wind direction and speed is to go to the control tower or ask the control tower by radio. A necessary and complementary tool are

the windsocks at both ends of the runway. They show the wind direction and speed. The longer and the stiffer the windsock, the more wind there is. The windsock on the picture below shows an airspeed of 5 knots:

So, you have to choose a runway start that allows you to take off with the airplane facing the wind. In real life you are not always allowed to do this. Either there is no runway aligned with the wind or the control tower tells you to use another runway. Then you have to take off under crosswind; the wind blowing towards a side of the airplane.

Basically, you can use the exact same procedure as listed above for a take off when there is no crosswind. Yet you have to be aware of several important facts listed below. To train this, start FlightGear with the parameter `-wind=0@10` which implies a wind of 10 knots blowing from the North (direction 0). If you take off from the usual San Francisco KSFO airport heading to the West, this makes the wind blow from the right.

- You will have to push the rudder at quite a strong angle to stay rolling aligned with the runway. Keep the rudder at that angle once the front wheel leaves the ground and a little later once the rear wheels leave the ground.
- Say the wind is blowing from the right. You would think you have to push the right rudder pedal, to head the airplane a little bit into the wind, to compensate for the leftwards push of the wind. Well you have to do the exact opposite: push the left rudder pedal. This is quite unnatural yet that's life. The reason of this is the rear vertical stabilizer is pushed by the wind leftwards. The plane reacts like a weather vane and heads in the wind. The plane as a whole turns to the right, with quite a strong force. You have to compensate by pushing the rudder to turn to the left. So, you take off with the ruder pedals pushed to the left. The picture below shows a rudder angle during a take off with a 10 knots crosswind blowing from the right:

- The airplane will tend to bank leftwards. Hence you will have to push the yoke/ailerons to the right. Actually you best place the yoke a little to the right before the wheels start leaving the ground. (Best is to push the yoke to the right from the start on. Indeed this is the best way to taxi safely under a crosswind blowing from the right.) The picture below shows an appropriate yoke/ailerons position, while taking off with that 10 knots crosswind blowing from the right.

- The airplane will rise in the air much slower. The vertical speed will be quite weak. This is because the rudder is at a a strong angle. The airplane moves through the air with its right flanc and brakes. You have to wait till the rudder is centered before you get the regular vertical speed. Center the rudder very slowly, a little angle step at a time. Meanwhile, using the yoke/ailerons, gradually head the airplane a little bit in the wind, to keep flying aligned with the runway. Wait till you are above a few hundreds feet before you start centering the rudder.

Why do you keep the yoke to the right and the rudder pedals to the left once the airplane rises in the air? This can seem odd. It's quite logical that way the airplane will fly straight. The ailerons and the rudder compensate each other and the airplane turns neither to the right, neither to the left. But again, why do this, why not simply let the yoke/ailerons and the rudder centered? The airplane will fly straight too and be far less braked. The reason why we do this is the ailerons keep the airplane banked to the right; towards the direction the wind is blowing from. Hence, the huge force on the wings, that keeps the airplane in the air, that huge force is now slightly directed to the right. In normal circumstances this would make the airplane move slowly sideways to the right, at 10 knots speed... Currently, it compensates for the 10 knots wind and keeps the airplane above the runway. So despite the wind, the airplane stays headed towards the runway end and stays above the runway middle. Everything's fine (except for the braking).

To me, 10 knots wind is a maximum to take off the Cessna 172p safely.

7.9.3 How to land when there is wind

You land the Cessna 172p under crosswind the same way you take off:

- Try to land with the wind blowing towards the airplane face. Bear in mind the wind blows the airplane away from the runway start. So start the dive later, when the engine cover already ate some length of the runway.
- Under crosswind, use the exact same rudder and ailerons tuning as for take off under the same crosswind. Train this by taking off and landing under crosswinds. When the wheels leave the ground and you find the appropriate yoke/ailerons angle, note down the rudder angle and the yoke/ailerons angle. Center the rudder and ailerons during the flight and make a circuit to land back. During the landing, when you fly at constant 500 feet altitude and 70 knots speed, knowing the crosswind is the same, tune in back the rudder and ailerons angle that where optimal during take off.

Under high crosswind, hence with a strong rudder angle, the plane brakes a lot. This implies two things:

- During the approach to the runway, at constant 500 feet altitude, 70 knots speed and 1 flaps step, you need much more engine power to keep the altitude stable.
- Once you dive towards the runway start, keep in mind the plane is braking. So you don't need to deploy additional flaps steps. Just decrease the engine power.

Landing that way is quite comfortable, despite the crosswind. You just have to be a bit more careful with the rudder once the airplane rolls over the runway. And best keep the ailerons as if turning towards the wind.

Note such a landing, with a steady crosswind, is unrealistic. In the real world the wind varies quickly. You get sudden increases and gusts of wind. The control tower just tells you by radio the maximum speed of the gusts. You have to adapt constantly during the landing, to react to the turbulences and gusts.

As for the take off, 10 knots wind seems a maximum to me. (Should you ever have to land under heavy wind, say 25 knots or more, and there is no runway aligned with the wind, maybe best don't land on the runway. Or don't try to align with the runway. Align exactly with the wind and make use of the fact you need less ground length to stop. When the plane is going to stop keep the rudder pushed. Don't try to taxi. Simply push the parking brakes in, push the trim and get help to latch the airplane to the ground. In fun mode, landing the Cessna 172p under 70 knots wind is great. You simply let it descent to the ground vertically. This is quite unrealistic because at such a wind speed there are tremendous turbulences close to the ground.)

The technique described here is the slip landing. Another crosswind landing technique is the crab landing.

7.9.4 How to taxi when there is wind

Under 10 knots wind the Cessna 172p seems not to need particular precautions when taxiing. Yet any sudden increase in wind speed can tilt it and tumble it over. So best apply the recommendations whenever there is wind.

To train taxiing on the ground when there is wind, ask for a strong wind like 20 knots. Such a wind can tilt the plane and blow it away tumbling any moment. One single error during taxiing and the plane is lost.

Main rule is you must *push the yoke towards the wind*. This deserves some physical explanation:

- When the wind is blowing from 12 o'clock, this is quite logical. The yoke is pushed (towards 12 o'clock) and the elevator makes the tail rise a little. That's the most stable position to avoid the plane be tilted by the wind.
- When the wind comes from 10 o'clock, pushing the yoke towards 10 o'clock makes the elevator is close to centered. The elevator almost no more trades in. Now the most important part is played by the ailerons. The left aileron is upward and the right aileron is downward. This pushes the left wing down and lifts the right wing. Again, that's the most stable position to avoid the plane be tilted by the wind.
- When the wind blows from 8 o'clock, you would think you should invert the position of the ailerons, to keep the left wing being pushed down. Hence you should push the yoke to 4 o'clock. Wrong! Keep pushing the yoke to 8 o'clock. The reason is the downward position of the aileron on the right wing makes it act like a slat. This increases the lift on the right wing and this is all we want. Symmetrically, the upward position of the left aileron decreases the lift of the left wing.
- When the wind comes from the rear, from 6 o'clock, the yoke is pulled (towards 6 o'clock). The upward position of the elevator tends to make the tail be pushed down. Once again this is the best. Strong wind can push the tail against the ground. This is impressive but the tail is conceived to withstand this.

Accept the plane nose can be tilted and the tail pushed against the ground. Keep cool. This can be impressive yet there is nothing dangerous with it. Go on using the brakes, rudder and engine to move the airplane.

If you want to move towards the wind, you will need more engine power. When the wind blows from the rear you may need no engine power at all. Always keep the engine power to the minimum needed.

Especially when turning, move very slowly. Make little changes at a time. Take your time and closely survey the yoke angle. Constantly keep it pushed towards the

wind. Constantly try to reduce the engine power. Keep in mind using the brakes too firmly may shortly tilt the plane at an angle that allows the wind to tilt it and blow it away.

7.10 The autopilot

An autopilot is not an “intelligent” pilot. It just takes over simple and wearing parts of your work as a pilot. You still are the sole real pilot aboard and have to keep aware of everything. Be prepared to shut the autopilot down. During take off and landing, relying on the autopilot would be suicidal, because you have to keep an immediate control on every function of the airplane. (Dumb autopilot systems are reported to cause less accidents than smart ones with artificial intelligence inside.)

The autopilot is that little rack to the right of the yoke:

Switch it on by pressing its **AP** button (standard mouse mode). The autopilot then controls the roll. It keeps the wings level with the horizon. This is displayed in the picture below by the “ROL”marking. To switch the autopilot down press again on **AP**.

If you press the **HDG** button the autopilot will try to keep the plane flying towards the direction tuned on the directional gyro by the red marking (see the section 7.7.6 about direction). “HDG” stands for “heading”. Press again on the **HDG** button to get back to roll control mode (or **AP** to switch the autopilot down).

The buttons **ALT**, **UP** and **DN** are used to tell the autopilot either to control the vertical speed VS or the altitude ALT.

From here on you maybe better study the document used by the author of the autopilot system in FlightGear:

https://www3.bendixking.com/servlet/com.honeywell.aes.utility
.PDFDownLoadServlet?FileName=/TechPubs/repository
/006-18034-0000_2.pdf

7.11 Security

Security is first of all a matter of common sense. Avoid to land with the landing gear retracted. Fill the reservoirs before take off and don't let them get empty in flight. This may seem funny recommendations, the fact remains I made several landings on the aircraft belly when I started using the flight simulator. I got angry on myself and now it nevermore happens that I forget such a simple and essential thing. In real life you are not allowed to land airplanes on the belly in order to get angry on yourself. I suppose it is a part of the role of the monitors to make you feel the angriness **before** your first solo landing. I suppose they don't let somebody fly on his own till they feel the angriness is rooted deeply enough in him. People who cannot cope with this are not meant to become pilots.

There are much more vital details than the landing gear and the fuel. That's why checklists exist. There are checklists for all kinds of normal or emergency situations. There a long checklists and short checklists. This link provides checklists for the Cessna 172p and for other airplanes: http://www.freechecklists.net/. Those checklists refer to much more levers, buttons and triggers than talked about in this tutorial. There is nothing complicated in those checklists provided you learned what all those little things are. For example one item is you have to verify the seats backs are upright.

You have to learn to cope with stress. Wherever I get access to computers I try to install ***FlightGear***. To me the computer industry should focus solely on building computers for ***FlightGear***. Secundary tools like browsers, mailers, spreadsheets and the like, should be regarded as optional sub-functions of ***FlightGear***. Once the installation is finished, I make a demo flight. Strangely, most people simply don't care about what I am doing. They just go on talking, asking questions, requesting

my attention... What's more I'm often not in the most adequate position toward the screen, the keyboard and the mouse. It becomes almost impossible to fly correctly, especially to land. Basically there are two possible attitudes. The first one is I get silently angry on the disturbing persons, I stop the demo and I consider it's their fault if I cannot succeed my flight. The second attitude is I breath deeply and calmly, I find ways to go on managing the burdens and the problems, I don't get angry on anybody, I claim nothing to be responsible for anything, I renounce to make a perfect demo flight and I focus on making a mediocre yet secure landing. The advantage of the first attitude is that you feel comfortable about your superiority on ***FlightGear***-unaware persons. The disadvantage of the second attitude is that you have to endure the humiliation of an ugly landing and the people around going on talking and requesting your attention. The advantage of the second attitude is that in real life, on a real airplane, it allows you to stay alive.

Communication is a basis for security. That means communication with the technicians, with the control tower, with your copilot, with the passengers and especially with yourself. You have to constantly gather data about the traffic, the meteorology and the state of mind of your passengers. You have to constantly inform the control tower and obey the instructions it sends you in return. You have to keep your passengers in an acceptable mood and at the same time you have to obtain they let you focus on your tasks when this is necessary. Lots of airline accidents occured because of a lack of communication between the pilot and other crew members. That has been called "the Superman syndrome". Once the problems start, the pilot focuses on his way to solve the situation. Either the copilot does not understand what the pilot is doing or he becomes aware of a danger the pilot did not realize. This results in contradictory commands sent to the airplane controls, shouting, up to fist fighting... till the final crash of the airplane. An important part of the training for modern pilots is to learn to communicate with the other crew members under high stress. They learn to go on communicating and how to do that a short and efficient way. (I was once told this anecdote: a monitor and a trainee were performing landings. The trainee was a strong guy with muscles like truck tires. At one moment the landing path appeared to be wrong. The monitor asked the trainee to release the commands so he could take them over. There is nothing wrong with failing a landing. Monitors themselves sometimes fail a landing, abort and restart a new landing. But the trainee panicked and crispated his hands on the yoke. The monitor could do nothing. The consequence was a damaged landing gear.)

There is no room for luck in real aviation. When you train to become a pilot, almost every possible situation is put into practice at least once. For example a monitor makes you take off with a heavily loaded airplane and suddenly shuts the engine down. You have to train to fly and land with a random airplane control or indicator out of order. ***FlightGear*** allows to reproduce some of these trainings. You can request flight instrument failures using ***FlightGear***s' menus or command options. A really bad instrument failure means the instrument still seems to operate

correctly. Yet it doesn't, and what it does or displays endangers you. While training you can decide to no more use a given instrument or control. For example you can glue a sticker on your screen to hide away an instrument. Best is you ask a friend to configure a failure without you knowing what he did. This heavy training and the numerous precautions and rules are the reason why so few accidents occur. In most cases, even a severe problem does not lead to an accident. Accidents are often due to the unlucky addition of several different problems.

The picture below shows the artificial horizon indicator. I hardly never use it. I fly looking at the real horizon. Anyway the artificial horizon saved me more than once on the simulator. When you penetrate by mistake in a cloud or a bank of mist, you suddenly get a white outside. There is no more way to keep the plane flying level, except by using the artificial horizon. You may argue this due to the lack of feedback own to the simulator. You're (dead) wrong. The same problem occurs on a real airplane. Quite many of the (very few) accidents in little airplanes like the Cessna 172 or the PA-28 happen that way. It is prohibited for a pilot with no IFR license to enter a cloud. Some do it anyway. Or they get caught in a rise of mist the control tower didn't warn for. The airplane banks and in two minutes time it goes flying upside down. The pilot is unaware of this. Even worse: some instruments seem to get mad, with no obvious reason. A crash is unavoidable. I learned the reflex to focus on the artificial horizon, the altimeter and the directional gyro. When this happens the plane is often already severely banked. I keep calm and I use the instruments to maintain the plane in a sound flight. It will oscillate a lot but serious problems will be avoided. Either I will wait till I get out of the cloud or I will gain or loose altitude to get out of the cloud layer. I strongly advice you train this using the simulator. Best is you make a complete IFR training.

One thing you have to train for your security is landing on very short distances. Some flight incidents, like an engine failure or a sudden change in the weather, can force you to land on the first strip of flat land you encounter.

The HUD allows to fly and land more easily, with less stress. It also allows to optimize what you are doing and this is good for security. For example it allows to touch the ground very close after the beginning of the runway. That way you have the whole length of the runway to brake. (A HUD is available for every aircraft on FlightGear, even the 1903 Wright Flyer. In real life, few little civil airplanes contain a HUD. It is too expensive and too recent.)

There are some strong differences between a flight simulator with minimalistic control hardware and a real airplane. The fact the mouse exerts no counterforce, the fact you don't feel the vibrations and forces inside the airplane... On one hand, some aspects of flying are made easier on the simulator. On the other hand, a real airplane constantly gives all sorts of valuable feedback you don't get with a simulator. One thing is common to the simulator and the real airplane: while landing you'd wish you had four arms and two more brains.

FlightGear contains bugs. Consider those problems as a training for real aircrafts. Problems on real aircrafts are not the same. But there are problems. When ***FlightGear*** suddenly puts you in a critical situation due to a bug, consider this as a training. Try to solve the situation fast and efficiently while keeping calm. It's not a bug, it's a feature!

The handbooks of airplanes contain procedures and checklists for emergency situations. It sometime happens that the adequate reaction to a problem is exactly the opposite for two different airplanes. That's one reason airline pilots are not allowed to fly different airplanes at the same time. If they choose to go flying another type of airliner, there are imposed to stop flying for a lengthy period, during which they will practice the other type of airplane on simulators. The wide range of aircrafts available under ***FlightGear*** allows you to experiment with this.

