The FlightGear Manual

Michael Basler, Martin Spott,
Stuart Buchanan, Jon Berndt,
Bernhard Buckel, Cameron Moore,
Curt Olson, Dave Perry,
Michael Selig, Darrell Walisser,
and others

The FlightGear Manual Version 0.9
May 15, 2007
For *FlightGear* version 0.9.11.

Contents

I Installation **9**

1 Want to have a free flight? Take *FlightGear*! **11**
1.1 Yet Another Flight Simulator? . . . 11
1.2 System Requirements . . . 14
1.3 Choosing A Version . . . 15
1.4 Flight Dynamics Models . . . 16
1.5 About This Guide . . . 17

2 Preflight: Installing *FlightGear* **19**
2.1 Installing scenery . . . 19
2.2 Installing aircraft . . . 21
2.3 Installing documentation . . . 21

II Flying with *FlightGear* **23**

3 Takeoff: How to start the program **25**
3.1 Environmental Variables . . . 25
3.1.1 FG_ROOT . . . 25
3.1.2 FG_SCENERY . . . 25
3.2 Launching the simulator under Unix/Linux . . . 26
3.3 Launching the simulator under Windows . . . 27
3.3.1 Launching from the command line . . . 28
3.4 Launching the simulator under Mac OS X . . . 29
3.5 Command line parameters . . . 29
3.5.1 General Options . . . 29
3.5.2 Features . . . 32
3.5.3 Aircraft . . . 32
3.5.4 Flight model . . . 32
3.5.5 Initial Position and Orientation . . . 33
3.5.6 Rendering Options . . . 34
3.5.7 HUD Options . . . 35
3.5.8 Time Options . . . 35

3.5.9 Network Options . . . 36
3.5.10 Route/Waypoint Options . . . 37
3.5.11 IO Options . . . 37
3.5.12 Debugging options . . . 38
3.6 Joystick support . . . 38
3.6.1 Built-in joystick support . . . 38
3.6.2 Joystick support via .fgfsrc entries . . . 44
3.7 A glance over our hangar . . . 45

4 In-flight: All about instruments, keystrokes and menus 51
4.1 Starting the engine . . . 51
4.2 Keyboard controls . . . 52
4.3 Menu entries . . . 56
4.4 The Instrument Panel . . . 60
4.5 The Head Up Display . . . 64
4.6 Mouse controlled actions . . . 65

5 Features 67
5.1 Aircraft Carrier . . . 67
5.1.1 Starting on the Carrier . . . 67
5.1.2 Launching from the Catapult . . . 68
5.1.3 Finding the Carrier - TACAN . . . 68
5.1.4 Landing on the Carrier . . . 69
5.2 Atlas . . . 69
5.3 Multiplayer . . . 70
5.3.1 Quick Start . . . 70
5.3.2 Troubleshooting . . . 71
5.4 Multiple Displays . . . 73
5.4.1 Hardware . . . 73
5.4.2 Basic Configuration . . . 73
5.4.3 Advanced Configuration . . . 74
5.5 Recording and Playback . . . 74
5.6 Text to Speech with Festival . . . 75
5.6.1 Installing the Festival system . . . 75
5.6.2 Running FlightGear with Voice Support . . . 75
5.6.3 Troubleshooting . . . 76
5.6.4 Installing more voices . . . 76
5.7 Air-Air Refuelling (AAR; feature available past 0.9.10) . . . 77
5.7.1 What's possible . . . 77
5.7.2 Necessary preparations . . . 78
5.7.3 In the cockpit . . . 78
5.7.4 In the Air . . . 78
5.7.5 More advanced topics . . . 79

III Tutorials **81**

6 Tutorials **83**
6.1 In-flight Tutorials . . . 83
6.1.1 Cessna 172P tutorials . . . 84
6.2 FlightGear Tutorials . . . 84
6.3 Other Tutorials . . . 84

7 A Basic Flight Simulator Tutorial **87**
7.1 Foreword . . . 87
7.2 Starting Up . . . 88
7.2.1 MS Windows . . . 88
7.2.2 Linux and other unices . . . 89
7.2.3 In the dark? . . . 90
7.3 The First Challenge - Flying Straight . . . 90
7.4 Basic Turns . . . 97
7.5 Taxiing on the ground . . . 98
7.5.1 (. . . 99
7.6 Advanced Turns . . . 102
7.7 A Bit of Wieheisterology . . . 103
7.7.1 Engine control . . . 103
7.7.2 Wings and speed . . . 107
7.7.3 The flaps . . . 109
7.7.4 The stall . . . 110
7.7.5 The trim . . . 111
7.7.6 What direction am I flying? . . . 112
7.8 Let's Fly . . . 113
7.8.1 A realistic take off . . . 113
7.8.2 Landing . . . 114
7.8.3 Engine Shutdown . . . 117
7.8.4 Aborted Landing . . . 117
7.9 Dealing with the Wind . . . 118
7.9.1 Crosswind Take Off . . . 119
7.9.2 Crosswind Landing . . . 120
7.9.3 Taxiing in the Wind . . . 121
7.10 The autopilot . . . 122
7.11 What Next? . . . 123
7.12 Thanks . . . 123
7.13 Flying Other Aircraft . . . 124
7.13.1 How to land the Cherokee Warrior II . . . 124
7.13.2 How to take off and land the Piper J3 Cub . . . 125
7.13.3 How to take off and land a jet . . . 126
7.13.4 How to take off and land the P-51D Mustang . . . 131
7.13.5 How to take off and land the B-52 Stratofortress . . . 132

8 A Cross Country Flight Tutorial **135**
8.1 Introduction . . . 135
8.1.1 Disclaimer and Thanks . . . 135
8.2 Flight Planning . . . 136
8.3 Getting Up . . . 138
8.3.1 Pre-Flight . . . 138
8.3.2 ATIS . . . 138
8.3.3 Radios . . . 139
8.3.4 Altimeter and Compass . . . 141
8.3.5 Take-Off . . . 142
8.4 Cruising . . . 142
8.4.1 The Autopilot . . . 142
8.4.2 Navigation . . . 143
8.4.3 Mixture . . . 143
8.5 Getting Down . . . 146
8.5.1 Air Traffic Control . . . 146
8.5.2 The Traffic Pattern . . . 147
8.5.3 Approach . . . 148
8.5.4 VASI . . . 149
8.5.5 Go Around . . . 150
8.5.6 Clearing the Runway . . . 151

9 An IFR Cross Country Flight Tutorial **153**
9.1 Introduction . . . 153
9.1.1 Disclaimer . . . 154
9.2 Before Takeoff . . . 154
9.2.1 Flight Planning . . . 154
9.2.2 VHF Omnidirectional Range . . . 155
9.2.3 Takeoff . . . 159
9.3 In the Air . . . 159
9.3.1 George I . . . 160
9.3.2 MISON Impossible . . . 160
9.3.3 George II . . . 161
9.3.4 Staying the Course . . . 163
9.3.5 Yet More Cross-Checks . . . 164
9.4 Getting Down . . . 165
9.4.1 Instrument Approach Procedures . . . 165
9.4.2 Nondirectional Beacons . . . 166
9.4.3 Procedure Turn . . . 169
9.4.4 Chasing the Needle . . . 170
9.4.5 FOOTO Time . . . 171
9.4.6 George III . . . 172
9.4.7 ILS Landings . . . 172
9.4.8 Intercepting the Localizer . . . 173

9.4.9 Intercepting the Glide Slope . . . 174
9.4.10 Touchdown I . . . 175
9.4.11 Touchdown II . . . 175
9.4.12 Touchdown III . . . 176
9.5 Epilogue . . . 176

10 A Helicopter Tutorial **179**
10.1 Preface . . . 179
10.2 Getting started . . . 180
10.3 Lift-Off . . . 181
10.4 In the air . . . 181
10.5 Back to Earth I . . . 182
10.6 Back to Earth II . . . 182

IV Appendices **185**

A Missed approach: If anything refuses to work **187**
A.1 FlightGear Problem Reports . . . 187
A.2 General problems . . . 188
A.3 Potential problems under Linux . . . 189
A.4 Potential problems under Windows . . . 189

B Building the plane: Compiling the program **191**
B.1 Preparing the development environment under Windows . . . 192
B.2 Preparing the development environment under Linux . . . 194
B.3 One-time preparations for Linux and Windows users . . . 194
B.3.1 Installation of ***ZLIB*** . . . 195
B.4 Compiling *FlightGear* under Linux/Windows . . . 195
B.5 Compiling *FlightGear* under Mac OS X . . . 198
B.6 Compiling on other systems . . . 200
B.7 Installing the base package . . . 201
B.8 For test pilots only: Building the CVS snapshots . . . 201

C Some words on OpenGL graphics drivers **203**
C.1 NVIDIA chip based cards under Linux . . . 204
C.2 NVIDIA chip based cards under Windows . . . 204
C.3 3DFX chip based cards under Windows . . . 204
C.4 An alternative approach for Windows users . . . 205
C.5 3DFX chip based cards under Linux . . . 205
C.6 ATI chip based cards under Linux . . . 205
C.7 Building your own OpenGL support under Linux . . . 205
C.8 OpenGL on Macintosh . . . 209

D Landing: Some further thoughts before leaving the plane **211**
D.1 A Sketch on the History of *FlightGear* 211
D.1.1 Scenery 212
D.1.2 Aircraft 213
D.1.3 Environment 214
D.1.4 User Interface 214
D.2 Those, who did the work 216
D.3 What remains to be done 224

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 version of the document was written for *FlightGear* version 0.9.11. Users of earlier versions of *FlightGear* will still find this document useful, but some of the features described may not be present.

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* describes some of the 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 7, *A Basic Flight Simulator Tutorial*, provides a tutorial on the basics of flying, illustrated with many examples on how things actually look in *FlightGear*.

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.

Chapter 9, *An IFR Cross Country Flight Tutorial*, describes a similar cross-country flight making use of the instruments to successfully fly in the clouds under Instrument Flight Rules (IFR).

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`.

- There is an official *FlightGear* **wiki** available at http://wiki.flightgear.org,

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.

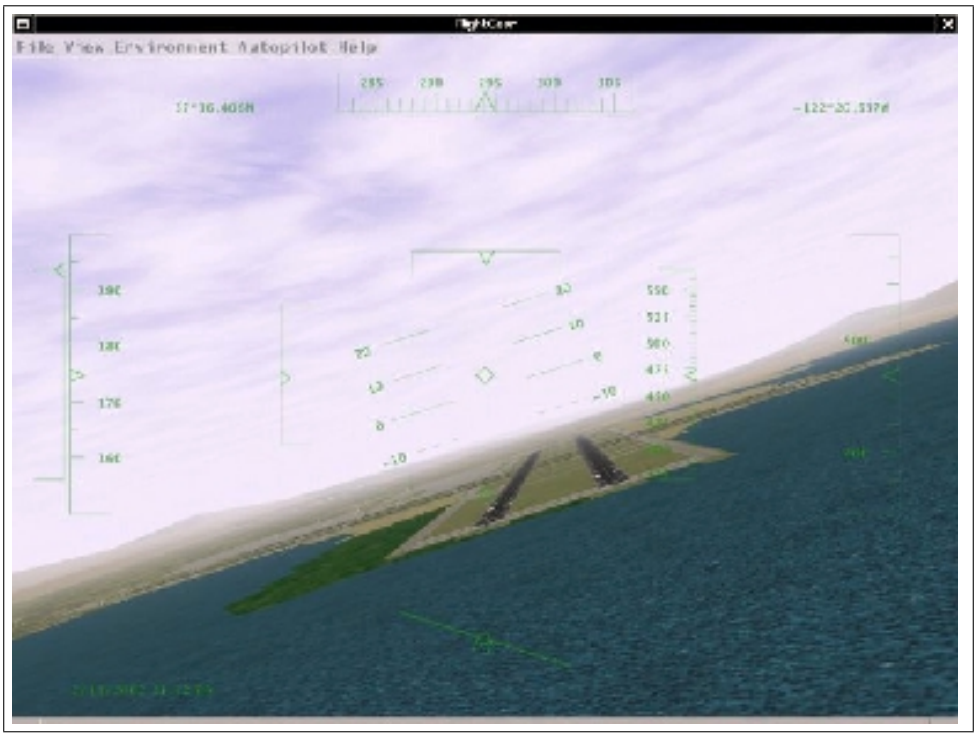


Fig. 1: *FlightGear under UNIX: Bad approach to San Francisco International - by one of the authors of this manual...*

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. The aircraft are downloaded as `.tar.gz` files. Some computers will download them as `.tar.gz.zip` files. If so, simply rename the file to `.tar.gz` and unpack. If you are successful, there will be a new sub-directory in your `FG_ROOT/data/Aircraft` directory containing the aircraft. 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 Environmental Variables

There are two environmental variables that must be defined to run *FlightGear*. These tell *FlightGear* where to find it's data and scenery.

You can set them in a number of ways depending on your platform and requirements.

3.1.1 FG_ROOT

This is where *FlightGear* will find data files such as aircraft, navigational beacon locations, airport frequencies. This is the `data` subdirectory of where you installed *FlightGear*. e.g. `/usr/local/share/FlightGear/data` or `c:\Program Files\FlightGear\data`.

3.1.2 FG_SCENERY

This is where *FlightGear* will look for scenery files. It consists of a list of directories that will be searched in order. The directories are separated by ":" on Unix and ";" on Windows.

For example, a `FG_SCENERY` value of
`c:\Program Files\FlightGear\data\WorldScenery;c:\Program Files\FlightGear\data\scenery`
would search for scenery in
`c:\Program Files\FlightGear\data\WorldScenery`
first, followed by
`c:\Program Files\FlightGear\data\scenery`.

This means that you can download different scenery to different locations.