7.12 What then?

Once you master the content of this tutorial, you can claim to have a basic understanding of what piloting is about. You still lack key knowledge and training, like these:

- How to follow real checklists.
- How to make emergency landing on very short fields, possibly with no engine power.
- How to navigate according to the rules, charts, laws, radio beacons and crosswinds.
- How to draw a flight plan.
- How to place the loads in an airplane to get a correct center of gravity.

- How to deal with the control tower and with other airplanes.
- How to deal with several fuel reservoirs and their valves, pumps and backup pumps. If two reservoirs are located on the wings ends and you let one of them empty while the other keeps full, you will get severe problems.
- How to deal with the failure of every possible part of the plane.

You probably will learn to deal with a retractable landing gear system and with variable pitch propellers.

Go to the ***FlightGear*** documentation page for more tutorials and reference pages:
http://www.flightgear.org/docs.html

These are great tutorials to learn further:

- http://www.flightgear.org/Docs/Tutorials/crosscountry/tutorial.html
- http://www.navfltsm.addr.com/

I wish to thank:

- Benno Schulenberg who corrected lots of mistakes in my English in this tutorial.
- Albert Frank who gave me key data on piloting and corrected technical errors.
- Vassilii Khachaturov who learned me new things about ***FlightGear***.
- Roy Vegard Ovesen for pointing me to the official Autopilot Pilots Guide.
- Dene Maxwell for his solution for problems under Windows Me.
- Mark Akermann and Paul Surgeon for their remarks.
- Michael “Sam van der Mac” Maciejewski who made the translation in Polish and converted the tutorial in usable TEX format.
- The ***FlightGear*** mailing list users for their hearty welcome.
- 4p8 webmaster my friend Frédéric Cloth for the web space used by this tutorial.

7.13 'Pilots Fly Not Only On Cessna'

I cross-checked all the data about the Cessna 172p, a friend who is a pilot verified I did not write too blatant crap and I made numerous virtual test flights. This appendix contains less reliable data about other airplanes. It can be helpful as an introduction to those airplanes but keep in mind my only goal was to make flights that seem OK and acquire basic knowledge. You need other sources if you want to pilot these airplanes a serious way.

7.13.1 How to land the Cherokee Warrior II

On Linux you get the Cherokee Warrior II (or PA-28) with the `-aircraft=pa28-161` command line parameter. The Cherokee Warrior II has some advantages upon the Cessna 172p. Thanks to its low wings it is far less sensitive to crosswind. Fully extended flaps are more braking and allow to land on a much shorter distance.

Take off is the same as for the Cessna 172p (in ***FlightGear***. In real life their take off checklists are not exactly the same).

You have to get used to some minor differences of the Cherokee Warrior II for the landing:

- During the steady horizontal flight before landing, the trim must be pulled a little below neutral in order to get the yoke oscillating around neutral.
- The optimal tachometer RPM during landing is at a lower RPM than the tachometer green zone. Roughly, keep the needle vertical.
- Only put one more flaps step (which makes two flaps steps deployed) when the dive towards the runway begins. Don't decrease the engine throttle too much.
- If you keep it to two flaps deployed during landing, the hover above the runway and the final roll will be similar to the Cessna 172p. Yet if you put the third flaps step in (after the final rounding), the plane will brake firmly. It will very quickly touch the runway then come to a near halt. Be prepared to lower the front wheel very soon. (It is possible to use the third flaps step during the dive towards the runway, instead of tuning the engine power down. Oscillating between two steps and three steps allows to aim the runway start. Yet keep two flaps steps and tune the engine seems easier. An interesting stunt is to fly stable till nearly above the runway start, then tune the engine to minimum and deploy three flaps steps. The plane almost falls to the runway. It's impressive but it works.)

(In real life, an advantage of the Cessna 172p upon the Cherokee Warrior II is the fuel reservoirs of the Cessna are located in the wings close above the center of the plane and higher than the engine. What's more an automatic system switches between the reservoirs. That makes you almost don't have to bother for the way the fuel gets to the engine in flight. On the contrary, on the Cherokee Warrior II the reservoirs are located separately, on both wings and lower than the engine. That means you have to constantly switch between the two reservoirs in flight. Should one reservoir become much lighter than the other, this would destabilize the airplane. The fact the reservoirs are lower than the engine means you have to control the fuel pumps and the backup fuel pumps.)

Some links:

- http://en.wikipedia.org/wiki/Piper_Cherokee
- http://www.alioth.net/flying/pa28-161/index.html
- http://faaflyingclub.homestead.com/files/Warrior_Checklist.pdf

7.13.2 How to take off and land the Piper J3 Cub

Use the `-aircraft=j3cub` parameter to get the Piper J3 Cub on Linux.

The Piper J3 Cub is a very different airplane from the Cessna 172p and the Cherokee Warrior II. The Cessna 172p and the Cherokee Warrior II are front wheel airplanes. While the Piper J3 Cub is a tail wheel airplane. Take off and landing with tail wheel airplanes is more difficult. You have to tightly use the rudder pedals when rolling over the runway. The yoke often needs to be pulled backwards to the maximum. I'll discuss this more thoroughly once I get more experience and knowledge about tail wheel airplanes. The Piper J3 Cub should be a good introduction and it is quite easy to take off and land provided you follow an appropriate procedure. Stall speed seems to be a little below 40 mph (the airspeed indicator is in mph) (about 27 knots according to the HUD). I guess an appropriate speed to rise in the air is a little above 50 mph.

My take off procedure for the Piper Cub is to fully pull the yoke backwards then throttle the engine to maximum. Once the front wheels clearly rises from the ground, gently push the yoke back to neutral, towards a normal flight close above the runway. Let the plane accelerate to 50 mph. Then pull the yoke to keep a little more than 50 mph while rising in the air.

The landing procedure... well in fact there are two different landing procedures:

1. The first one involves the fact the Piper J3 Cub is a very lightweight airplane. While still high in the air, throttle the engine down to minimum and slowly

pull the yoke completely in while the speed decreases. This slows the plane down to stall speed (a little less than 40 mph airspeed). It makes a steep descent to the runway. Keep the yoke pulled in completely. The wings seemingly act as a parachute. The plane hits the ground and bounces on its legendary gummy landing gear. It rolls at very low speed. While still pulling the yoke in to maximum, push in the wheel brakes (key **b**).

2. Second procedure lets you land the plane like a "normal" airplane. Yet with no flaps available, at quite a lower speed and with some big differences on the yoke

   - Fly at say 500 feet constant altitude and "exactly" 52 mph speed towards the runway (and align with it). Let the engine cover eat up the runway start. The engine cover will hide the runway completely. To see where the runway is, push the yoke/mouse very shortly then stabilize again in normal flight.
   - Once the runway start matches with the set of instruments (if you could see through the instrument panel), reduce the throttle to a near minimum and begin the dive towards the runway start. Keep 52 mph using the yoke. Add some throttle if you are going to miss the runway edge. (Keep in mind just a little wind is enough to change things a lot for the Piper J3 Cub).
   - Make the rounding and pull the throttle to minimum. Do not pull steadily on the yoke. Instead let the wheels roll on the runway immediately.
   - Once the wheels roll on the runway, *push* firmly on the yoke, to its maximum. This rises the tail in the air. You would think the propeller will hit the runway or the airplane will tilt over and be damaged. But everything's fine. The wings are at a strong negative angle and this brakes the plane. (Don't push the yoke this way on other airplanes, even if their shape seems close to that of the Piper J3 Cub. Most of them will tumble forwards.)
   - The yoke being pushed in to its maximum, push the left mouse button and keep it pushed to go in rudder control mode. Keep the plane more or less centered on the runway. This is quite uneasy. One tip is to stop aiming the rudder to say the left already when the plane just starts to turn to the left.
   - Once the speed is really low (and the rudder control stabilized), you will see the tail begins to sink to the ground. Release the left mouse button to go back to yoke control. Pull the yoke backwards completely, to the other extreme. The tail now touches the ground and the nose is high up. Now you can use the wheel brakes (**b**). (If you use the brakes too early, the plane nose will hit the ground.)

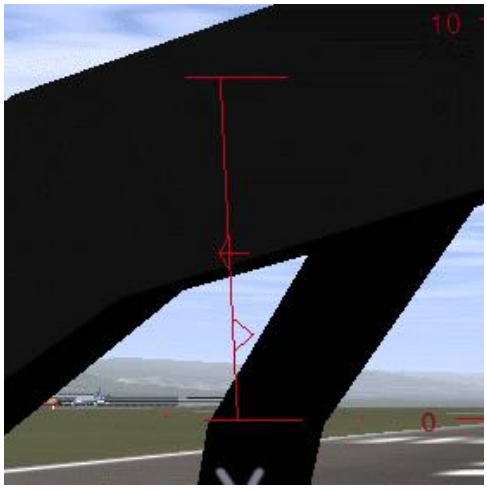
The take off procedure mentioned above is symmetrical to the first landing procedure. There exists a second take off procedure, symmetrical to the second landing procedure. Yet I don't succeed it properly so I won't write about it.

7.13.3 How to take off and land a jet

Take off on a jet is easy but you must have fast reflexes. My favorite jet on FlightGear is the A-4 Skyhawk. You get it with the `-aircraft=a4-uiuc` parameter on Linux, provided it is installed.

This is the "calm" procedure to take off:

- Ask for a red and full HUD by typing **h** two times. The engine throttle indicator is the leftmost on the HUD.
- The airspeed indicator is the one labeled "KIAS" on the upper left side of the instrument panel. You can also use the airspeed indicator on the HUD, of course.
- Tune in $\frac{1}{2}$ engine power.
- Keep the yoke pulled in $\frac{1}{2}$ of its total way (picture below: the red arrow on the right side of the vertical line in the middle of the picture).



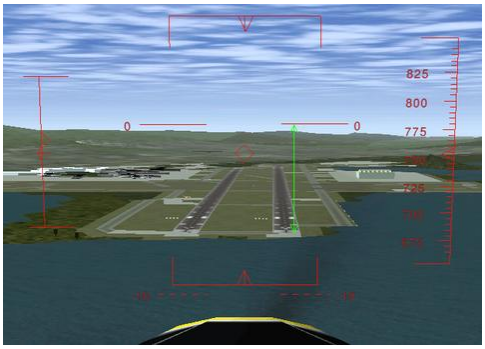
- It is not mandatory to use the rudder to keep on the runway. The airplane will take off before it drifts off the runway. (For sure it is better and more "secure" to keep in the middle of the runway. But using the rudder can make things hectic for a beginner.)

- Once above about 160 knots, the plane rises its nose in the air. Immediately push the yoke back to neutral or almost and stabilize at 200 knots airspeed (which makes a fair climb angle) (I've no idea whether 200 knots is the right climb speed for a real A-4. What's more I suppose one should rather use the AOA (see below).).
- Retract the landing gear using key **g**.
- Either maintain $\frac{1}{2}$ engine power and a speed of 200 knots to get above the clouds, or reduce the engine power to less than $\frac{1}{4}$ and fly normally. (Off course you can "fly normally" with full engine power. Great fun.)

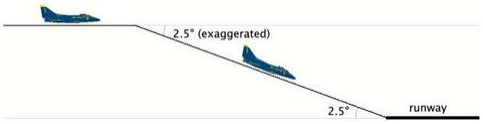
The "nervous" take off procedure is the same but you push in full engine power. The plane takes off quickly and you need to settle a very steep climb angle to keep 200 knots. Best retract the landing gear immediately.

You don't land a jet the same way you land a little propeller airplane. My way to land the A-4, inspired by some texts I found on the Web, is this:

- Really far from the runway, keep below 2,000 feet and get the speed below 200 knots. Then lower the landing gear (key **G**) and I deploy full flaps (all three steps, by hitting **]** three times).
- Keep a steady altitude of about 1,000 feet and a speed of "exactly" 150 knots. Use the mouse/yoke/elevator to tune the altitude and the engine throttle to tune the speed. (The opposite from the Cessna.)
- Try to align with the runway.
- When do you know the dive towards the runway must begin? For this you need the HUD; the full default HUD with lots of features. Look at the picture below. When you see the "distance" between the red "0" lines and the runway start is 25% the distance between the red "0" lines and the red "-10" dotted line, it is time to dive, aiming at the runway start. (In the picture below, that "distance" is 64%, far too much to start a landing.)

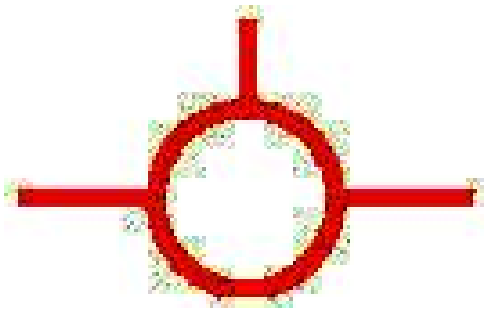


Let's explain this. The two horizontal lines labeled "0" show the horizon line. Rather they show where the horizon would be if the Earth was flat. When your eyes aim at those "0" lines, you are looking horizontally. Look at the dotted red lines labeled "-10". A feature on the ground situated there is situated 10° below the ideal horizon. In other words: when you look to objects "hidden" by the lines labeled "0", you have to lower your eyes of 10° to look at objects "hidden" by the dotted lines labeled "-10". This implies, and it is very important, that a person in a rowboat, "hidden" by the dotted lines labeled "-10", has to rise his eyes up 10° to look at your plane. He sees you 10° above the horizon. In the picture above, the start of the runway is situated at 64% of the way towards the red "-10" dotted lines. That means you have to lower your eyes of $6{,}4^\circ$ to look at the runway start. This also means that if you start now to descent towards the runway start, the descent path will be of $6{,}4^\circ$ (too steep). So, the HUD allows to measure precisely the angle of the descent path. On a jet plane you need an angle of $2{,}5^\circ$ (up to 3°), that is 25% of -10° (up to 30%).



- Once descending towards the runway start, aim at it using the yoke/mouse. And keep 150 knots speed using the engine throttle lever.
- Keep measuring the angle between the ideal horizon and the runway start. It must keep $2{,}5^\circ$ (that is 25% of 10°):
  - If the angle increases above $2{,}5^\circ$, you are above the desired path and you must loose altitude faster. Both decrease the engine power and dive the nose a little.
  - If the angle decreases below $2{,}5^\circ$, you are under the desired path. I wouldn't say you should gain altitude, rather you should loose altitude less fast. Both add a little engine power and rise the nose a little.
- Once very close to the runway start, do no rounding. Don't pull steadily on the yoke like you would for the Cessna 172p. Simply let the plane touch the ground immediately, at high speed. Let it smash on the runway, so to say. All three wheels almost together. Just throttle the engine down to minimum. (If you try to pull steadily on the yoke and hover over the runway while the plane nose rises steadily, on a F-16 you would scrape the plane rear and probably destroy it.)
- Keep the key **b** down to brake and use the rudder to stay aligned with the runway. Make only very little tunings with the rudder, otherwise the plane will tumble on one of its sides.

The HUD in a real jet contains a symbol to show towards what the airplane is moving. It is shown in the picture below. When you are flying at constant altitude, that symbol is on the ideal horizon line. Once you dive towards the runway start, you simply have to place that symbol on the runway start. This is quite an easy and precise way to aim at the runway start. (The diamond in the center of the FlightGear HUD sometimes can help but it does not have the same purpose. It shows towards what the airplane nose is pointing. For example is you descent towards the ground at low speed, the symbol would be somewhere on the ground while the FlightGear diamond will be up in the sky.) (By the way, the HUD on the virtual B-52 on FlightGear has that symbol. It is great to use while landing.)



Also, a real HUD shows a dotted line at $-2{,}5°$, to help find the correct descend path. Simply keep that dotted line too on the runway start.