3.2 Launching the simulator under Unix/Linux

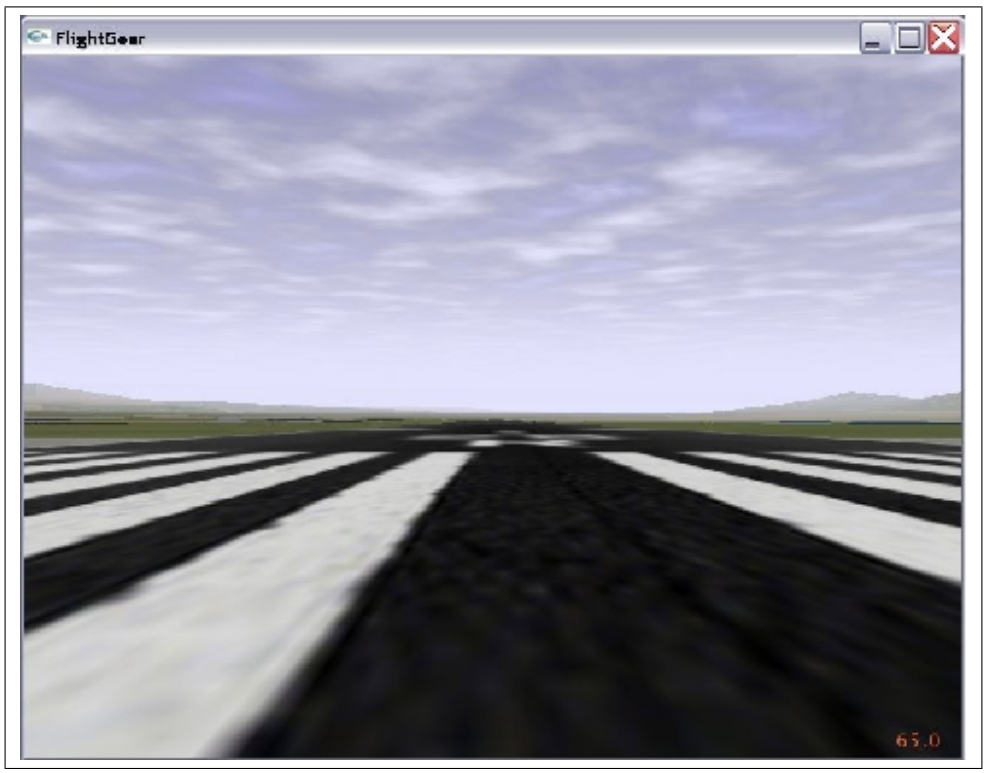


Fig. 3: *Ready for takeoff. Waiting at the default startup position at San Francisco Itl., KSFO.*

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_ROOT` must be set to the data 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 3.5.

3.3 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

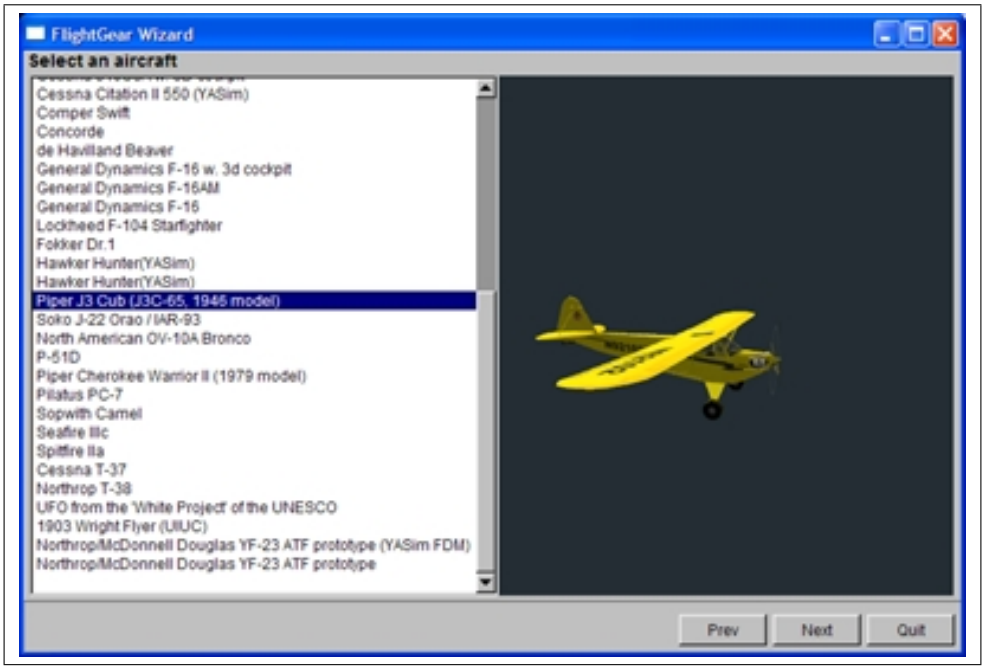


Fig. 4: *The FlightGear Launcher*

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.
```

If you set invalid values or change your scenery directories later, you can change the settings by pressing the ”Prev” button from the first page of the launcher.

3.3.1 Launching from the command line

Alternatively, you can run FlightGear from the command line. To do this, you need to set up the `FG_ROOT` and `FG_SCENERY` environmental variables manually.

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*.

3.4 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
```

3.5 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`).

3.5.1 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.

3.5.2 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.

3.5.3 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. 3.7 below.
- `--show-aircraft`: Print a sorted list of the currently available aircraft types.

3.5.4 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

3.5.5 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).

3.5.6 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.

3.5.7 HUD Options

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

3.5.8 Time Options

- `--time-match-real`

  Synchronize time with real-world time
- `--time-match-local`

  Synchronize time with local real-world time
- `--start-date-sys=`*yyyy:mm:dd:hh:mm:ss*:

  Specify a starting date/time with respect to system time
- `--start-date-gmt=`*yyyy:mm:dd:hh:mm:ss*

  Specify a starting date/time with respect to Greenwich Mean Time

- `--start-date-lat=`*yyyy:mm:dd:hh:mm:ss*

  Specify a starting date/time with respect to Local Aircraft Time

--time-match-real, basically gives our initial behavior: Simulator time is read from the system clock, and is used as is. When your virtual flight is in the same timezone as where your computer is located, this may be desirable, because the clocks are synchronized. However, when you fly in a different part of the world, it may not be the case, because there is a number hours difference, between the position of your computer and the position of your virtual flight.

The option --time-match-local takes care of this by computing the timezone difference between your real world time zone and the position of your virtual flight, and local clocks are synchronized.

The next three options are meant to specify the exact startup time/date. The three functions differ in that the take either your computer systems local time, greenwich mean time, or the local time of your virtual flight as the reference point.

Each of these five options were designed to be used exclusively, however, the value set by these functions could be modified using the --time-offset function. The latter would add a specified amount of time to the value already set by the other functions.

3.5.9 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

3.5.10 Route/Waypoint Options

- `--wp=`*ID[@alt]*

  Allows specifying a waypoint for the GC autopilot; it is possible to specify multiple waypoints (i.eà 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.

3.5.11 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.

3.5.12 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.

3.6 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*.

FlightGear has 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! You can see what *FlightGear* has detected your joystick as by selecting Help -> Joystick Information from the menu.

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.

3.6.1 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
... etcùp 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"/>
```

You can tell how *FlightGear* has interpretted your joystick setup by selecting Help -> Joystick Information from the Menu.

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.

3.6.2 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.

3.7 A glance over our hangar

The following table lists most of the aircraft presently available in the *FlightGear* CVS repository. Your installation will include a sub-set of these, chosen by your distributor. Further aircraft can be downloaded from http://www.flightgear.org/Downloads/aircraft/index.shtml. See Chapter 2 for instructions on how to install them.

In the first column, you will find the name of the aircraft, the second one provides the `-aircraft=` start option if you are using the command line, 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 available. As a result, you will see the default blue-yellow glider when you change to the external view.

Aircraft type	–aircraft option	FDM	Remarks
1903 Wright Flyer	wrightFlyer1903	UIUC	The very first powered aircraft
A6M2 Zero	A6M2	YASim	Japanese WWII fighter
Aerostar Super 700	aerostar700	YASim	Twin-engined light aircraft
Airbus A300	A300	JSBSim	2-engined airliner
Airbus A320	A320	JSBSim	2-engined airliner
Airbus A320	A320	JSBSim	2-engined airliner
Airbus A320-131	A320-111	JSBSim	2-engined airliner
Airbus A300-111	A320-131	JSBSim	2-engined airliner
Airbus A380	A380	JSBSim	4-engined double-decker airliner
Airwave Xtreme 150	airwaveXtreme150	UIUC	Hang-glider
Anotov AN-2	an2	JSBSim	Russian light transport biplane
Antonov AN-225	AN-225	YASim	Russian over-sized cargo aircraft
ASW-20 sailplane	asw20	YASim	Glider
ASW-20 sailplane (UIUC)	asw20-v1-nl-uiuc	UIUC	Glider
Avro Vulcan B.2	vulcanb2	JSBSim	British sub-sonic strategic bomber
BAC TSR2	BAC-TSR2	YASim	Ill-fated strike aircraft
Beech 99	beech99-v1-uiuc	UIUC	Twin-engined turbo-prop aircraft
Beech 99	beech99-yasim	YASim	Twin-engined turbo-prop aircraft
Beechcraft B1900D	b1900d	YASim	Twin-engined turbo-prop aircraft
Bell 206 JetRanger	bell206	YASim	Helicopter
Boeing 314	Boeing314A	JSBSim	Flying boat airline
Boeing 707	707	JSBSim	4-engined early jet airliner
Boeing 737-300	737-300	JSBSim	2-engined airliner
Boeing 747	747	YASim	4-engined airliner
Boeing 747-100	747-100	JSBSim	4-engined airliner
Boeing 747-200	747-200	JSBSim	4-engined airliner
Boeing 787-8 Dreamliner	787	YASim	2-engined airliner
Boeing B-29 Superfortress	b29	YASim	4-engined strategic prop bomber
Boeing B-29 Superfortress	b29-jsbsim	JSBSim	4-engined strategic prop bomber
Boeing B-29 Superfortress	b29-yasim	YASim	4-engined strategic prop bomber
Boeing B-52F	B-52F	YASim	Strategic bomber
Boeing CH47 Chinook	ch47	YASim	Helicopter
Boeing E-3B	E3B	JSBSim	Airborne Warning and Control System (AWACS)
Boeing KC-135	KC135	JSBSim	Air-to-air refueling tanker
Bristol Beaufighter TF X	beaufighter	YASim	4-engined strategic prop bomber
British Aerospace Harrier	harrier	YASim	VTOL jet fighter/bomber
Canberra BI8	CanberraBI8	YASim	First-generation jet bomber
Cessna 150x prototype	c150	JSBSim	Light aircraft
Cessna 172P Skyhawk (1981 model)	c172p	JSBSim	Light aircraft (default)
Cessna 172P Skyhawk (1981 model), 2D panel	c172p-2dpanel	JSBSim	Light aircraft
Cessna 172R	c172r	JSBSim	Light aircraft
Cessna 182	c182	JSBSim	Light aircraft
Cessna 182 (2D panel)	c182-2dpanel	JSBSim	Light aircraft
Cessna 182RG	c182rg	JSBSim	Light aircraft with retractable gear
Cessna 310R	c310	JSBSim	Twin-engined light aircraft
Cessna 310R with IFR panel	c310-ifr	JSBSim	Twin-engined light aircraft
Cessna 310 (YASim)	c310-yasim	YASim	Twin-engined light aircraft
Cessna 310R (1979 model) with 3D cockpit	c310dpm-3d	JSBSim	Twin-engined light aircraft
Cessna 310U3A w. 3D cockpit	c310u3a	JSBSim	US Navy twin-engined light aircraft
Cessna 310U3A	c310u3a-jsbsim	JSBSim	US Navy twin-engined light aircraft
Cessna Citation-Bravo	Bravo	YASim	Executive jet
Cessna Citation-II	Citation-II	YASim	Executive jet
Cessna Citation-X	CitationX	YASim	Executive jet
Cessna T-37	T37	JSBSim	Jet training aircraft
ComperSwift	ComperSwift	YASim	1930s air-racing monoplane
Colditz Escape Glider	colditz	JSBSim	Glider
Concorde	Concorde	JSBSim	Supersonic airliner
Dassault Mirage 2000C/RDI	mirage2000	JSBSim	French delta-winged fighter
Dornier Do 335	do335-yasim	YASim	German WWII fighter
Douglas A4 Skyhawk (YASim)	a4	YASim	US Navy attack aircraft
Douglas A4D (A-4C) Skyhawk	a4-uiuc	UIUC	US Navy attack aircraft
Douglas A4F Skyhawk	a4f	YASim	US Navy attack aircraft
Douglas DC-3	dc3	YASim	Twin-engined early airliner

Aircraft type	–aircraft option	FDM	Remarks
Ecureuil AS 350	as350	YASim	Helicopter
English Electric Lightning F.1A	lightning	JSBSim	British supersonic fighter
Eurocopter Bo105	bo105	YASim	Helicopter
Eurocopter EC135	ec135	YASim	Helicopter
Fairchild-Republic A-10	A-10	YASim	Close Air Support Attack Aircraft
Fokker 100	fokker100	JSBSim	2-engined airliner
Fokker 50	fokker50	JSBSim	2-engined turboprop airliner
Fokker 70	fokker70	JSBSim	2-engined airliner
Fokker Dr.1	fkdr1-v1-nl-uiuc	UIUC	German WWI fighter
General Dynamics F-16	f16	JSBSim	Fighter
General Dynamics F-16 w. 3d cockpit	f16-3d	JSBSim	Fighter
General Dynamics F-16AM	f16-mlu	JSBSim	Fighter
General Dynamics F-16AT (Falcon-21)	f16at	JSBSim	Fighter
Grumman A-6E	A-6E	YASim	USAF Navy Attack Aircraft
de Havilland Beaver (floats)	dhc2F	YASim	Single-engined bush-plane
de Havilland Beaver (wheels)	dhc2W	YASim	Single-engined bush-plane
de Havilland Mosquito	mosquito	YASim	British WWII light bomber
de Havilland SeaVixen FAW2	sea-vixen	YASim	British carrier-borne jet aircraft
Hawker Hunter GA11	hunter	YASim	Jet fighter
Hawker Hunter 2 Tanks	hunter-2tanks	YASim	Jet fighter
Hawker Hurricane IIb	hurricaneIIb	YASim	British WWII fighter
Hawker Seahawk	seahawk	YASim	British carrier-borne jet aircraft
Hughes H4 Hercules	h4-hercules-jsbsim	JSBSim	The Spruce Goose, military transport
Hughes H4 Hercules	h4-hercules-yasim	YASim	The Spruce Goose, military transport
Junkers Ju-52-3m	ju52	YASim	German 3-engined transport/bomber