In a real jet you don't look at the airspeed indicator to land. Rather you look at a tool on the HUD or at the set of three lamps shown below. When the upper $\vee$ is on, this means the speed is too slow. When the lower $\wedge$ is on, the speed is too fast. The center $\bigcirc$ means the speed is OK. This indicator exists in ***FlightGear***. On ***FlightGear*** version 0.9.8 it seems to have wrong speeds tuned in so I didn't use it. On FlightGear version 0.9.9 it seems OK. This indicator does not rely on the speed itself. Rather it relies on the AOA. That is the Angle Of Attack, the angle at which the wings are pitched up against the relative airstream. There is a close link between the AOA and the speed. I suppose the advantage of the AOA indicator is that the optimal AOA does not depend on the plane load. While the speed does. By tuning the correct AOA, always the same for every landing, you get the optimal speed whatever the plane load. (The A-4 on ***FlightGear*** has also an AOA indicator but I don't understand its output.)

The Cessna 172 and the A-4 Skyhawk are two extremes. Most other airplanes are in-between these two extremes. If you trained them both (and one or two tail wheel airplanes), you should be able to find out how to take off and land most other airplanes.

160 knots seems an appropriate landing speed for the F-16 Falcon. Also you need to throttle down the engine to minimum just before the plane should touch the runway. Otherwise it will hover over the runway. Don't bother for the flaps. It seems they are deployed automatically with the landing gear. (Read the section 7.7.4 about the stall).

140 up to 150 knots and all 8 flaps steps deployed seem appropriate to land the virtual Boeing 737. But don't trust me especially on that one. I just made a few experiments and didn't search for serious data. The landing speed varies a lot depending on the plane load, I suppose 140 knots is for a plane with no load. The Boeing 737 seems to like a gentle rounding before the wheels touch the runway. Start the rounding early.

In the take off procedure for the Cessna 172 and the A-4 Skyhawk I recommend you pull the yoke/mouse/elevator to $\frac{1}{2}$ the total way, from the start on. This seems to be a bad practice on the Pilatus PC-7. Keep the elevator neutral. Let the plane accelerate and wait till the speed gets over 100 knots. Then pull calmly on the yoke. During landing, deploy full flaps once you start plunging to the runway but don't decrease the engine throttle. Decrease it only when the hovering above the runway starts. 100 knots seems a good landing speed.

For the Cessna 310 too you better leave the elevator neutral during the acceleration on the runway. The plane will raise its nose by its own provided you deployed one flaps step. (If you keep the yoke pulled from the start on, the nose will rise sooner and you will get yawful yaw problems.)

(Some virtual airplanes, like some big airliners or fast aircraft, need faster physical computations. Then add the `-model-hz=480` parameter to the command

line. If the plane is difficult to control during landings, try this.)

The angle at which you land a Cessna 172p is far steeper than the narrow $2,5^{\circ}$ for a jet. Nevertheless you are allowed to land the Cessna at a narrow angle too. (Provided the terrain around the runway allows for this, of course.) If you have passengers who have ears problems with the variation of air pressure...

7.13.4 How to take off and land the P-51D Mustang

Should you ever get a chance to pilot a P-51 Mustang, just say no. It is quite dangerous to take off and land. That's the kind of airplane you fly only when your country is in danger. You need a lot of training. Yet once in the air the P-51 Mustang seems no more dangerous to its pilot than other common military airplanes. It is quite easy to pilot.

At low and medium altitude the P-51 wasn't better than the Spitfire and the Messerschmitts. The big difference was at high altitude. The P-51 kept efficient and maneuverable while enemy fighters were just capable to hang in the air. This was an advantage at medium altitude too because the P-51 was able to plunge towards enemy airplanes from high altitude. Another key difference was the P-51 is very streamlined. Hence it was capable to fly much further than the Spitfire. These two differences let the P-51 Mustang fulfill its purpose: escort Allied bombers all the way to their targets in Germany. This allowed the bombings to be much more efficient and contributed to the defeat of the Nazis.

To get the The P-51D Mustang in Linux use the `-aircraft=p51d` command line parameter.

To take off the P-51D Mustang in ***FlightGear***, deploy one flaps step, pull and keep the yoke completely backwards, push the engine throttle to maximum and keep the left mouse button pressed to control the rudder and keep on the runway. Once you reach exactly 100 mph, suddenly push the rudder 1/3 of its total way to the right. Immediately release the left mouse button and push the yoke to rise the tail (don't push it too much, as the sooner the wheels leave the ground the better). From now on, keep the left mouse button released. Only make very short adjustments to the rudder. Let the plane rise from the runway and get to altitude at a speed of say 150 mph. Don't forget to retract the landing gear and the flaps.

Don't make too steep turns. You would loose control on the plane and crash.

To land, deploy full flaps and lower the landing gear from the start on. 130 mph speed seems fine, up to 140 mph. Make an approach from 1,000 feet altitude and a dive at a low angle, like for a jet. Once over the runway, shut the engine down completely (key{). Don't hover over the runway. Get the wheels rolling soon (like for a jet). Hold the left mouse button down to steer the plane using the rudder. Once the tail sinks in, briskly pull the yoke (left mouse button shortly released) to

force the tail on the runway. Go on steering the plane using the rudder. Now the tail is firmly on the ground, use the brakes if you want.

7.13.5 How to take off and land the B-52 Stratofortress

The B-52F bomber implemented in ***FlightGear***is a success. It is one of my favorite airplanes. I'm sorry it was conceived to terrify me. One single B-52 bomber can wipe out every main town of my country and rise a nightmare of sicknesses and children malformation for centuries. All B-52 bombers united can wipe out mankind and almost every kinds of plants and animals on Earth.

The differences between the virtual B-52F bomber and the Cessna 172p are these:

- The B-52F starts with the flaps deployed and the parking brakes set.
- There are only two flaps steps: retracted and deployed. When deployed they are only meant to make the wings lift more, not to brake. If you want to brake, you need the spoilers. They deploy on the the upper side of the wings. Use the key **k** to deploy the spoilers and the key **j** to retract them. There are seven steps of spoilers.
- The main landing gear of the Cessna 172p is composed of two wheels, one on each side of the airplane. In order for these wheels to leave and touch the ground altogether, you need to keep the wings parallel with the ground. The main landing gear of the B-52F is composed of a set of wheels at the front and a set of wheels at the rear. This implies that in order for these wheels to leave and touch the ground altogether, you need to keep the airplane body parallel with the ground.

This is my procedure to take off the virtual B-52F:

- *Push* the yoke $\frac{1}{3}$ of the total way.
- Push the engine throttle to maximum.
- Release the parking brakes (key **B**).
- Push down the left mouse button to control the rudder pedals and keep the airplane on the runway
- The whole runway length is needed till the B-52F rises from the ground (KSFO).
- Once the B-52F leaves the ground, around 190 knots seems appropriate to get to altitude.

- Retract the flaps and the landing gear.

To land, the B-52F's HUD offers that great airplane-shaped symbol I talked about in the section about jets. So you just have to put that symbol on the airplane threshold (a few pixels further seems optimal) and keep the runway start 2,5° below the ideal horizon line. 130 up to 140 knots seems a good landing speed. (Instead of the speed you can make use of the AOA indicator displayed on the schematic instrument panel (**P**).). Simply keep the AOA at 3°. I must confess I prefer to tune the speed rather than the AOA.) If the plane gets to the runway at 130 up to 140 knots, simply "let it smash" on the runway. Otherwise, if the speed is higher, make a rounding and a short hover. The brakes seem to be very effective **b**). They allow to stop the B-52F on roughly the same short runway length as the Cessna 172p.

Replays of the flights are a delight. They allow to check the plane body left the runway and landed back parallel with it. One of the points of view is situated inside the B-52F rear turret, which allows you to be your own passenger and to compare what you see with what you experienced as a passenger in airliners. The key **K** allows to visualize the airplane trajectory.

To cause an accident with the B-52 do this:

- Make a steep turn with a very strong bank; the wings nearly perpendicular to the ground.
- Try to get the plane back level. It will obey but very slowly. You will get aware that the turn will go on for a while and that you will turn further than your intended flight direction.
- Do something that accelerates the stabilization on some airplanes: push the rudder to an extreme, opposite to the current turn. This will suddenly make the airplane drop from the sky.

Chapter 8

A Cross Country Flight Tutorial

8.1 Introduction

Figure 8.1: Flying over the San Antonio Dam to Livermore

This tutorial simulates a cross-country flight from Reid-Hillview (KRHV) to Livermore (KLVK) under Visual Flight Rules (VFR). Both airports are included in the standard FlightGear package, so no additional scenery is required.

I'll assume that you are happy taking off, climbing, turning, descending and landing in FlightGear. If not, have alook at the tutorials listed above. This tutorial is designed to follow on from them and provide information on some of the slightly more complicated flight systems and procedures.

8.1.1 Disclaimer and Thanks

A quick dislaimer. I'm not a pilot. Most of this information has been gleaned from various non-authoritive sources. If you find an error or misunderstanding, please let me know. Mail me at stuart_d_buchanan -at- yahoo.co.uk.

I'd like to thank the following people for helping make this tutorial accurate and readable. Benno Schulenberg, Sid Boyce. Vassilii Khachaturov, James Briggs.

8.2 Flight Planning

Before we begin, we need to plan our flight. Otherwise we'll be taking off not knowing whether to turn left or right.

First, have a look at the Sectional for the area. This is a map for flying showing airports, navigational aids, and obstructions. There are two scales of sectionals for VFR flight - the 1:500,000 sectionals themselves, and a number of 1:250,000 VFR Terminal Area Charts which cover particularly busy areas.

They are available from pilot shops, or on the web from various sources. You can access a Google-map style interface here:

http://www.runwayfinder.com/

Simple search for Reid-Hillview. An extract from the chart is shown in Figure 8.2.

If you want a map of the entire area showing exactly where the plane is, you can use Atlas. This is a moving-map program that connects to FlightGear. See Section 5.2 for details.

So, how are we going to fly from Reid-Hillview to Livermore?

We'll be taking off from runway 31R at KRHV. KRHV is the ICAO code for Reid-Hillview airport, and is shown in the FlightGear wizard. (It is marked on the sectional as RHV for historic reasons. To get the ICAO code, simply prefix a 'K'.)

The 31 indicates that the magnetic heading of the runway is around 310 degrees, and the R indicates that it's the runway on the right. As can be seen from the sectional, there are two parallel runways at KRHV. This is to handle the large amount of traffic that uses the airport. Each of the runways can be used in either direction. Runway 31 can be used from the other end as runway 13. So, the runways available are 13R, 13L, 31R, 31L. Taking off and landing is easier done into the wind, so when the wind is coming from the North West, runways 31L and 31L will be in use. The name of the runway is written in large letters at the beginning and is easily seen from the air.

Once we take off we'll head at 350 degrees magnetic towards Livermore (KLVK). We'll fly at about 3,500ft about sea-level. This puts us at least 500ft above any terrain or obstructions like radio masts on the way.

We'll fly over the Calaveras Reservoir then the San Antonio Reservoir. These are both large bodies of water and we can use them as navigation aids to ensure we

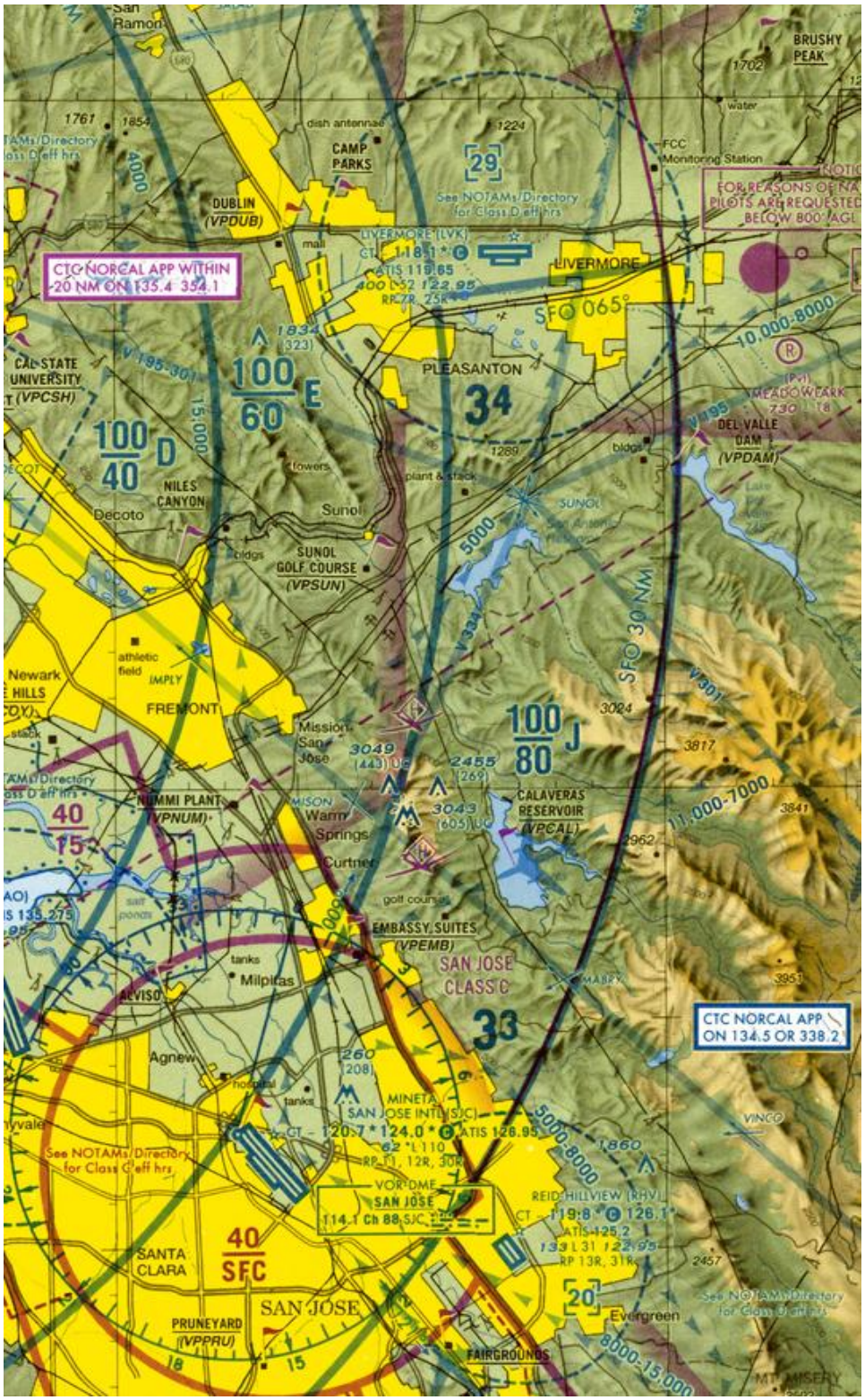


Figure 8.2: Sectional extract showing Reid-Hillview and Livermore airports

stay on the right track.

Once we get about 10 miles out of Livermore (above the San Antonia Reservoir), we'll contact the Livermore Air Traffic Control (ATC) to find out where we should land. We'll then join the circuit and land.

8.3 Getting Up

OK, we know where we're going and how we'll get there. Time to get started.

Start FlightGear using the Wizard (or command-line if you prefer). We want to use a C172P and take off from runway 31R at Reid-Hillview of Santa Clara County (KRHV). Dawn is a nice time to fly in California.

If you want, you can fly in the current weather at KRHV by clicking the Advanced button on the final screen of the Wizard, selecting Weather from the left-hand pane, selecting 'Fetch real weather' and clicking OK.

Figure 8.3: On the runway at KRHV

8.3.1 Pre-Flight

Before we take off, we need to pre-flight the aircraft. In the real world, this consists of walking around the aircraft to check nothing has fallen off, and checking we have enough fuel.

In our case, we'll take the opportunity to check the weather, set our altimeter and pre-set things that are easier to do when you're not flying.

The weather is obviously important when flying. We need to know if there is any sort of cross-wind that might affect take-off, at what altitude any clouds are (this is a VFR flight - so we need to stay well away from clouds at all times), and any wind that might blow us off course.