Kyushu J7W Shinden	j7w	YASim	Japanese WWII fighter, with canard wing
Lockheed 1049	Lockheed1049	JSBSim	Prop airliner
Lockheed C130 Hercules	c130	JSBSim	Military transport
Lockheed F-104 Starfighter	f104	JSBSim	1960s supersonic interceptor
Lockheed F-80C	F80C	JSBSim	First USAF jet-fighter
Mainair Flash 2 Alpha	flash2a	JSBSim	Flexwing microlight
McDonell Douglas F-15 Eagle	f15	JSBSim	Air superiority Fighter
McDonell Douglas F-15C Eagle	f15c	JSBSim	Air superiority Fighter
McDonell Douglas F-15C Eagle	f15c-3d	JSBSim	Air superiority Fighter 3D Cockpit
McDonell Douglas F-18 Hornet	f18	JSBSim	RCAF Air superiority Fighter
McDonell Douglas F-18 Hornet	f183d	JSBSim	RCAF Air superiority Fighter 3D Cockpit
McDonnel Douglas MD11	MD11	JSBSim	3-engined wide-body airliner
McDonnel Douglas MD11	MD11-FINNAIR	JSBSim	MD11 in FinnAir livery
McDonnel Douglas MD11	MD11-KLM	JSBSim	MD11 in KLM livery
Messerschmitt BF-109 G14	bf109g	YASim	German WWII fighter
Messerschmitt Me262	me262	YASim	German WWII jet fighter
MiG-15bis	MiG-15bis	YASim	Russian early jet fighter
Nakajima Ki-84 Hayate	ki-84	YASim	Japanese Nakajima Ki-84 Type 4 Fighter "Hayate"
North American OV-10A	OV10_CDF	JSBSim	California Department of Forestry
North American OV-10A	OV10_NASA	JSBSim	NASA
North American OV-10A	OV10_USAFE	JSBSim	US Airforce
North American P-51D	p51d	YASim	WWII fighter
North American X-15	X15	YASim	Rocketplane
North American X-15	X15-new	JSBSim	Rocketplane
Northrop B-2 Spirit	B-2	YASim	USAF Stealth Bomber
Northrop T-38	T38	JSBSim	Super-sonic jet training aircraft
Northrop/MD YF-23	YF-23	YASim	Prototype fighter aircraft
NTPS	NTPS	YASim	YF-23 ATF prototype
NTPS-Eng	NTPS-Eng	YASim	NTPS Engineer panel
NTPS-HD1	NTPS-HD1	YASim	NTPS Heads Down Display 1
NTPS-HD2	NTPS-HD2	YASim	NTPS Heads Down Display 2
NTPS-OTW-HUD	NTPS-OTW-HUD	YASim	NTPS OTW with HUD
NTPS-OTW-NOHUD	NTPS-OTW-NOHUD	YASim	NTPS OTW without HUD
NTPS-OTW-NOHUD	NTPS-OTW-NOHUD	YASim	NTPS OTW without HUD
ogeL experimental	ogel	JSBSim	Educational aircraft
Ornithopter	ornithopter	UIUC	Bird-like aircraft
paraglider	paraglider	JSBSim	Paraglider
Pilatus PC-7	pc7	JSBSim	Turbo-prop training aircraft
Piper J3C-65 Cub	j3cub	YASim	Classic light aircraft (1946 model)
Piper PA24-250 Comanche 250	pa24-250	YASim	Light aircraft (1962 model)
Piper PA28-161 Cherokee Warrior II	pa28-161	YASim	Light aircraft (1979 model)
Piper PA34-200T Seneca II	SenecaII-jsbsim	JSBSim	Twin-engined piston aircraft
Piper PA34-200T Seneca II	SenecaII-yasim	YASim	Twin-engined piston aircraft

Aircraft type	–aircraft option	FDM	Remarks
Rascal 110	`Rascal110-JSBSim`	JSBSim	Radio Controlled model
Rascal 110	`Rascal110-YASim`	YASim	Radio Controlled model
Santa Claus (3d cockpit)	`santa`	YASim	Unique supersonic, cargo aircraft
Schweizer 2-33	`sgs233`	YASim	Glider
Siai Marchetti S.211 (UIUC)	`marchetti`	UIUC	Italian fast light aircraft
Sikorsky CH-53E Super Stallion	`ch53e`	YASim	Helicopter
Sikorsky S76C	`s76c`	YASim	Helicopter
Soko J-22 Orao / IAR-93	`j22`	YASim	Yugoslav/Rumanian aircraft
Sopwith Camel 1F.1	`sopwithCamel`	UIUC	British WWI fighter
Sopwith Camel	`sopwithCamel-v1-nl-uiuc`	UIUC	British WWI British fighter
Space Shuttle	`shuttle`	JSBSim	Re-entry simulation
Sukoi SU-37	`SU-37`	YASim	Russian multi-role jet aircraft
Supermarine Seafire MkIIIc	`seafireIIIc`	YASim	British WWII carrier prop aircraft
Supermarine Spitfire IIa	`spitfireIIa`	YASim	British WWII fighter
Tupolev TU-114	`TU-114`	YASim	Russian 4-engined turboprop airliner
Tupolev TU-154	`tu154`	YASim	Russian 3-engined jet airliner
UFO	`ufo`	ufo	From the 'White Project' of UNESCO
USAF/NACA X-24B	`x24b`	JSBSim	Re-entry testbed
Vought F4U-1 Corsair	`f4u`	YASim	Carrier-capable fighter/bomber
Yard Stik	`YardStik`	JSBSim	Radio controlled model

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.

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

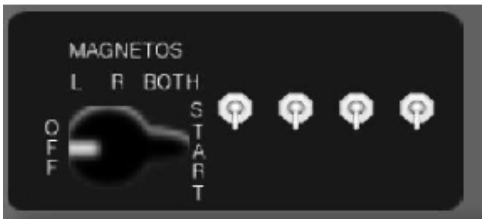


Figure 4.1: The ignition switch.

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.
  - **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).
  - **Toggle Dynamic Cockpit View** Toggle between a fixed view-point (controlled by the mouse) and one that simulates the movement of your head as you fly the aircraft.
  - **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.
  - **Chat** Displays a dialog allowing you chat with other aircraft in the multi-player environment.

- **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.
  - **Joystick Information** Displays information about any joystick in use, including axis and button assignements.
  - **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

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

Aircraft within *FlightGear* can have both a 2-dimensional instrument panel and a 3-dimensional cockpit. The 3-dimensional cockpit provide a much more realistic pilot-eye view, but can be difficult to read with small monitors.

The default Cessna 172P (c172p) has both a 3-dimensional and 2-dimensional cockpit. The 3-dimensional cockpit is actived by default when you start *Flight-*

Gear, but you can over-lay the 2-dimensional instrument panel by selecting View -> Toggle 2D Panel from the menu, or 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.eả 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.eśtep onto the left rudder pedal in case the ball is on the l.h.s.

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, respmarker 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`.

Fig. 5: *The Cessna 172p 2-D instrument panel.*

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. e. the 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.

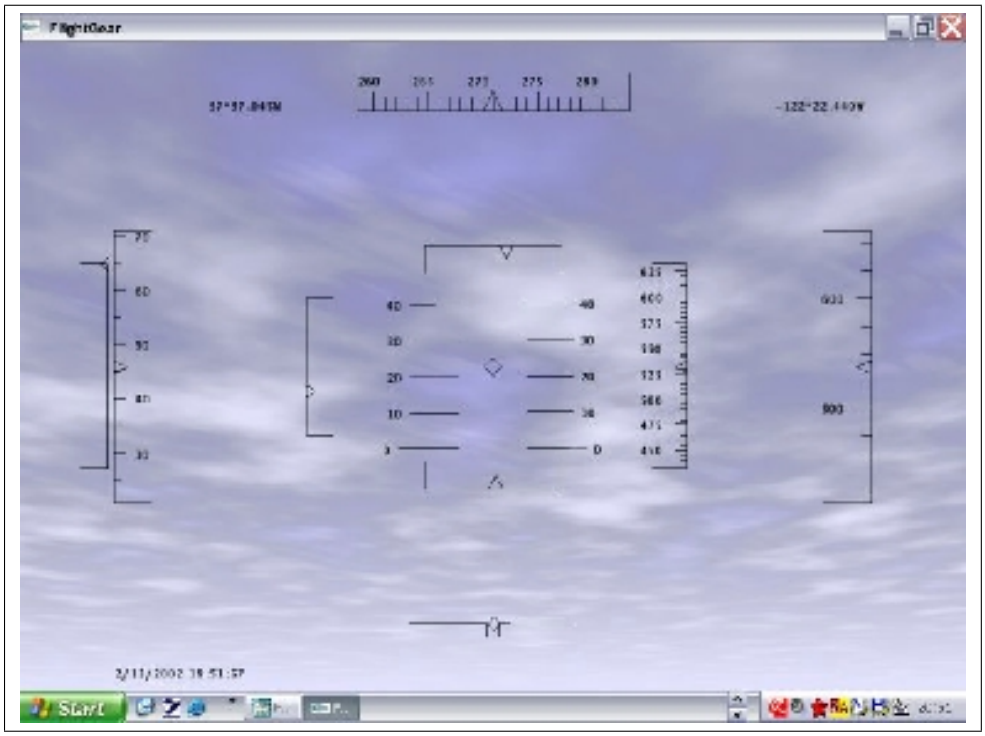


Fig. 7: *The HUD, or Head Up Display.*

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://wiki.flightgear.org/

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 and hold “L” to engage the launchbar. You must hold down “L” until you have successfully launched. 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. You must also check The AI models checkbox under Features to make other aircraft visible.

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>
--enable-ai-models
```

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
     --enable-ai-models
```

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
```

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
--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

If you are new to flying, an advanced simulator such as *FlightGear* can seem daunting: You are presented with a cockpit of an aircraft with little information on how to fly it.

In real life, you would have an instructor sitting next to you to teach you how to fly and keep you safe.

While we cannot provide a personal instructor for every virtual pilot, there are a number of tutorials available that you can follow to become a proficient virtual pilot.

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).

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.

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.

Within a tutorial, 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.

6.1.1 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.

6.2 FlightGear Tutorials

The following chapters provide *FlightGear* specific tutorials to take the budding aviator from their first time in an aircraft to flying in the clouds, relying on their instruments for navigation. If you have never flown a small aircraft before, following the tutorials provides an excellent introduction to flight.

Outside of this manual, 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.

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 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 work, 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 high speeds. Yet it is one of the safest forms of transport.
- Pilots must constantly follow rules and procedures. Yet an airplane is a symbol of freedom.
- With a little training, flight a small aircraft is easy. Yet if a problem occurs, you must be able to resolve it in a few seconds.
- Many flight tutorials are written with a lot of humor. Yet not taking flying seriously will bring you down to earth prematurely.

The aircraft used in this tutorial is the Cessna 172p. This is the aircraft used in many real life flight schools and a great airplane to fly.

The following articles complement this tutorial and will answer most questions that may arise as you read through.This first one in paricular is a good introduction to the airplane's main components 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/

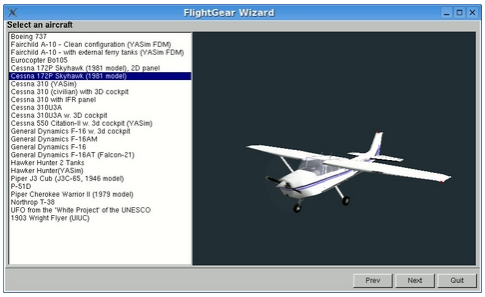
This tutorial is accurate to the best of my knowledge, but will inevitably contain some mistakes. I apologize in advance for these.

7.2 Starting Up

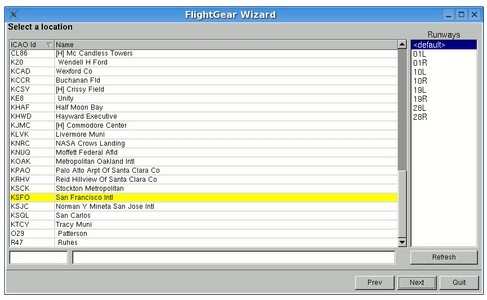
There are a number of different ways to start *FlightGear* based on your platform and the distribution you are using

7.2.1 MS Windows

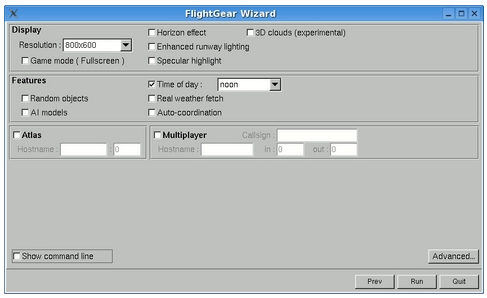
On MS Windows, *FlightGear* has a GUI Wizard in which you can choose your aircraft and starting postion. First choose the Cessna 172p airplane as shown below. To match this tutorial do not choose the 2D panel version. (You may however find in the future that the 2D version is more appropriate for training) Press the **Next** button to choose your airport.



You can start from any airport for this tutorial, but I will assume that you will start from FlightGear's default airport of San Francisco (KSFO):



Once you have selected KSFO and pressed the **Next** button, you can set any number of options for the simulator. For your first flight, I suggest starting at noon. I would also recommend that you start with a small resolution of 800×600. Later on you can play around with the options and use a higher resolution, but this obviously adversly affects performance.Press the **Run** button and the *FlightGear* will start with the options you selected.



If you have problems running the latest version FlightGear on your Windows system, you may want to try an earlier version with lower graphics requirements (for example 0.9.8) You can find previous releases on the FTP mirrors mentioned at the top of the *FlightGear* download page: .

If you are running under Windows Me and the flight simulator suddenly starts stuttering, with the frame rate dropping, 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 an obvious security risk. Also, on one Windows Me machine, a *FlightGear* of 800×600 yielded good results, while a lower resolution of 640×480 triggered much lower FPS levels (Frames Per Second).)

7.2.2 Linux and other unices

On Linux and other Unix-like systems, you may have to run *FlightGear* from the command line. If you have installed *FlightGear* but cannot find it in your menu system, try the following:

- From a terminal window (also named "console" window) try running the `fgrun` command. If installed, this will run the same GUI Wizard as under Windows as described above.