We also need to calibrate our altimeter. Altimeters calculate the current alititude indirectly by measuring air pressure, which decreases as you ascend. However, weather systems can affect the air pressure and lead to incorrect altimeter readings, which can be deadly if flying in mountains.

8.3.2 ATIS

Conveniently, airports broadcast the current sea-level pressure along with useful weather and airport information over the ATIS. This is a recorded message that is broadcast over the radio. However, to listen to it, we need to tune the radio to the correct frequency.

The ATIS frequency is displayed on the sectional (look for 'ATIS' near the airport), but is also available from within FlightGear. To find out the frequencies for an airport (including the tower, ground and approach if appropriate), use the ATC/AI menu and select Frequencies. Then enter the ICAO code (KRHV) into the dialog box. The various frequencies associated with the airport are then displayed. Duplicates indicate that the airport uses multiple frequencies for that task, and you may use either.

Either way, the ATIS frequency for Reid-Hillview is 125.2MHz.

8.3.3 Radios

We now need to tune the radio. The radio is located in the Radio Stack to the right of the main instruments. There are actually two independent radio systems, 1 and 2. Each radio is split in two, with a communications (COMM) radio on the left, and a navigation (NAV) radio on the right. We want to tune COMM1 to the ATIS frequency.

Figure 8.4: The C172 communications stack with COMM1 highlighted

The radio has two frequencies, the active frequency, which is currently in use,

and the standby frequency, which we tune to the frequency we wish to use next. The active frequency is shown on the left 5 digits, while the standby frequency is shown on the right. We change the standby frequency, then swap the two over, so the standby becomes active and the active standby. This way, we don't lose radio contact while tuning the radio.

Figure 8.5: COMM1 adjustment knob

To change the frequency, click on the grey knob below the standby frequency (highlighted in Figure 8.5), just to the right of the 'STBY'. Using the left mouse button changes the number after the decimal place, using the middle button changes the numbers before the decimal place. Click on the right side of the button to change the frequency up, and the left of the button to change the frequency down. Most of the FlightGear cockpit controls work this way. If you are having difficulty clicking on the correct place, press Ctrl-C to highlight the hot-spots for clicking.

Figure 8.6: COMM1 switch

Once you have changed the frequency to 125.2, press the white button between the words 'COMM' and 'STBY' to swap the active and standby frequencies (highlighted in Figure 8.6). After a second or so, you'll hear the ATIS information.

8.3.4 Altimeter and Compass

Figure 8.7: Altimeter calibration knob

Listen for the 'Altimeter' setting. If you are not using 'real weather', the value will be 2992, which is standard and already set on the plane. If you are using 'real weather', then the altimeter value is likely to be different. We therefore need to set the altimeter to the correct value. To do this, use the knob at the bottom left of the altimeter (circled in red in Figure 8.7), in the same way as you changed the radio frequency. This changes the value in the little window on the right of the altimeter, which is what you are trying to set, as well as the altitude displayed by the altimeter.

The other way to set the altimeter is to match it to the elevation above sea-level of the airport. The elevation is listed on the sectional. For KRHV it is 133ft. This means you can double-check the pressure value reported over ATIS.

Figure 8.8: Heading adjust knob

We will also take the opportunity to set the heading bug on the compass to 350 - our bearing from KRHV to KLVK. To do this, use the the red button on the compass housing (highlighted in Figure 8.8), just as you've done before. Use the left mouse button for small adjustments, and middle mouse button to make

big adjustments. The value of 350 is just anti-clockwise of the labeled value of N (North - 0 degrees).

Figure 8.9: Take-off from KRHV

8.3.5 Take-Off

OK, now we've done that we can actually take off!. In my case this usually involves weaving all over the runway, and swerving to the left once I've actually left the ground, but you'll probably have better control than me. Once above 1000ft, make a gentle turn to the right to a heading of 350 degrees. As we've set the heading bug, it will be easy to follow. We're aiming for a fairly prominent valley.

Continue climbing to 3,500 ft at around 500-700 fpm. Once you reach that altitude, reduce power, level off to level flight and trim appropriately. Check the power again and adjust so it's in the green arc of the RPM guage. We shouldn't run the engine at maximum RPM except during take-off.

8.4 Cruising

OK, we've taken off and are on our way to Livermore. Now we can make our life a bit easier by using the autopilot and our plane more fuel efficient by tuning the engine. We'll also want to check we're on-course

8.4.1 The Autopilot

We can make our life a bit easier by handing some control of the aircraft over to 'George' - the autopilot.

The autopilot panel is located towards the bottom of the radio stack (highlighted in Figure 8.10). It is easily distinguishable as it has many more buttons than

Figure 8.10: The C172 Autopilot

the other components on the stack. It can work in a number of different modes, but we are only interested in one of them for this flight - HDG. As the names suggest, HDG will cause the autopilot to follow the heading bug on the compass, which we set earlier.

To set the autopilot, press the AP button to switch the autopilot on, then press the HDG button to activate heading mode. While the autopilot is switched on, it will use the trim controls to keep the plane on the heading. You can change the heading bug, and the autopilot will maneuver appropriately. However, the autopilot doesn't make any allowances for wind speed or direction, it only sets the heading of the airplane. If flying in a cross-wind, the plane may be pointed in one direction, but be travelling in quite another.

You should use the trim controls to keep a level flight. You can use the autopilot for this, but it is a bit more complicated.

Once the aircraft has settled down under the autopilot's control, we can pay more attention to the outside world and higher level tasks.

8.4.2 Navigation

As we noted above, we're going to be travelling over a couple of reservoirs. When you leveled off, the first (Calaveras) was probably right in front of you. You can use them to check your position on the map. If it looks like you're heading off course, twist the heading bug to compensate.

8.4.3 Mixture

As altitude increases, the air gets thinner and contains less oxygen. This means that less fuel can be burnt each engine cycle. The engine in the C172 is simple and doesn't automatically adjust the amount of fuel to compensate for this lack of

Figure 8.11: The Calaveras Reservoir

Figure 8.12: The Calaveras Reservoir

oxygen. This results in an inefficient fuel burn and a reduction in power because the fuel-air mixture is too 'rich'. We can control the amount of fuel entering the engine every cycle using the mixture control. This is the red lever next to the throttle. By pulling it out, we 'lean' the mixture. We don't want the mixture too rich, nor too lean. Both these conditions don't produce as much power as we'd like. Nor do we want it perfect, because this causes the fuel-air to explode, rather than burn in a controlled manner, which is a quick way to trash an engine.

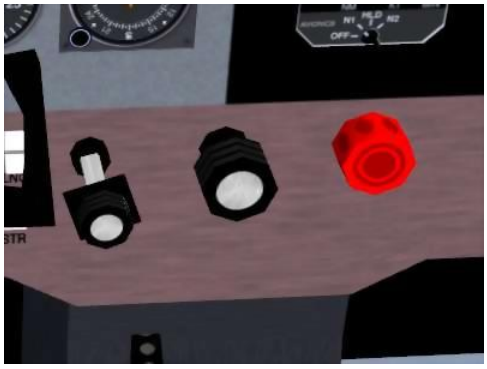


Figure 8.13: Mixture Control

The mixture is controlled by the red lever to the right of the yoke. You may need to pan your cockpit view to see it.

To pan the cockpit view, right-click with the mouse-button within the FlightGear window until the cursor becomes a double-headed arrow. Moving the mouse now pans the view. Once you can see the mixture lever clearly, right-click again to change the mouse back to the normal arrow.

Figure 8.14: Fuel Flow and EGT guages

Pull the mixture lever out slowly (use Ctrl-C to see the hot spots), leaning the mixture. As you do so, you'll see various engine instruments (on the left of the panel) change. Fuel flow will go down (we're burning less fuel), EGT (Exhaust Gas Temperature) will go up (we're getting closer to a 'perfect mixture') and RPM

will increase (we're producing more power). Pull the mixture lever out until you see the EGT go off the scale, then push it in a bit. We're now running slightly rich of peak. While at 3,500ft we don't need to lean much, at higher altitudes leaning the engine is critical for performance.

8.5 Getting Down

Once you reach the second reservoir (the San Antonio Reservoir), we need to start planning our descent and landing at Livermore. Landing is a lot more complicated than taking off, assuming you want to get down in one piece, so you may want to pause the simulator (press 'p') while reading this.

8.5.1 Air Traffic Control

In the Real World, we'd have been in contact with Air Traffic Control (ATC) continually, as the bay area is quite congested in the air as well as on the ground. ATC would probably provide us with a 'flight following' service, and would continually warn us about planes around us, helping to avoid any possible collisions. The FlightGear skies are generally clear of traffic, so we don't need a flight following service. If you want to change the amount of traffic in the sky, you can do so from the AI menu.

Livermore Airport is Towered (towered airports are drawn in blue on the sectional), so we will need to communicate with the tower to receive instructions on how and where to land.

Before that, we should listen to the ATIS, and re-adjust our altimeter, just in case anything has changed. This is quite unlikely on such a short flight, but if flying hundreds of milesm it might make a difference. To save time when tuning radios, you can access the Radio Settings dialog from the Equipment menu. The Livermore ATIS frequency is 119.65MHz.

An ATIS message also has a phonetic letter (Alpha, Bravo, ... Zulu) to identify the message. This phonetic is changed each time the recorded message is updated. When first contacting a tower, the pilot mentions the identifier, so the tower can double-check the pilot has up to date information.

Besides the altitude and weather information, the ATIS will also say which runway is in use. This is useful for planning our landing. Normally, due to the prevalent Westerly wind, Livermore has runways 25R and 25L in use.

Once you've got the ATIS, tune the radio to Livermore Tower. The frequency is 118.1MHz. Depending on the level of AI traffic you have configured on your system, you may hear Livermore Tower talking to other aircraft that are landing or

departing. This information is not played over the speakers, it is only displayed on the screen.

Once the frequency goes quiet, press the ' key. This will bring up the ATC menu. Click on the radio button on the left to select what you wish to say (you only have one option), then OK.

Your transmission will be displayed at the top of the screen. It will indicate who you are (type and tail number), where you are (e.g. 6 miles south), that you are landing, and the ATIS you have.

After a couple of seconds, Livermore Tower will respond, addressing you by name and telling you what runway to use, which pattern is in use and when to contact them, for example

> "Golf Foxtrot Sierra, Livermore Tower, Report left downwind runway two five left."

To understand what this means, we'll have to describe the Traffic Pattern.

8.5.2 The Traffic Pattern

With the number of aircraft flying around, there have to be standard procedures for take-off and landing, otherwise someone might try to land on-top of an aircraft taking off.

The Traffic Pattern is a standard route all aircraft must follow when near an airport, either taking off or landing. The traffic pattern has four stages (or 'legs'), shown in Figure 8.15. The 'downwind' mentioned above refers to one of these, the one with the number 3.

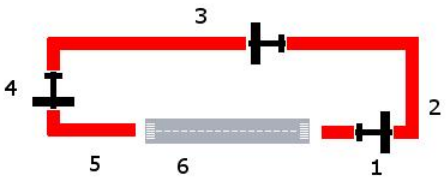


Figure 8.15: The Traffic Pattern

1. Aircraft take off from the runway and climb. If they are leaving the airport, they just continue climbing straight ahead until clear of the pattern and then do whatever they like. If they are returning to the runway (for example to practise landing), they continue climbing until they reach a couple of hundred feet below 'pattern altitude'. This varies from country to country, but is

usually between 500ft and 1000ft Above Ground Level (AGL). This is called the *upwind* leg.

2. The pilot makes a 90 degree left-hand turn onto the *crosswind* leg. They continue their climb to 'pattern altitude' and level out.

3. After about 45 seconds to a minute on the crosswind leg, the pilot again makes a 90 degree left turn onto the *downwind* leg. Aircraft arriving from other airports join the pattern at this point, approaching from a 45 degree angle away from the runway.

4. When a mile or so past the end of the runway (a good guide is when the runway is 45 degrees behind you), the pilot turns 90 degrees again onto the *base* leg and begins the descent to the runway, dropping flaps as appropriate. A descent rate of about 500fpm is good.

5. After about 45 seconds the pilot turns again onto the *final* leg. It can be hard to estimate exactly when to perform this turn. Final adjustments for landing are made. I usually have to make small turns to align with the runway properly.

6. The aircraft lands. If the pilot is practising take-offs and landings, full power can be applied and flaps retracted for takeoff, and the aircraft can take off once more. This is known as 'touch-and-go'.

Most patterns at left-handed, i.e. all turns are to the left, as described above. Right-hand patterns also exist, and are marked as 'RP' on the sectional. ATC will also advise you what pattern is in use.

8.5.3 Approach

We're approaching Livermore airport from the South, while the runways run East/West. Due to the prevailing Westerly wind, we'll usually be directed to either runway 25R or 25L. 25R uses a right-hand pattern, while 25L uses a left-hand pattern. Both the patterns are illustrated in Figure 8.16. Depending on the runway we've been assigned, we'll approach the airport in one of two ways. If we've been asked to land on runway 25R, we'll follow the blue line in the diagram. If we've been asked to land on runway 25L, we'll follow the green line.

We also need to reduce our altitude. We want to end up joining the pattern at pattern altitude, about 1,000ft above ground level (AGL). Livermore airport is at 400 ft above sea-level (ASL), so we need to descend to an altitude of 1400 ASL.

We want to begin our maneuvers well before we reach the airport. Otherwise we're likely to arrive too high, too fast, and probably coming from the wrong direction. Not the best start for a perfect landing :).

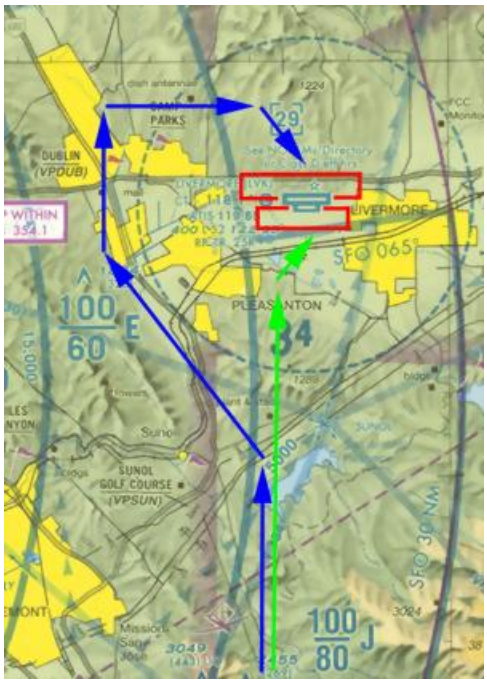


Figure 8.16: Sectional extract showing approaches to Livermore

So, let‘s start descending immediately.

1. First switch off the autopilot by pressing the AP switch.

2. Return mixture to fully rich (pushed right in). If we were landing at a high airport, we’d just enrich the mixture slightly and re-adjust when we reached the pattern.

3. Apply carb-heat. This stops ice forming when the fuel and air mix before entering the cylinder, something that can often happen during descent in humid air. The carb-heat lever is located between the throttle and mixture. Pull it out to apply heat.

4. Reduce power quite a bit. Otherwise we might stress the airframe due to over-speeding.

5. Drop the nose slightly to start the descent.

6. Trim the aircraft.

Use your location relative to the airport and the two towns of Pleasanton and Livermore to navigate yourself to the pattern following the general guide above.

Once you're established on the downwind leg, you'll need to report to ATC again. Do this in the same way as before. They will then tell you where you are in the queue to land. 'Number 1' means there are no planes ahead of you, while 'Number 9' means you might want to go to a less busy airport! They'll also tell you who is ahead of you and where. For example 'Number 2 for landing, follow the Cessna on short final' means that there is a single aircraft in front of you that is currently on the final leg of the pattern. When they land and are clear of the runway, they'll tell ATC, who can then tell you 'Number 1 for landing'.

8.5.4 VASI

Figure 8.17: On Final at Livermore with VASI on the left

Once on final, you'll notice two sets of lights on the left of the runway (enhanced in Figure 8.17). This is the VASI and provides a nice visual clue as to whether you're too low or too high on approach. Each set of lights can either be white or red. White means too high, red means too low. White and red together means just perfect. On a Cessna approaching at 60kts, a descent rate of about 500fpm should be fine. If you are too high, just decrease power to increase your descent rate to 700fpm. If you are too low, increase power to decrease your descent rate to 200fpm.