- Alternatively, open a terminal window and type the following command: `fgfs -timeofday=noon -geometry=800x600`.

7.2.3 In the dark?

Without the `-timeofday=noon` option, *FlightGear* will start at the current time in San Francisco - often night-time if you are in Europe. To change the time of day in the simulator to daytime, select **Weather->Time of Day** from the menu and select **Noon**.

If running *FlightGear* from a menu (e.g. under KDE or Gnome), you can edit the FlightGear launch icon properties and change the simple `fgfs` fgfs command to something like `fgfs -timeofday=noon -geometry=1024x768`, or include whatever command options you wish. Further details of the command line options can be found in Chapter 3, *Takeoff: How to start the program*.

7.3 The First Challenge - Flying Straight

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

On startup, the aircraft is at the end of the runway with the engine running at low power. The airplane will occasionally tremble a little, but it won't move.

About the keyboard.

- In this tutorial, a lowercase key letter indicates you should simply press that key. An uppercase means you must press shift and 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", press the **Shift** key down and while you have it pushed down, hit the **v** key, then release the **Shift** key. (In short: V is the same as **Shift-v**.)
- The tutorial will assume you have the the **NumLock** switched on. When switched on, you should find a small green light on at the right of your keyboard. Press the NumLockkey repeatedly until the lamp is on.

Press **v**, to view the aircraft from the outside. Type v repeatedly to scroll through a number of different views until you return to the cockpit. Typing **V** will cycle backwards through the views.):

In real life, we would have inspected the airplane all around to check everything is working, nothing is hampering the moving parts, and nothing is obstructing the instrument openings. In the simulator, this is already done for us before we start.

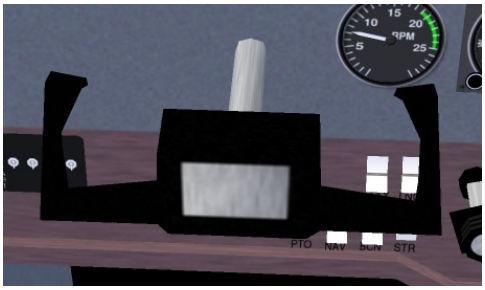
Hold the **Page Up** key down for about eight seconds. You will hear the engine sound rise.

The airplane will start accelerating down the runway. As it does so, it will drift to the left, before finally taking off, banking to the left, falling to the ground and crashing (probably).

You can see a replay of the crash using the **View -> Instant Replay** menu. Click the **Replay** button at the bottom of the dialog window, then use **v** and **V** to see the airplane from the outside. The picture below shows the end part of the flight. You can take a snapshot by typing the **F3** key. You can also use the **F10** key to toggle the menu bar on or off.

Having observed your crash, exit from *FlightGear*(using **File->Quit**) and restart the simulator using the same options as before.

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



You can control the yoke using a joystick, or by moving the mouse. To use the mouse 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. If you move the mouse again you will see the tail elevator and the ailerons at both wings ends move. If your viewpoint is too far from the aircraft to see any movement, type **x** a

few times to zoom in. Type **X** to zoom back out. **Ctrl-x** returns the view to the default zoom level. Type **V** to change the view back to the cockpit.

Clicking the right mouse button again gets you in mouse view mode. In this mode the mouse cursor will be a ↔. sign. This allows you to look around easily. Clicking the left mouse button will re-center the view. A further right click will return you to the normal mouse mode.

To summarize, the right mouse button cycles the mouse through three modes:

- *Normal mode*. This mode allows you to click on the menu and on the instrument panel.
- *Yoke mode*. The mouse controls the yoke (+ pointer shape).
- *View mode*. The mouse controls the view direction (↔ pointer shape).

Try taking off again using the mouse to control the yoke. Right-click to put the mouse in yoke mode (+pointer shape) and raise the engine throttle to maximum by holding the **Page Up** key 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 will find that you must prevent the airplane from banking to the left:

... or to the right:

... or from plunging to the ground:

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

Whatever your skills at video games or simpler simulators, you will probably not succeed at first. The airplane will crash, probably quite soon after take-off. This is the moment where most candidates get desperate and abandon trying to fly a simulator or a real aircraft. Just hold tight and keep trying. Eventually you will develop a feel for the subtle control inputs required.

The most common error is moving the mouse forwards to bring the nose up. In fact, you must pull the yoke by moving the mouse backwards to do this.

Equally, when you want to lower the airplane's nose, you must move the mouse forwards. This can seem odd, but all airplane control yokes are designed that way. With time, you will wonder how you every thought it worked any other way. You will also find that small mouse movements have a large effect on the aircraft. You may find that decreasing your mouse sensitivity may help initially.

If you have difficulty visualising this, the following analogy may help. Imagine a soccer ball is on your desk and you have "glued" your hand to the top of it. If you move your hand forwards the ball will roll forwards and your fingers will point to to the desk. If you move your hand backwards the ball will roll back and your fingers will now point up at the ceiling. Your hand is the airplane:

Another common error is the assumption that the control inputs directly match airplane bank. In other words, you believe if the control yoke is level, the airplane will fly level. This is not true. The yoke controls the *rate* 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 until some other force affects it. If you want to return the airplane to level flight, 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, bring the control yoke level too. Then the airplane will stay level (until some other force changes its orientation).

A third error is trying to find "the right position" for the yoke/mouse. Naturally, you will want to find the fine tuning that will leave the airplane fly straight. Actually there is no such ideal yoke position. The airplane is inherintely unstable in the air. You must constantly correct the airplane's attitude and keep it flying straight with tiny movements of the mouse. This may seem to take all your concentration intially, but just like driving a car, keeping the aircraft straight and level will soon become second nature. For longer flights, you will eventually use the autopilot to keep the airplane level, but this is outside the scope of this tutorial.

To help fine-tune your senses to the control inputs required, keep your eyes on the outside scenery and not get fixated on the instruments or the yoke. Check the angle of the horizon and its height above the airplane's nose. The horizon line and the 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. Once the mouse leaves the window, it stops controlling the aircraft, often at the worse possible moment! If you wish to use 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. While initially this may seem easier than the mouse, you cannot make the very fine adjustments required for accurate flying, so it is much better to persevere with the mouse.

You may hear beeping sounds while flying around the airport. These are landing aid signals. Don't worry about them for the moment.

You will know that you have mastered this when you can make the aircraft climb steadily in the air. The next step is to learn to keep the aircraft at a constant altitude, or to make it ascend or descend slowly and under your control.

Keeping the aircraft at a constant altitude involves observing the altimeter and making small changes with the mouse forwards or backwards to stop the aircraft ascending or descending respectively.

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. The altimeter below shows an altitude of 300 feet, approximately 100 meters.

As you ascend or descend the altimeter will change accordingly, turning anticlockwise as you descend, and clockwise as you gain height. If you see the altimeter "unwinding" you will be able to tell that you are losing height and move the mouse backwards slightly to raise the nose. After a while you will notice that when flying level the nose of the aircraft is always in the same position relative to the horizon. This is the aircraft attitude for level flight. By putting the nose in that same position, you will achieve almost level flight without having to reference the instruments. From there you can fine-tune your altitude.

Beware: an altimeter does not automatically show the absolute altitude above sea level. You must adjust for the local air pressive. The little black knob on the lower left side of the altimeter allows you to adjust 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 forward. Use that little knob to tune in the current 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.