8.5.5 Go Around

If for some reason it looks like you're going to mess up the landing you can abort the landing and try again. This is called a 'Go Around'. To do this

1. Apply full power
2. Wait until you have a positive rate of climb - i.e. your altitude is increasing according to the altimeter.
3. Raise your flaps to 10 degrees (first-stage).

Figure 8.18: Missed approach at Livermore

4. Tell ATC you are ‘going around’
5. Climb to pattern height
6. If you aborted on final approach, continue over the runway to re-join the pattern on the crosswind leg. If on base, fly past the turn for final, then turn and fly parallel to the runway on the opposite side from downwind to rejoin on the crosswind leg.
7. Fly the complete pattern, telling ATC when you are on downwind, and try again.

8.5.6 Clearing the Runway

Figure 8.19: Landing at Livermore

Once you’re on the ground, you should taxi off the runway, then tell ATC you are clear. At high-altitude airports, you would lean the engine to avoid fouling the spark-plugs with an over-rich mixture. Find somewhere nice to park, shut down the engine by pulling mixture to full lean, then throttle off and magnetos to off (knob on the bottom left of the panel). Switch off the avionics master switch, tie down the aircraft, then go get that hamburger!

I hope this tutorial is of some use. If you have any comments, please let me know at stuart_d_buchanan {at} yahoo.co.uk.

Part IV

Appendices

Appendix A

Missed approach: If anything refuses to work

In the following section, we tried to sort some problems according to operating system, but if you encounter a problem, it may be a wise idea to look beyond "your" operating system – just in case. If you are experiencing problems, we would strongly advise you to first check the FAQ maintained by Cameron Moore at

http://www.flightgear.org/Docs/FlightGear-FAQ.html.

Moreover, the source code contains a directory `docs-mini` containing numerous ideas on and solutions to special problems. This is also a good place to go for further reading.

A.1 FlightGear Problem Reports

The best place to look for help is generally the mailing lists, specifically the **[Flightgear-User]** mailing list. If you happen to be running a CVS version of ***FlightGear***, you may want to subscribe to the **[Flightgear-Devel]** list. Instructions for subscription can be found at

http://www.flightgear.org/mail.html.

It's often the case that someone has already dealt with the issue you're dealing with, so it may be worth your time to search the mailing list archives at

http://www.mail-archive.com/flightgear-users%40flightgear.org/

http://www.mail-archive.com/flightgear-devel%40flightgear.org/.

There are numerous developers and users reading the lists, so questions are generally answered. However, messages of the type

FlightGear does not compile on my system. What shall I do?

are hard to answer without any further detail given, aren't they? Here are some things to consider including in your message when you report a problem:

- **Operating system:** (Linux Redhat 7.0. . . /Windows 98SE. . .)
- **Computer:** (Pentium III, 1GHz. . .)
- **Graphics board/chip:** (Diamond Viper 770/NVIDIA RIVA TNT2. . .)
- **Compiler/version:** (Cygnus version 1.0. . .)
- **Versions of relevant libraries:** (PLIB 1.2.0, Mesa 3.0. . .)
- **Type of problem:** (Linker dies with message. . .)
- **Steps to recreate the problem:** Start at KSFO, turn off brakes . . .

For getting a trace of the output which ***FlightGear*** produces, then following command may come in handy (may need to be modified on some OSs or may not work on others at all, though):

```
%FG_ROOT/BIN/fgfs >log.txt 2>&1
```

One final remark: Please avoid posting binaries to these lists! List subscribers are widely distributed, and some users have low bandwidth and/or metered connections. Large messages may be rejected by the mailing list administrator. Thanks.

A.2 General problems

- ***FlightGear*** runs SOOO slow.
  If ***FlightGear*** says it's running with something like 1 fps (frame per second) or below you typically don't have working hardware OpenGL support. There may be several reasons for this. First, there may be no OpenGL hardware drivers available for older cards. In this case it is highly recommended to get a new board.

  Second, check if your drivers are properly installed. Several cards need additional OpenGL support drivers besides the "native" windows ones. For more detail check Appendix C.

- Either `configure` or `make` dies with not found ***PLIB*** headers or libraries. Make sure you have the latest version of ***PLIB*** (> version 1.8.4) compiled and installed. Its headers like `pu.h` have to be under `/usr/include/plib` and its libraries, like `libplibpu.a` should be under `/lib`. Double check there are no stray ***PLIB*** headers/libraries sitting elsewhere!

  Besides check careful the error messages of `configure`. In several cases it says what is missing.

A.3 Potential problems under Linux

Since we don't have access to all possible flavors of Linux distributions, here are some thoughts on possible causes of problems. (This Section includes contributions by Kai Troester.)

- Wrong library versions
  This is a rather common cause of grief especially when you prefer to install the libraries needed by ***FlightGear*** by hand. Be sure that especially the Mesa library contains support for the 3DFX board and that GLIDE libraries are installed and can be found. If a `ldd ` ``which fgfs`` complains about missing libraries you are in trouble.

  You should also be sure to *always* keep the *latest* version of ***PLIB*** on your system. Lots of people have failed miserably to compile ***FlightGear*** just because of an outdated plib.

- Missing permissions
  In case you are using XFree86 before release 4.0 the ***FlightGear*** binary may need to be setuid root in order to be capable of accessing some accelerator boards (or a special kernel module as described earlier in this document) based on 3DFX chips. So you can either issue a

  ```
  chown root.root /usr/local/bin/fgfs ;
  chmod 4755 /usr/local/bin/fgfs
  ```

  to give the ***FlightGear*** binary the proper rights or install the 3DFX module. The latter is the "clean" solution and strongly recommended!

- Non-default install options
  FlightGear will display a lot of diagnostics while starting up. If it complains about bad looking or missing files, check that you installed them in the way they are supposed to be installed (i.ewith the latest version and in the proper location). The canonical location ***FlightGear*** wants its data files under `/usr/local/lib`. Be sure to grab the latest versions of everything that might be needed!

- Compile problems in general
  Make sure you have the latest (official) version of gcc. Old versions of gcc are a frequent source of trouble! On the other hand, some versions of the RedHat 7.0 reportedly have certain problems compiling ***FlightGear*** as they include a preliminary version of GCC.

A.4 Potential problems under Windows

- The executable refuses to run.
  You may have tried to start the executable directly either by double-clicking `fgfs.exe` in Windows Explorer or by invoking it within a MS-DOS shell. Double-clicking via Explorer never works (unless you set the environment variable `FG_ROOT` in `autoexec.bat` or otherwise). Rather double-click `fgrun`. For more details, check Chapter 3.

  Another cause of grief might be that you did not download the most recent versions of the base package files required by ***FlightGear***, or you did not download any of them at all. Have a close look at this, as the scenery/texture format is still under development and may change frequently. For more details, check Chapter 2.

  Next, if you run into trouble at runtime, do not use windows utilities for unpacking the `.tar.gz`. If you did, try it in the Cygnus shell with `tar xvfz` instead.

- ***FlightGear*** ignores the command line parameters.
  There is a problem with passing command line options containing a ”=” on the command line. Instead create a batch job to include your options and run that instead.

- I am unable to build ***FlightGear*** under MSVC/MS DevStudio.
  By default, ***FlightGear*** is build with GNU GCCṪhe Win32 port of GNU GCC is known as CygwinḞor hints on Makefiles required for MSVC for MSC DevStudio have a look into

  ftp://www.flightgear.org/pub/flightgear/Source/.

  In principle, it should be possible to compile ***FlightGear*** with the project files provided with the source code.

- Compilation of ***FlightGear*** dies.
  There may be several reasons for this, including true bugs. However, before trying to do anything else or report a problem, make sure you have the latest version of the ***Cygwin*** compiler, as described in Section B. In case of doubt, start `setup.exe` anew and download and install the most recent versions of bundles as they possibly may have changed.

Appendix B

Building the plane: Compiling the program

This appendix describes how to build ***FlightGear*** on several systems. In case you are on a Win32 (i. e. Windows95/98/ME/NT/2000/XP) platform or any of the other platforms for which binary executables are available, you may not want to go through that potentially troublesome process but skip this section for now and go back to the chapter on installing ***FlightGear*** one. (Not everyone wants to build his or her plane himself or herself, right?) However, there may be good reason for at least trying to build the simulator:

- In case you are on a UNIX/Linux platform there may be no pre-compiled binaries available for your system. In practice it is common to install programs like this one on UNIX systems by recompiling them.
- There are several options you can set during compile time only.
- You may be proud you did.

On the other hand, compiling ***FlightGear*** is not a task for novice users. Thus, if you're a beginner (we all were once) on a platform which binaries are available for, we recommend postponing this task and just starting with the binary distribution to get you flying.

As you will notice, this Chapter is far from being complete. Basically, we describe compiling for two operating systems only, Windows and Linux, and for only one compiler, the GNU C compiler. ***FlightGear*** has been shown to be built under different compilers (including Microsoft Visual C) as well as different systems (Macintosh) as well. The reason for these limitations are:

- Personally, we have access to a Windows machine running the Cygnus compiler only.

- According to the mailing lists, these seem to be the systems with the largest user base.
- These are the simplest systems to compile ***FlightGear*** on. Other compilers may need special add-ons (workplace etc.) or even modification of the code.
- The GNU compiler is free in the same sense of the GPL as ***FlightGear*** is.

You might want to check Section A, *Missed approach*, if anything fails during compilation. In case this does not help we recommend sending a note to one of the mailing lists (for hints on subscription see Chapter D).

There are several Linux distributions on the market, and most of them should work. Some come even bundled with (often outdated) versions of ***FlightGear***. However, if you are going to download or buy a distribution, Debian (Sarge) is highly recommended by most people. SuSE and Ubuntu works well, too.

Contrary to Linux/Unix systems, Windows usually comes without any development tools. This way, you first have to install a development environment. On Windows, in a sense, before building the plane you will have to build the plant for building planes. This will be the topic of the following section, which can be omitted by Linux users.

B.1 Preparing the development environment under Windows

There is a powerful development environment available for Windows and this even for free: The Cygnus development tools, resp. ***Cygwin***. Their home is at

http://sources.redhat.com/cygwin/,

and it is always a good idea to check back what is going on there now and then.

Nowadays, installing ***Cygwin*** is nearly automatic. First, make sure the drive you want ***Cygwin***, ***OpenSceneGraph***, ***PLIB***, ***SimGear*** and ***FlightGear*** to live on, has nearly 1 GB of free disk space. Create a temporary directory and download the installer from the site named above to that directory. (While the installer does an automatic installation of the Cygnus environment, it is a good idea to download a new installer from time to time.)

Invoke the installer now. It gives you three options. To avoid having to download stuff twice in case of a re-installation or installation on a second machine, we highly recommended to take a two-step procedure. First, select the option `Download from Internet`. Insert the path of your temporary directory, your Internet connection settings and then choose a mirror form the list. Near servers might be preferred, but may be sometimes a bit behind with mirroring. We found

ftp://mirrors.rcn.net

a very recent and fast choice. In the next windows the default settings are usually a good start. Now choose `Next`, sit back and wait.

If you are done, invoke the installer another time, now with the option `Install from local directory`. After confirming the temporary directory you can select a root directory (acting as the root directory of your pseudo UNIX file system). Cygnus does not recommend taking the actual root directory of a drive, thus choose `c:/Cygwin` (while other drives than `c:` work as well). Now, all ***Cygwin*** stuff and all ***FlightGear*** stuff lives under this directory. In addition, select

```
Default text file type:  Unix
```

In addition, you have the choice to install the compiler for all users or just for you.

The final window before installation gives you a selection of packages to install. It is hard, to provide a minimum selection of packages required for ***FlightGear*** and the accompanying libraries to install. We have observed the following (non minimum) combination to work:

- `Admin` skip
- `Archive` install
- `Base` install
- `Database` skip
- `Devel` install
- `Doc` install
- `Editors` skip
- `Graphics` install
- `Interpreters` install
- `Libs` install
- `Mail` skip
- `Net` skip
- `Shells` install
- `Text` install
- `Utils` install

- `Web` skip
- `XFree86` do not install!

Note XFree86 must be not installed for building ***FlightGear*** and the accompanying libraries. If it is installed you have to deinstall it first. Otherwise ***FlightGear***'s configuration scripts will detect the XFree86 OpenGL libraries and link to them, while the code is not prepared to do so.

As a final step you should include the binary directory (for instance: `c:/Cygwin/bin`) into your path by adding `path=c:\Cygwin\bin` in your `autoexec.bat` under Windows 95/98/MEUnder WindowsNT/2000/XP, use the `Extended` tab under the `System properties` page in Windows `control panel`. There you'll find a button `Environment variables`, where you can add the named directory.

Now you are done. Fortunately, all this is required only once. At this point you have a nearly UNIX-like (command line) development environment. Because of this, the following steps are nearly identical under Windows and Linux/Unix.

B.2 Preparing the development environment under Linux

Linux, like any UNIX, usually comes with a compiler pre-installed. On the other hand, you still have to make sure several required libraries are present.

First, make sure you have all necessary OpenGL libraries. Fortunately, most of the recent Linux distributions (i.e. SuSE-7.3) put these already into the right place. (There have been reports, though, that on Slackware you may have to copy the libraries to `/usr/local/lib` and the headers to `/usr/local/include` by hand after building `glut-3.7`). Be sure to install the proper packages: Besides the basic X11 stuff you want to have - SuSE as an example - the following packages: mesa, mesa-devel, mesasoft, xf86_glx, xf86glu, xf86glu-devel, mesaglut, mesaglut-devel and plib.

Also you are expected to have a bunch of tools installed that are usually required to compile the Linux kernel. So you may use the Linux kernel source package to determine the required dependencies. The following packages might prove to be useful when fiddling with the ***FlightGear*** sources: automake, autoconf, libtool, bison, flex and some more, that are not required to build a Linux kernel.

Please compare the release of the plib library with the one that ships with your Linux distribution. It might be the case that ***FlightGear*** requires a newer one that is not yet provided by your vendor.

B.3 One-time preparations for Linux and Windows users

There are a couple of 3rd party libraries which your Linux or Windows system may or may not have installed, i.e. the ***ZLIB*** library. You can either check your list of installed packages or just try building ***SimGear***: It should exit and spit an error message (observe this!) if one of these libraries is missing.

If you make this observation, install the missing libraries, which is only required once (unless you re-install your development environment).

Both libraries come bundled with ***SimGear***, which links to them, but does not automatically install them. For installing either of them, get the most recent file `SimGear-X.X.X.tar.gz` from

http://www.simgear.org/downloads.html

Download it to `/usr/local/source`. Change to that directory and unpack ***SimGear*** using

`tar xvfz SimGear-X.X.X.tar.gz`.

You will observe a directory `src-libs` which contains the two names libraries.

B.3.1 Installation of *ZLIB*

`cd` into `SimGear-X.X.X/scr-libs` and unpack ***ZLIB*** using

`tar xvfz zlib-X.X.X.tar.gz`.

Next, change to the newly created directory `zlib-X.X.X` and type

```
./configure
make
make install
```

Under Linux, you have to become root for being able to `make install`, for instance via the `su` command.

You may want to consult the Readme files under `SimGear-X.X.X/scr-libs` in case you run into trouble.

B.4 Compiling *FlightGear* under Linux/Windows

The following steps are identical under Linux/Unix and under Windows with minor modifications. Under Windows, just open the ***Cygwin*** icon from the Start menu or

from the desktop to get a command line.

To begin with, the ***FlightGear*** build process is based on five packages which you need to built and installed in this order:

- OSG
- PLIB
- SimGear
- FlightGear, program
- FlightGear, base (data - no compilation required)

1. First, choose an install directory for FlightGear. This will not be the one your binaries will live in but the one for your source code and compilation files. We suggest

   `cd:/usr/local/`

   `mkdir source`

2. Now, you have to install three support libraries, ***OpenAL***, ***OpenSceneGraph*** and ***PLIB*** which are absolutely essential for the building process. ***OpenSceneGraph*** contains most of the basic graphics rendering, ***OpenAL*** is for audio and ***PLIB*** contains keyboard and joystick routines. Download the latest stable versions of these libraries from

   http://www.openscenegraph.org/

   http://www.openal.org/

   http://plib.sourceforge.net/

   to `/usr/local/source`. Change to that directory and unpack ***PLIB*** using

   `tar xvfz plib-X.X.X.tar.gz`.

   `cd` into `plib-X.X.X` and run

   ```
   ./configure
   make
   make install.
   ```

   Under Linux, you have to become root for being able to `make install`, for instance via the `su` command.

   Confirm you now have ***PLIB***'s header files (as `ssg.h` etc.) under `/usr/include/plib` (and nowhere else).

3. Next, you have to install another library ***SimGear*** containing the basic simulation routines. Get the most recent file `SimGear-X.X.X.tar.gz` from

   http://www.simgear.org/downloads.html

   Download it to `/usr/local/source`. Change to that directory and unpack ***SimGear*** using

   `tar xvfz SimGear-X.X.X.tar.gz`.

   `cd` into `SimGear-X.X.X` and run

   ```
   ./configure
   make
   make install
   ```

   Again, under Linux, you have to become root for being able to `make install`, for instance via the `su` command.

4. Now, you're prepared to build ***FlightGear*** itself, finally. Get `FlightGear-X.X.X.tar.gz` from

   http://www.flightgear.org/Downloads/

   and download it to `/usr/local/source`. Unpack ***FlightGear*** using

   `tar xvfz FlightGear-X.X.X.tar.gz`.

   `cd` into `FlightGear-X.X.X` and run

   `./configure`

   configure knows about numerous options, with the more relevant ones to be specified via switches as

   - `--with-network-olk`: Include Oliver Delise's multi-pilot networking support,
   - `--with-new-environment`: Include new experimental environment subsystem,
   - `--with-weathercm`: Use WeatherCM instead of FGEnvironment,
   - `--with-plib=`PREFIX: Specify the prefix path to ***PLIB***,
   - `--with-simgear=`PREFIX: Specify the prefix path to ***SimGear***,
   - `--prefix=/XXX`: Install ***FlightGear*** in the directory XXX.
   - `--disable-jsbsim`: Disable ***JSBSim***m FDM (in case of trouble compiling it).
   - `--disable-yasim`: Disable ***YASim*** FDM (in case of trouble compiling it).

- `--disable-larcsim`: Disable ***LaRCsim*** FDM (in case of trouble compiling it).
- `--disable-uiuc`: Disable UIUC FDM (in case of trouble compiling it).

A good choice would be `--prefix=/usr/local/FlightGear`. In this case ***FlightGear***'s binaries will live under `/usr/local/FlightGear/bin`. (If you don't specify a `--prefix` the binaries will go into `/usr/local/bin` while the base package files are expected under `/usr/local/share/FlightGear`.)

Assuming `configure` finished successfully, run

```
make
make install.
```

Again, under Linux, you have to become root for being able to `make install`, for instance via the `su` command.

Note: You can save a significant amount of space by stripping all the debugging symbols off the executable. To do this, make a

```
cd /usr/local/FlightGear/bin
```

to the directory in the `install tree` where your binaries live and run

```
strip *.
```

This completes building the executable and should result in a file `fgfs` (Unix) or `fgfs.exe` (Windows) under `/usr/local/FlightGear/bin`

Note: If for whatever reason you want to re-build the simulator, use the command `make distclean` either in the `SimGear-X.X.X` or in the `FlightGear-X.X.X` directory to remove all the build. If you want to re-run `configure` (for instance because of having installed another version of ***PLIB*** etc.), remove the files `config.cache` from these same directories before.

B.5 Compiling *FlightGear* under Mac OS X

For compiling under Mac OS X you will need

- Mac X OS 10.1+ with developer tools installed.
- 500MB disk (minimum) free disk space.
- Fearlessness of command line compiling.

This will need a bit more bravery than building under Windows or Linux. First, there are less people who tested it under sometimes strange configurations. Second, the process as described here itself needs a touch more experience by using CVS repositories.

First, download the development files. They contain files that help simplify the build process, and software for automake, autoconf, and plib:

http://expert.cc.purdue.edu/~walisser/fg/fgdev.tar.gz

or

http://homepage.mac.com/walisser

Once you have this extracted, make sure you are using TCSH as your shell, since the setup script requires it.

Important for Jaguar users:

If you run Mac OS X 10.2 or later, gcc 3.1 is the default compiler. However, only version 2.95 works with ***FlightGear*** as of this writing. To change the default compiler, run this command (as root). You'll only have to do this once and it will have a global effect on the system.

```
sudo gcc_select 2
```

1. Setup the build environment:
   ```
   cd fgdev
   source bin/prepare.csh
   ```
2. Install the latest versions of the automake and autoconf build tools:
   ```
   cd $BUILDDIR/src/automake-X.X.X
   ./configure --prefix=$BUILDDIR
   make install
   rehash
   ```
   ```
   cd $BUILDDIR/src/autoconf-X.XX
   ./configure --prefix=$BUILDDIR
   make install
   rehash
   ```
3. Download PLIB
   ```
   cd $BUILDDIR/src
   setenv CVSROOT :pserver:anonymous@cvs.plib.sourceforge.net:/cvsroot/plib
   cvs login
   ```
   Press <enter> for password
   ```
   cvs -z3 checkout plib
   ```

4. Build PLIB
   ```
   cd $BUILDDIR/src/plib
   ./autogen.sh
   ./configure --prefix=$BUILDDIR
   make install
   ```

5. Get the ***SimGear*** sources
   ```
   cd $BUILDDIR/src
   setenv CVSROOT :pserver:cvsguest@cvs.simgear.org:/var/cvs/SimGear
   cvs login
   ```
   Enter <guest> for password
   ```
   cvs -z3 checkout SimGear
   ```

6. Build ***SimGear***
   ```
   cd $BUILDDIR/src/SimGear
   ./autogen.sh
   ./configure -prefix=$BUILDDIR
   make install
   ```

7. Get the ***FlightGear*** sources
   ```
   cd $BUILDDIR/src
   setenv CVSROOT :pserver:cvsguest@cvs.flightgear.org:/var/cvs/Flig
   cvs login
   ```
   Enter <guest> for password
   ```
   cvs -z3 checkout FlightGear
   ```

8. Build ***FlightGear***
   ```
   cd $BUILDDIR/src/FlightGear
   patch -p0 < ../jsb.diff
   ./autogen.sh
   ./configure -prefix=$BUILDDIR
   -with-threads -without-x
   ```
   (one line)
   ```
   make install
   ```

9. Get the base data files (if you don't have them already)
   ```
   cd $BUILDDIR
   setenv CVSROOT :pserver:cvsguest@cvs.flightgear.org:/var/cvs/Flig
   cvs login
   ```
   Password is "guest"
   ```
   cvs -z3 checkout data
   ```

10. Move data files (if you have them already)
    just make a symlink or copy data files to "fgfsbase" in $BUILDDIR
    alternatively adjust `--fg-root=xxx` parameter appropriately

11. Run FlightGear
```
cd $BUILDDIR
src/FlightGear/src/Main/fgfs
```

B.6 Compiling on other systems

Compiling on other UNIX systems - at least on IRIX and on

Solaris, is pretty similar to the procedure on Linux - given the presence of a working GNU C compiler. Especially IRIX and also recent releases of Solaris come with the basic OpenGL libraries. Unfortunately the "glut" libraries are mostly missing and have to be installed separately (see the introductory remark to this chapter). As compilation of the "glut" sources is not a trivial task to everyone, you might want to use a pre-built binary. Everything you need is a static library "libglut.a" and an include file "glut.h". An easy way to make them usable is to place them into `/usr/lib/` and `/usr/include/GL/`. In case you insist on building the library yourself, you might want to have a look at FreeGLUT

http://freeglut.sourceforge.net/

which should compile with minor tweaks. Necessary patches might be found in

ftp://ftp.uni-duisburg.de/X11/OpenGL/freeglut_portable.patch

Please note that you do **not** want to create 64 bit binaries in IRIX with GCC (even if your CPU is a R10/12/14k) because GCC produces a broken "fgfs" binary (in case the compiler doesn't stop with "internal compiler error"). Things look better since Eric Hofman managed to tweak the ***FlightGear*** sources for proper compiling with MIPSPro compiler.

There should be a workplace for Microsoft Visual C++ (MSVC6) included in the official ***FlightGear*** distribution. Macintosh users find the required CodeWarrior files as a `.bin` archive at

http://icdweb.cc.purdue.edu/~walisser/fg/.

Numerous (although outdated, at times) hints on compiling on different systems are included in the source code under `docs-mini`.

B.7 Installing the base package

If you succeeded in performing the steps named above, you will have a directory holding the executables for ***FlightGear***. This is not yet sufficient for performing ***FlightGear***, though. Besides those, you will need a collection of support data

files (scenery, aircraft, sound) collected in the so-called base package. In case you compiled the latest official release, the accompanying base package is available from

ftp://www.flightgear.org/pub/flightgear/Shared/fgfs-base-X.X.X.tar.gz.

This package is usually quite large (around 25 MB), but must be installed for ***FlightGear*** to run. There is no compilation required for it. Just download it to `/usr/local` and install it with

```
tar xvfz fgfs-base-X.X.X.tar.gz.
```

Now you should find all the ***FlightGear*** files under `/usr/local/Flightgear` in the following directory structure::

```
/usr/local/Flightgear
/usr/local/Flightgear/Aircraft
/usr/local/Flightgear/Aircraft-uiuc
...
/usr/local/Flightgear/bin
...
/usr/local/Flightgear/Weather.
```

B.8 For test pilots only: Building the CVS snapshots

It you are into adventures or feel you're an advanced user, you can try one of the recent bleeding edge snapshots at

http://www.flightgear.org/Downloads/.

In this case you have to get the most recent Snapshot from ***SimGear*** at

http://www.simgear.org/downloads.html

as well. But be prepared: These are for development and may (and often do) contain bugs.

If you are using these CVS snapshots, the base package named above will usually not be in sync with the recent code and you have to download the most recent developer's version from

http://rockfish.net/fg/.

We suggest downloading this package `fgfs_base-snap.X.X.X.tar.gz` to a temporary directory. Now, decompress it using

`tar xvfz fgfs_base-snap.X.X.X.tar.gz`.

Finally, double-check you got the directory structure named above.

Appendix C

Some words on OpenGL graphics drivers

FlightGear's graphics engine is based on a graphics library called OpenGL. Its primary advantage is its platform independence, i. e., programs written with OpenGL support can be compiled and executed on several platforms, given the proper drivers having been installed in advance. Thus, independent of if you want to run the binaries only or if you want to compile the program yourself you must have some sort of OpenGL support installed for your video card.

A good review on OpenGL drivers can be found at

http://www.flightgear.org/Hardware.

Specific information is collected for windows at

http://www.x-plane.com/SYSREQ/v5ibm.html

and for Macintosh at

http://www.x-plane.com/SYSREQ/v5mac.html.

An excellent place to look for documentation about Linux and 3-D accelerators is the*Linux Quake HOWTO* at

http://www.linuxquake.com.

This should be your first aid in case something goes wrong with your Linux 3-D setup.

Unfortunately, there are so many graphics boards, chips and drivers out there that we are unable to provide a complete description for all systems. Given the

present market dominance of NVIDIA combined with the fact that their chips have indeed been proven powerful for running ***FlightGear***, we will concentrate on NVIDIA drivers in what follows.

C.1 NVIDIA chip based cards under Linux

Recent Linux distributions include and install anything needed to run OpenGL programs under Linux. Usually there is no need to install anything else.

If for whatever reason this does not work, you may try to download the most recent drivers from the NVIDIA site at

http://www.nvidia.com/Products/Drivers.nsf/Linux.html

At present, this page has drivers for all NVIDIA chips for the following Linux distributions: RedHat 7.1, Redhat 7.0, Redhat 6.2, Redhat 6.1, Mandrake 7.1, Mandrake 7.2, SuSE 7.1, SuSE 7.0 in several formats (.rpm, .tar.gz). These drivers support OpenGL natively and do not need any additional stuff.

The page named above contains a detailed `README and Installation Guide` giving a step-by-step description, making it unnecessary to copy the material here. Please enshure to replace any OpenGL related libraries with those that are shipped with the NVIDIA driver - not only user space libraries but also those in the X server extension modules directory.

C.2 NVIDIA chip based cards under Windows

Again, you may first try the drivers coming with your graphics card. Usually they should include OpenGL support. If for whatever reason the maker of your board did not include this feature into the driver, you should install the Detonator reference drivers made by NVIDIA (which might be a good idea anyway). These are available in three different versions (Windows 95/98/ME, Windows 2000, Windows NT) from

http://www.nvidia.com/products.nsf/htmlmedia/detonator3.html

Just read carefully the Release notes to be found on that page. Notably do not forget to uninstall your present driver and install a standard VGA graphics adapter before switching to the new NVIDIA drivers first.

C.3 3DFX chip based cards under Windows

With the Glide drivers no longer provided by 3DFX there seems to be little chance to get it running (except to find older OpenGL drivers somewhere on the net or privately). All pages which formerly provided official support or instructions for 3DFX are gone now. For an alternative, you may want to check the next section, though.

C.4 An alternative approach for Windows users

There is now an attempt to build a program which detects the graphics chip on your board and automatically installs the appropriate OpenGL drivers. This is called OpenGL Setup and is presently in beta stage. It's home page can be found at

http://www.glsetup.com/.

We did not try this ourselves, but would suggest it for those completely lost.

C.5 3DFX chip based cards under Linux

Notably, with 3DFX now having been taken over by NVIDIA, manufacturer's support already has disappeared. However with XFree86-4.x (with x at least being greater than 1) Voodoo3 cards are known to be pretty usable in 16 bit color mode. Newer cards should work fine as well. If you are still running a version of Xfree86 3.X and run into problems, consider an upgrade. The recent distributions by Debian or SuSE have been reported to work well.

C.6 ATI chip based cards under Linux

There is support for ATI chips in XFree86-4.1 and greater. Lots of AGP boards based on the Rage128 chip - from simple Rage128 board to ATI Xpert2000 - are mostly usable for FlightGear. Since XFree86-4.1 you can use early Radeon chips - up to Radeon7500 with XFree86-4.2, up to Radeon9100 with XFree86-4.3. Be careful with stock XFree86-4.3.0, it was released with (known) bugs in the Radeon driver. Ongoing development provides functional drivers for R100 (Radeon7000 up to 7500) and R200 (Radeon8500 and 9100) chips.

ATI provides an alternative with their binary drivers. You need to build a kernel module using a script that is supplied with the package and add several X server modules into your XFree86 tree. In most cases, the RPM installation will do that for you.

C.7 Building your own OpenGL support under Linux

Setting up proper OpenGL support with a recent Linux distribution should be pretty simple. As an example SuSE ships everything you need plus some small shell scripts to adjust the missing bits automagically. If you just want to execute pre-built binaries of FlightGear, then you're done by using the supplied ***FlightGear*** package plus the mandantory runtime libraries (and kernel modules). The package manager will tell you which ones to choose.

In case you want to run a self-made kernel, you want to compile ***FlightGear*** yourself, you're tweaking your X server configuration file yourself or you even run a homebrewed Linux "distribution" (this means, you want to compile everything yourself), this chapter might be useful for you.

Now let's have a look at the parts that build OpenGL support on Linux. First there's a Linux kernel with support for your graphics adapter.

Examples on which graphics hardware is supported natively by Open Source drivers are provided on

http://dri.sourceforge.net/dri_status.phtml.

There are a few graphics chip families that are not directly or no more than partly supported by XFree86, the X window implementation on Linux, because vendors don't like to provide programming information on their chips. In these cases - notably IBM/DIAMOND/now: ATI FireGL graphics boards and NVIDIA GeForce based cards - you depend on the manufacturers will to follow the on-going development of the XFree86 graphics display infrastructure. These boards might prove to deliver impressing performance but in many cases - considering the CPU's speed you find in today's PC's - you have many choices which all lead to respectable performance of ***FlightGear***.

As long as you use a distribution provided kernel, you can expect to find all necessary kernel modules at the appropriate location. If you compile the kernel yourself, then you have to take care of two sub-menus in the kernel configuration menu. You'll find them in the "Character devices" menu. Please notice that AGP support is not compulsory for hardware accelerated OpenGL support on Linux. This also works quite fine with some PCI cards (3dfx Voodoo3 PCI for example, in case you still own one). Although every modern PC graphics card utilizes the AGP 'bus' for fast data transfer.

Besides "AGP Support" for your chipset - you might want to ask your main-board manual which one is on - you definitely want to activate "Direct Rendering Manager" for your graphics board. Please note that recent releases of XFree86 - namely 4.1.0 and higher might not be supported by the DRI included in older Linux kernels. Also newer 2.4.x kernels from 2.4.8 up to 2.4.17 do not support DRI in XFree86-4.0.x.

After building and installing your kernel modules and the kernel itself this task might be completed by loading the ‘agpgart’ module manually or, in case you linked it into the kernel, by a reboot in purpose to get the new kernel up and running. While booting your kernel on an AGP capable mainboard you may expect boot messages like this one:

```
> Linux agpgart interface v0.99 (c) Jeff Hartmann
> gpgart:  Maximum main memory to use for agp memory:  439M
> agpgart:  Detected Via Apollo Pro chipset
> agpgart:  AGP aperture is 64M @ 0xe4000000
```

If you don’t encounter such messages on Linux kernel boot, then you might have missed the right chip set. Part one of activation hardware accelerated OpenGL support on your Linux system is now completed.

The second part consists of configuring your X server for OpenGL.This is not a big deal as it simply consists of to instructions to load the appropriate modules on startup of the X server. This is done by editing the configuration file `/etc/X11/XF86Config`. Today’s Linux distributions are supposed to provide a tool that does this job for you on your demand. Please make sure there are these two instructions:

```
Load “glx”
Load “dri”
```

in the “Module” section your X server configuration file. If everything is right the X server will take care of loading the appropriate Linux kernel module for DRI support of your graphics card. The right Linux kernel module name is determined by the ‘Driver’ statement in the “Device” section of the XF86Config. Please see three samples on how such a “Device” section should look like:

```
Section “Device”
   BoardName “3dfx Voodoo3 PCI”
   BusID “0:8:0”
   Driver “tdfx”
   Identifier “Device[0]”
   Screen 0
   VendorName “3Dfx”
EndSection

Section “Device”
   BoardName “ATI Xpert2000 AGP”
```

```
   BusID “1:0:0”
   Driver “ati”
   Option “AGPMode” “1”
   Identifier “Device[0]”
   Screen 0
   VendorName “ATI”
EndSection

Section “Device”
   BoardName “ATI Radeon 32 MB DDR AGP”
   BusID “1:0:0”
   Driver “radeon”
   Option “AGPMode” “4”
   Identifier “Device[0]”
   Screen 0
   VendorName “ATI”
EndSection
```

By using the Option “AGPMode” you can tune AGP performance as long as the mainboard and the graphics card permit. The BusID on AGP systems should always be set to “1:0:0” - because you only have one AGP slot on your board - whereas the PCI BusID differs with the slot your graphics card has been applied to. ‘lspci’ might be your friend in desperate situations. Also a look at the end of /var/log/XFree86.0.log, which should be written on X server startup, should point to the PCI slot where your card resides.

This has been the second part of installing hardware accelerated OpenGL support on your Linux box.

The third part carries two subparts: First there are the OpenGL runtime libraries, sufficient to run existing appliactions. For compiling FlightGear you also need the suiting developmental headers. As compiling the whole X window system is not subject to this abstract we expect that your distribution ships the necessary libraries and headers. In case you told your package manager to install some sort of OpenGL support you are supposed to find some OpenGL test utilities, at least there should be ‘glxinfo’ or ‘gl-info’.

These command-line utilities are useful to say if the previous steps where successfull. If they refuse to start, then your package manager missed something because he should have known that these utilities usually depend on the existence

of OpenGL runtime libraries. If they start, then you're one step ahead. Now watch the output of this tool and and have a look at the line that starts with

OpenGL renderer string:

If you find something like

```
OpenGL renderer string:  FireGL2 / FireGL3 (Pentium3)
```

or

```
OpenGL renderer string:  Mesa DRI Voodoo3 20000224
```

or

```
OpenGL renderer string:  Mesa DRI Radeon 20010402
AGP 4x x86
```

```
OpenGL renderer string:  Mesa GLX Indirect
```

mind the word 'Indirect', then it's you who missed something, because OpenGL gets dealt with in a software library running solely on your CPUİn this case you might want to have a closer look at the preceding paragraphs of this chapter. Now please make sure all necessary libraries are at their proper location. You will need three OpenGL libraries for running ***FlightGear***. In most cases you will find them in /usr/lib/:

```
/usr/lib/libGL.so.1
/usr/lib/libGLU.so.1
/usr/lib/libglut.so.3
```

These may be the libraries itself or symlinks to appropriate libraries located in some other directories. Depending on the distribution you use these libraries might be shipped in different packages. SuSE for example ships libGL in package 'xf86_glx', libGLU in 'xf86glu' and libglut in 'mesaglut'. Additionally for ***FlightGear*** you need libplib which is part of the 'plib' package.

For compiling ***FlightGear*** yourself - as already mentioned - you need the appropriate header files which often reside in /usr/include/GL/. Two are necessary for libGL and they come in - no, not 'xf86glx-devel' (o.k., they do but they do not work correctly) but in 'mesa-devel':

```
/usr/include/GL/gl.h
/usr/include/GL/glx.h
```

One comes with libGLU in 'xf86glu-devel':

`/usr/include/GL/glu.h`

and one with libglut in ‘mesaglut-devel’

`/usr/include/GL/glut.h`

The ‘plib’ package comes with some more libraries and headers that are too many to be mentioned here. If all this is present and you have a comfortable compiler environment, then you are ready to compile ***FlightGear*** and enjoy the result.

Further information on OpenGL issues of specific XFree86 releases is available here:

http://www.xfree86.org/<RELEASE NUMBER>/DRI.html

Additional reading on DRI:

http://www.precisioninsight.com/piinsights.html

In case you are missing some ‘spare parts’:

http://dri.sourceforge.net/documentation.phtml

C.8 OpenGL on Macintosh

OpenGL is pre-installed on Mac OS 9.x and later. You may find a newer version than the one installed for Mac OS 9.x at

http://www.apple.com/opengl

You should receive the updates automatically for Mac OSX.

One final word: We would recommend that you test your OpenGL support with one of the programs that accompany the drivers, to be absolutely confident that it is functioning well. There are also many little programs, often available as screen savers, that can be used for testing. It is important that you are confident in your graphics acceleration because ***FlightGear*** will try to run the card as fast as possible. If your drivers aren’t working well, or are unstable, you will have difficulty tracking down the source of any problems and have a frustrating time.

Appendix D

Landing: Some further thoughts before leaving the plane

D.1 A Sketch on the History of *FlightGear*

History may be a boring subject. However, from time to time there are people asking for the history of ***FlightGear***. As a result, we'll give a short outline.

The ***FlightGear*** project goes back to a discussion among a group of net citizens in 1996 resulting in a proposal written by David Murr who, unfortunately, dropped out of the project (as well as the net) later. The original proposal is still available from the ***FlightGear*** web site and can be found under

http://www.flightgear.org/proposal-3.0.1.

Although the names of the people and several of the details have changed over time, the spirit of that proposal has clearly been retained up to the present time.

Actual coding started in the summer of 1996 and by the end of that year essential graphics routines were completed. At that time, programming was mainly performed and coordinated by Eric Korpela from Berkeley University. Early code ran under Linux as well as under DOS, OS/2, Windows 95/NT, and Sun-OS. This was found to be quite an ambitious project as it involved, among other things, writing all the graphics routines in a system-independent way entirely from scratch.

Development slowed and finally stopped in the beginning of 1997 when Eric was completing his thesis. At this point, the project seemed to be dead and traffic on the mailing list went down to nearly nothing.

It was Curt Olson from the University of Minnesota who re-launched the project in the middle of 1997. His idea was as simple as it was powerful: Why invent the wheel a second time? There have been several free flight simulators available run-

ning on workstations under different flavors of UNIX. One of these, LaRCsim (developed by Bruce Jackson from NASA), seemed to be well suited to the approach. Curt took this one apart and re-wrote several of the routines such as to make them build as well as run on the intended target platforms. The key idea in doing so was to exploit a system-independent graphics platform: OpenGL.

In addition, a clever decision on the selection of the basic scenery data was made in the very first version. ***FlightGear*** scenery is created based on satellite data published by the U. S. Geological Survey. These terrain data are available from

http://edc.usgs.gov/geodata/

for the U.S., and

http://edcdaac.usgs.gov/gtopo30/gtopo30.html,

resp., for other countries. Those freely accessible scenery data, in conjunction with scenery building tools included with ***FlightGear***!, are an important feature enabling anyone to create his or her own scenery.

This new ***FlightGear*** code - still largely being based on the original LaRCsim code - was released in July 1997. From that moment the project gained momentum again. Here are some milestones in the more recent development history.

D.1.1 Scenery

- Texture support was added by Curt Olson in spring 1998. This marked a significant improvement in terms of reality. Some high-quality textures were submitted by Eric Mitchell for the ***FlightGear*** project. Another set of high-quality textures was added by Erik Hofman ever since.

- After improving the scenery and texture support frame rate dropped down to a point where ***FlightGear*** became unflyable in spring 1998. This issue was resolved by exploiting hardware OpenGL support, which became available at that time, and implementing view frustrum culling (a rendering technique that ignores the part of the scenery not visible in a scene), done by Curt Olson. With respect to frame rate one should keep in mind that the code, at present, is in no way optimized, which leaves room for further improvements.

- In September 1998 Curt Olson succeeded in creating a complete terrain model for the U.S. The scenery is available worldwide now, via a clickable map at:

  http://www.flightgear.org/Downloads/world-scenery.html.

- Scenery was further improved by adding geographic features including lakes, rivers,and coastlines later, an effort still going on. Textured runways were added by Dave Cornish in spring 2001. Light textures add to the visual impression at night. To cope with the constant growth of scenery data, a binary scenery format was introduced in spring 2001. Runway lighting was introduced by Curt Olson in spring 2001. Finally, a completely new set of scenery files for the whole world was created by William Riley based on preparatory documentation by David Megginson in summer 2002. This is based on a data set called VMap0 as an alternative to the GSHHS data used so far. This scenery is a big improvement as it has world wide coverage of main streets, rivers, etc., while it's downside are much less accurate coast lines. ***FlightGear***'s base scenery is based on these new scenery files since summer 2002. The complete set is available via a clickable map, too, from

  http://www.randdtechnologies.com/fgfs/newScenery/world-scenery.html.

- There was support added for static objects to the scenery in 2001, which permits placing buildings, static planes, trees and so on in the scenery. However, despite a few proofs of concept systematic inclusion of these landmarks is still missing.

- The world is populated with random ground objects with appropriate type and density for the local ground cover type since summer 2002. This marks a mayor improvement of reality and is mainly thanks to work by D. Megginson.

D.1.2 Aircraft

- A HUD (head up display) was added based on code provided by Michele America and Charlie Hotchkiss in the fall of 1997 and was improved later by Norman Vine. While not generally available for real Cessna 172, the HUD conveniently reports the actual flight performance of the simulation and may be of further use in military jets later.

- A rudimentary autopilot implementing heading hold was contributed by Jeff Goeke-Smith in April 1998. It was improved by the addition of an altitude hold and a terrain following switch in October 1998 and further developed by Norman Vine later.

- Friedemann Reinhard developed early instrument panel code, which was added in June 1998. Unfortunately, development of that panel slowed down later. Finally, David Megginson decided to rebuild the panel code from scratch in January 2000. This led to a rapid addition of new instruments and features to the panel, resulting in nearly all main instruments being included until spring 2001. A handy minipanel was added in summer 2001.

- Finally, LaRCsims Navion was replaced as the default aircraft when the Cessna 172 was stable enough in February 2000 - as move most users will welcome. There are now several flight model and airplane options to choose from at runtime. Jon Berndt has invested a lot of time in a more realistic and versatile flight model with a more powerful aircraft configuration method. ***JSBSim***, as it has come to be called, did replace LaRCsim as the default flight dynamics model (FDM), and it is planned to include such features as fuel slosh effects, turbulence, complete flight control systems, and other features not often found all together in a flight simulator. As an alternative, Andy Ross added another flight dynamics model called ***YASim*** (Yet Another Flight Dynamics Simulator) which aims at simplicity of use and is based on fluid dynamics, by the end of 2001. This one bought us flight models for a 747, an A4, and a DC-3. Alternatively, a group around Michael Selig from the UIUC group provided another flight model along with several planes since around 2000.

- A fully operational radio stack and working radios were added to the panel by Curt Olson in spring 2000. A huge database of Navaids contributed by Robin Peel allows IFR navigation since then. There was basic ATC support added in fall 2001 by David Luff. This is not yet fully implemented, but displaying ATIS messages is already possible. A magneto switch with proper functions was added at the end of 2001 by John Check and David Megginson.. Moreover, several panels were continually improved during 2001 and 2002 by John and others. ***FlightGear*** now allows flying ILS approaches and features a Bendix transponder.

- In 2002 functional multi-engine support found it's way into ***FlightGear***. ***JSBSim*** is now the default FDM in ***FlightGear***.

- Support of ''true" 3D panels became stable via contributions from John Check and others in spring 2002. In addition, we got movable control surfaces like propellers etc., thanks to David Megginson.

D.1.3 Environment

- The display of sun, moon and stars have been a weak point for PC flight simulators for a long time. It is one of the great achievements of ***FlightGear*** to include accurate modeling and display of sun, moon, and planets very early. The corresponding astronomy code was implemented in fall 1997 by Durk Talsma.

- Christian Mayer, together with Durk Talsma, contributed weather code in the winter of 1999. This included clouds, winds, and even thunderstorms.

D.1.4 User Interface

- The foundation for a menu system was laid based on another library, the Portable Library ***PLIB***, in June 1998. After having been idle for a time, the first working menu entries came to life in spring 1999.

  PLIB underwent rapid development later. It has been distributed as a separate package by Steve Baker with a much broader range of applications in mind, since spring 1999. It has provided the basic graphics rendering engine for ***FlightGear*** since fall 1999.

- In 1998 there was basic audio support, i. eản audio library and some basic background engine sound. This was later integrated into the abovementioned portable library, ***PLIB***. This same library was extended to support joystick/yoke/rudder in October 1999, again marking a huge step in terms of realism. To adapt on different joystick, configuration options were introduced in fall 2000. Joystick support was further improved by adding a self detection feature based on xml joystick files, by David Megginson in summer 2002.

- Networking/multiplayer code has been integrated by Oliver Delise and Curt Olson starting fall 1999. This effort is aimed at enabling ***FlightGear*** to run concurrently on several machines over a network, either an Intranet or the Internet, coupling it to a flight planner running on a second machine, and more. There emerged several approaches for remotely controlling ***FlightGear*** over a Network during 2001. Notably there was added support for the "Atlas" moving map program. Besides, an embedded HTTP server developed by Curt Olson late in 2001 can now act a property manager for external programs.

- Manually changing views in a flight simulator is in a sense always "unreal" but nonetheless required in certain situations. A possible solution was supplied by Norman Vine in the winter of 1999 by implementing code for changing views using the mouse. Alternatively, you can use a hat switch for this purpose, today.

- A property manager was implemented by David Megginson in fall 2000. It allows parsing a file called `.fgfsrc` under UNIX/Linux and `system.fgfsrc` under Windows for input options. This plain ASCII file has proven useful in submitting the growing number of input options, and notably the joystick settings. This has shown to be a useful concept, and joystick, keyboard, and panel settings are no longer hard coded but set using *.xml files since spring 2001 thanks to work mainly by David Megginson and John Check.

During development there were several code reorganization efforts. Various code subsystems were moved into packages. As a result, code is organized as

follows at present:

The base of the graphics engine is **OpenGL**, a platform independent graphics library. Based on OpenGL, the Portable Library ***PLIB*** provides basic rendering, audio, joystick etcroutines. Based on ***PLIB*** is ***SimGear***, which includes all of the basic routines required for the flight simulator as well as for building scenery. On top of ***SimGear*** there are (i) ***FlightGear*** (the simulator itself), and (ii) ***TerraGear***, which comprises the scenery building tools.

This is by no means an exhaustive history and most likely some people who have made important contributions have been left out. Besides the above-named contributions there was a lot of work done concerning the internal structure by: Jon S. Berndt, Oliver Delise, Christian Mayer, Curt Olson, Tony Peden, Gary R. Van Sickle, Norman Vine, and others. A more comprehensive list of contributors can be found in Chapter D as well as in the `Thanks` file provided with the code. Also, the ***FlightGear*** Website contains a detailed history worth reading of all of the notable development milestones at

http://www.flightgear.org/News/

D.2 Those, who did the work

Did you enjoy the flight? In case you did, don't forget those who devoted hundreds of hours to that project. All of this work is done on a voluntary basis within spare time, thus bare with the programmers in case something does not work the way you want it to. Instead, sit down and write them a kind (!) mail proposing what to change. Alternatively, you can subscribe to the ***FlightGear*** mailing lists and contribute your thoughts there. Instructions to do so can be found at

http://www.flightgear.org/mail.html.

Essentially there are two lists, one of which being mainly for the developers and the other one for end users. Besides, there is a very low-traffic list for announcements.

The following names the people who did the job (this information was essentially taken from the file `Thanks` accompanying the code).

A1 Free Sounds
Granted permission for the ***FlightGear*** project to use some of the sound effects from their site. Homepage under

http://www.a1freesoundeffects.com/

Raul Alonzo
Mr. Alonzo is the author of Ssystem and provided his kind permission for using the moon texture. Parts of his code were used as a template when adding the texture. Ssystem Homepage can be found at:

http://www1.las.es/~amil/ssystem/.

Michele America
Contributed to the HUD code.

Michael Basler
Author of Installation and Getting Started. Flight Simulation Page at

http://www.geocities.com/pmb.geo/flusi.htm

Jon S. Berndt
Working on a complete C++ rewrite/reimplimentation of the core FDM. Initially he is using X15 data to test his code, but once things are all in place we should be able to simulate arbitrary aircraft. Jon maintains a page dealing with Flight Dynamics at:

http://jsbsim.sourceforge.net/

Special attention to X15 is paid in separate pages on this site. Besides, Jon contributed via a lot of suggestions/corrections to this Guide.

Paul Bleisch
Redid the debug system so that it would be much more flexible, so it could be easily disabled for production system, and so that messages for certain subsystems could be selectively enabled. Also contributed a first stab at a config file/command line parsing system.

Jim Brennan
Provided a big chunk of online space to store USA scenery for ***FlightGear***!.

Bernie Bright
Many C++ style, usage, and implementation improvements, STL portability and much, much more. Added threading support and a threaded tile pager.

Bernhard H. Buckel
Contributed the README.Linux. Contributed several sections to earlier versions of Installation and Getting Started.

Gene Buckle
A lot of work getting ***FlightGear*** to compile with the MSVC++ compiler. Numerous hints on detailed improvements.

Ralph Carmichael
Support of the project. The Public Domain Aeronautical Software web site at

http://www.pdas.com/

has the PDAS CD-ROM for sale containing great programs for astronautical engineers.

Didier Chauveau
Provided some initial code to parse the 30 arcsec DEM files found at:

http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html.

John Check
John maintains the base package CVS repository. He contributed cloud textures, wrote an excellent Joystick Howto as well as a panel Howto. Moreover, he contributed new instrument panel configurations. ***FlightGear*** page at

http://www.rockfish.net/fg/.

Dave Cornish
Dave created new cool runway textures plus some of our cloud textures.

Oliver Delise
Started a FAQ, Documentation, Public relations. Working on adding some networking/multi-user code. Founder of the FlightGear MultiPilot

Jean-Francois Doue
Vector 2D, 3D, 4D and Matrix 3D and 4D inlined C++ classes. (Based on Graphics Gems IV, Ed. Paul S. Heckbert)

http://www.animats.com/simpleppp/ftp/public_html/topics/developers.html.

Dave Eberly
Contributed some sphere interpolation code used by Christian Mayer's weather data base system.

Francine Evans
Wrote the GPL'd tri-striper we use.

http://www.cs.sunysb.edu/~stripe/

Oscar Everitt
Created single engine piston engine sounds as part of an F4U package for FS98. They are pretty cool and Oscar was happy to contribute them to our little project.

Bruce Finney
Contributed patches for MSVC5 compatibility.

Melchior Franz
Contributed joystick hat support, a LED font, improvements of the telnet and the http interface. Notable effort in hunting memory leaks in ***FlightGear***, ***SimGear***, and ***JSBSim***.

Jean-loup Gailly and **Mark Adler**
Authors of the zlib library. Used for on-the-fly compression and decompression routines,

http://www.gzip.org/zlib/.

Mohit Garg
Contributed to the manual.

Thomas Gellekum
Changes and updates for compiling on FreeBSD.

Neetha Girish
Contributed the changes for the xml configurable HUD.

Jeff Goeke-Smith
Contributed our first autopilot (Heading Hold). Better autoconf check for external timezone/daylight variables.

Michael I. Gold
Patiently answered questions on OpenGL.

Habibe
Made RedHat package building changes for SimGear.

Mike Hill
For allowing us to concert and use his wonderful planes, available form

http://www.flightsimnetwork.com/mikehill/home.htm,

for ***FlightGear***.

Erik Hofman
Major overhaul and parameterization of the sound module to allow aircraft-specific sound configuration at runtime. Contributed SGI IRIX support (including binaries) and some really great textures.

Charlie Hotchkiss
Worked on improving and enhancing the HUD code. Lots of code style tips and code tweaks.

Bruce Jackson (NASA)
Developed the LaRCsim code under funding by NASA which we use to provide the flight model. Bruce has patiently answered many, many questions.

Ove Kaaven
Contributed the Debian binary.

Richard Kaszeta
Contributed screen buffer to ppm screen shot routine. Also helped in the early development of the "altitude hold autopilot module" by teaching Curt Olson the basics of Control Theory and helping him code and debug early versions. Curt's ŕossŕob Hain also contributed to that. Further details available at:

http://www.menet.umn.edu/~curt/fgfs/Docs/Autopilot/AltitudeHold/AltitudeHold.html.

Rich's Homepage is at

http://www.kaszeta.org/rich/.

Tom Knienieder
Ported the audio library first to OpenBSD and IRIX and after that to Win32.

Reto Koradi
Helped with setting up fog effects.

Bob Kuehne
Redid the Makefile system so it is simpler and more robust.

Kyler B Laird
Contributed corrections to the manual.

David Luff
Contributed heavily to the IO360 piston engine model.

Christian Mayer
Working on multi-lingual conversion tools for fgfs as a demonstration of technology. Contributed code to read Microsoft Flight Simulator scenery textures. Christian is working on a completely new weather subsystem. Donated a hot air balloon to the project.

David Megginson
Contributed patches to allow mouse input to control view direction yoke. Contributed financially towards hard drive space for use by the flight gear project. Updates to README.running. Working on getting fgfs and ssg to work without textures. Also added the new 2-D panel and the save/load support. Further, he

developed new panel code, playing better with OpenGL, with new features. Developed the property manager and contributed to joystick support. Random ground cover objects

Cameron Moore
FAQ maintainer. Reigning list administrator. Provided man pages.

Eric Mitchell
Contributed some topnotch scenery textures being all original creations by him.

Anders Morken
Former maintainer of European web pages.

Alan Murta
Created the Generic Polygon Clipping library.

http://www.cs.man.ac.uk/aig/staff/alan/software/

Phil Nelson
Author of GNU dbm, a set of database routines that use extendible hashing and work similar to the standard UNIX dbm routines.

Alexei Novikov
Created European Scenery. Contributed a script to turn fgfs scenery into beautifully rendered 2-D maps. Wrote a first draft of a Scenery Creation Howto.

Curt Olson
Primary organization of the project.
First implementation and modifications based on LaRCsim.
Besides putting together all the pieces provided by others mainly concentrating on the scenery subsystem as well as the graphics stuff. Homepage at

http://www.menet.umn.edu/~curt/

Brian Paul
We made use of his TR library and of course of Mesa:

http://www.mesa3d.org/brianp/TR.html, http://www.mesa3d.org

Tony Peden
Contributions on flight model development, including a LaRCsim based Cessna 172. Contributed to ***JSBSim*** the initial conditions code, a more complete standard atmosphere model, and other bugfixes/additions.

Robin Peel
Maintains worldwide airport and runway database for ***FlightGear*** as well as X-Plane.

Alex Perry
Contributed code to more accurately model VSI, DG, Altitude. Suggestions for improvements of the layout of the simulator on the mailing list and help on documentation.

Friedemann Reinhard
Development of an early textured instrument panel.

Petter Reinholdtsen
Incorporated the GNU automake/autoconf system (with libtool). This should streamline and standardize the build process for all UNIX-like platforms. It should have little effect on IDE type environments since they don't use the UNIX make system.

William Riley
Contributed code to add "brakes". Also wrote a patch to support a first joystick with more than 2 axis. Did the job to create scenery based on VMap0 data.

Andy Ross
Contributed a new configurable FDM called ***YASim*** (Yet Another Flight Dynamics Simulator, based on geometry information rather than aerodynamic coefficients.

Paul Schlyter
Provided Durk Talsma with all the information he needed to write the astro code. Mr. Schlyter is also willing to answer astro-related questions whenever one needs to.

http://www.welcome.to/pausch/

Chris Schoeneman
Contributed ideas on audio support.

Phil Schubert
Contributed various textures and engine modeling.

http://www.zedley.com/Philip/.

Jonathan R. Shewchuk
Author of the Triangle program. Triangle is used to calculate the Delauney triangulation of our irregular terrain.

Gordan Sikic
Contributed a Cherokee flight model for LaRCsim. Currently is not working and needs to be debugged. Use configure `--with-flight-model=cherokee` to build the cherokee instead of the Cessna.

Michael Smith
Contributed cockpit graphics, 3-D models, logos, and other images. Project Bonanza

Martin Spott
Co-Author of the "Getting Started".

Durk Talsma
Accurate Sun, Moon, and Planets. Sun changes color based on position in sky. Moon has correct phase and blends well into the sky. Planets are correctly positioned and have proper magnitude. Help with time functions, GUI, and other things. Contributed 2-D cloud layer. Website at

http://people.a2000.nl/dtals/.

UIUC - Department of Aeronautical and Astronautical Engineering
Contributed modifications to LaRCsim to allow loading of aircraft parameters from a file. These modifications were made as part of an icing research project.

Those did the coding and made it all work:
Jeff Scott
Bipin Sehgal
Michael Selig

Moreover, those helped to support the effort:
Jay Thomas
Eunice Lee
Elizabeth Rendon
Sudhi Uppuluri

U. S. Geological Survey

Provided geographic data used by this project.

http://edc.usgs.gov/geodata/

Mark Vallevand
Contributed some METAR parsing code and some win32 screen printing routines.

Gary R. Van Sickle
Contributed some initial GameGLUT support and other fixes. Has done preliminary work on a binary file format. Check

http://www.woodsoup.org/projs/ORKiD/fgfs.htm.

His Ćygwin Tipsṕage might be helpful for you at

http://www.woodsoup.org/projs/ORKiD/cygwin.htm.

Norman Vine
Provided more numerous URL's to the "FlightGear Community". Many performance optimizations throughout the code. Many contributions and much advice for the scenery generation section. Lots of Windows related contributions. Contributed wgs84 distance and course routines. Contributed a great circle route autopilot mode based on wgs84 routines. Many other GUI, HUD and autopilot contributions. Patch to allow mouse input to control view direction. Ultra hires tiled screen dumps. Contributed the initial ǵoto airportánd ŕesetf́unctions and the initial http image server code

Roland Voegtli
Contributed great photorealistic textures. Founder of European Scenery Project for X-Plane:

http://www.g-point.com/xpcity/esp/

Carmelo Volpe
Porting ***FlightGear*** to the Metro Works development environment (PC/Mac).

Darrell Walisser
Contributed a large number of changes to porting ***FlightGear*** to the Metro Works development environment (PC/Mac). Finally produced the first Macintosh port. Contributed to the Mac part of Getting Started, too.

Ed Williams
Contributed magnetic variation code (impliments Nima WMM 2000). We've also borrowed from Ed's wonderful aviation formulary at various times as well. Website at http://williams.best.vwh.net/.

Jim Wilson
Wrote a major overhaul of the viewer code to make it more flexible and modular. Contributed many small fixes and bug reports. Contributed to the PUI property browser and to the autopilot.

Jean-Claude Wippler
Author of MetaKit - a portable, embeddible database with a portable data file format previously used in ***FlightGear***. Please see the following URL for more info:

http://www.equi4.com/metakit/

Woodsoup Project

While ***FlightGear*** no longer uses Woodsoup servies we appreciate the support provided to our project during the time they hosted us. Once they provided computing resources and services so that the ***FlightGear*** project could have a real home.

http://www.woodsoup.org/

Robert Allan Zeh
Helped tremendously in figuring out the Cygnus Win32 compiler and how to link with dll's. Without him the first run-able Win32 version of ***FlightGear*** would have been impossible.

D.3 What remains to be done

If you read (and, maybe, followed) this guide up to this point you may probably agree: ***FlightGear*** even in its present state, is not at all for the birds. It is already a flight simulator which sports even several selectable flight models, several planes with panels and even a HUD, terrain scenery, texturing, all the basic controls and weather.

Despite, ***FlightGear*** needs – and gets – further development. Except internal tweaks, there are several fields where ***FlightGear*** needs basics improvement and development. A first direction is adding airports, buildings, and more of those things bringing scenery to real life and belonging to realistic airports and cities. Another task is further implementation of the menu system, which should not be too hard with the basics being working now. A lot of options at present set via command line or even during compile time should finally make it into menu entries. Finally, ***FlightGear*** lacks any ATC until now.

There are already people working in all of these directions. If you're a programmer and think you can contribute, you are invited to do so.

Achnowledgements

Obviously this document could not have been written without all those contributors mentioned above making ***FlightGear*** a reality.

First, I was very glad to see Martin Spott entering the documentation effort. Martin provided not only several updates and contributions (notably in the OpenGL section) on the Linux side of the project but also several general ideas on the documentation in general.

Besides, I would like to say special thanks to Curt Olson, whose numerous scattered Readmes, Thanks, Webpages, and personal eMails were of special help to me and were freely exploited in the making of this booklet.

Next, Bernhard Buckel wrote several sections of early versions of that Guide and contributed at lot of ideas to it.

Jon S. Berndt supported me by critical proofreading of several versions of the document, pointing out inconsistences and suggesting improvements.

Moreover, I gained a lot of help and support from Norman Vine. Maybe, without Norman's answers I would have never been able to tame different versions of the ***Cygwin*** – ***FlightGear*** couple.

We were glad, our Mac expert Darrell Walisser contributed the section on compiling under Mac OS X. In addition he submitted several Mac related hints and fixes.

Further contributions and donations on special points came from John Check, (general layout), Oliver Delise (several suggestions including notes on that chapter), Mohit Garg (OpenGL), Kyler B. Laird (corrections), Alex Perry (OpenGL), Kai Troester (compile problems), Dave Perry (joystick support), and Michael Selig (UIUC models).

Besides those whose names got lost withing the last-minute-trouble we'd like to express our gratitude to the following people for contributing valuable 'bug fixes' to this version of The FlightGear Manual (in random order): Cameron Moore, Melchior Franz, David Megginson, Jon Berndt, Alex Perry,, Dave Perry,, Andy Ross, Erik Hofman, and Julian Foad.

Index