To make settings the altimeter easier, airports advertise their altitude in various ways. They may provide a radio service (called ATIS in the USA) to broadcast the current air pressure at sea level. This is expressed in inches of mercury. The altimeter contains a small scale inside which is calibrated in this way. You can set your altimeter using this scale. Alternatively, if you are on the ground and know the altitude of the airport, you can simply adjust your altimeter until it displays the correct altitude.

Note that there is an important difference between “altitude above sea level” and “altitude above the ground”. If you fly near Mount Everest at an altitude of 24,000 feet above sea level (AMSL), your altitude above the ground (AGL) will be much less. Knowing the altitude of the ground around you is obviously useful.

7.4 Basic Turns

While if you had enough fuel you could return to the same airport by flying straight head for thousands of miles, being able to change direction will make your flying more enjoyable and useful.

Once you are able to fly more or less straight, it is time 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.

To turn, you do not need high levels of bank. 20° is more than enough for a safe and steady turn. The turn coordinator indicates your angle of bank by showing a depiction of your aircraft from behind. The picture below shows the turn coordinator when the airplane is banked 20° to the right. You can also tell the bank angle by observing the angle of the horizon.

Try the following: keep the airplane banked around those 20° for a few minutes and keep your eyes outside the aircraft 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 particularly useful when navigating. Whatever speed the airplane is flying, if you bank at 20° you always need 60 seconds to make a 180° turn in the Cessna 172P.

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. In real life you would feel these as your turn, however it is not possible to simulate these, so you must simply keep an eye on the ball. If you turn neatly (using the rudder a little bit), 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 the centrifugal force.

By experimenting you will notice you can make much steeper turns by banking the airplane to high angles and pulling back on the yoke. Turns at over 60° bank angle are the realm of aerobatics and military flying, and dangerous is aircraft such as the Cessna.

7.5 Taxiing on the ground

While *FlightGear* starts you by default conveniently lined up on the runway and ready to go, you may be wondering how to get your aircraft from its hangar, along the taxi-ways to the runway. This is taxiing.

The picture below shows the instrument. It displays how fast the engine is turning in hundreds of revolutions per minute (RPM).

Type the **`Page Up`** key a few times, until the tachometer is showing 1,000 RPM (as shown above). If required type the **`Page Down`** key to decrease the engine speed.

At roughly 1,000 RPM, the airplane will move forward on the runway, but it will not accelerate and take off.

Type the "`.`"key (**`Shift-;`** on Azerty keyboards). The airplane will make a sharp turn to the right. If you keep the "`.`"key down the airplane will halt. When you type the "`.`" key, you are activating the brake on the right wheel of the airplane.

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. Using the throttle and the brake pedals you can control the speed of the aircraft and cause it to turn on the ground.

The brakes can be very useful when taxiing slowly on the runway. You can also steer the nose-wheel of the aircraft. 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. If you don't have real rudder pedals, there are two ways to control the virtual 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 that way. 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 pedals instead of the yoke. The rudder pedals are connected to both the rudder and nose-wheel. This method is much more precise.

Start the simulator, Type **v** or **V** to view the airplane from the outside and keep **x** down a couple of seconds to zoom in 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 while the front wheel is on the ground and use the keypad **0** and **Enter** keys once it has lifted off. In other words: I keep the left mouse button down while the front wheel is on the ground. This allows for a precise and easy rudder control on the ground. Then I simply release the left mouse button once the front wheel lifts off.

7.5.1 (

Airspeed)

Just like driving a car, it is good to know how fast you are traveling. The aviation equivalent of a speedometer is the airspeed indicator (ASI), calibrate in nautical miles per hour (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. Note that some aircraft ASIs (in particular the Piper J3 Cub) display mph instead of knots.

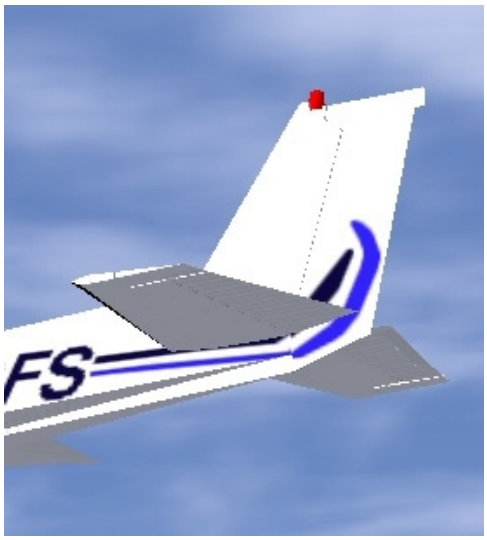
The airspeed indicator displays the speed of the aircraft compared to the surrounding air, not the speed compared to the ground like a car speedometer does. If the plane is halted on the ground and there is a 10 knot wind blowing from straight ahead, the airspeed indicator will display 10 knots airspeed, although the plane will not be moving relative to the ground.

When the airplane rolls over the runway at more than 40 knots, you must prevent the front wheel from touching the ground. The nosewheel is not designed for high speeds and in real life would shimmy and wear out.

During take off, once over 40 knots you can make the front wheel leave the ground by pulling back gently on the control yoke. Don't turn sharply at high speed on the ground. Doing so may cause the aircraft to tip over.

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 very slightly.

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



At airspeeds above 40 knots, the rudder has enought air-flow over it to steer the airplane.

Note the front wheel and the tail rudder don't make the airplane turn at exactly the same rate. 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've become familiar with the nose-wheel and rudder, you can use these new controls to keep the airplane straight on the runway during take-off.

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. Don't wait till the aircraft has straightened up completely. Type keypad **Enter** before the aircraft reaches the direction you wish to travel. Otherwise you will find that you will over-correct and have to turn back again. If you use the mouse, such corrections are much easier and more precise.

To summarise: two methods exist to steer the airplane on the ground: the differential brakes on the side wheels and the rudder pedals. This control redundancy is very common in aviation. If one method fails, you still have another method available to perform the task.

You may be wondering why the aircraft drifts to the left when it rolls on the ground, forcing your to compensate with a little push on the right rudder pedal? The main reason is the flow of air produced by the propeller. It blows along the airplane body, but also corkscrews around the airplane fuselage. The upper part of that slight vortex pushes the vertical tail to the right. This causes makes the front of the aircraft to yaw 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 if you find yourself with controls all over the place!

7.6 Advanced Turns

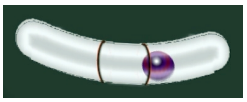
As with turning on the ground, there are two methods of turning in the air. You can use the wing ailerons (steered by the yoke/mouse) as described above or you can use the tail rudder (steered by the rudder pedals / the keypad keys **/** `0` and **`Enter`**.

Why two ways? Partially for redundancy, but mainly because they are complementary. The main effect of the rudder is yaw (rotation around the vertical axis), while the main effect of the ailerons is roll (rotation around the longitudonal axis).

- When flying close to the ground, it is better not to bank the airplane in order to turn. The rudder is used more instead. Acting on the rudder pedals allows you to turn the airplane without excessive banking.
- When the plane is just above the runway, the two side wheels need to be at the same height above the runway for landing. 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 is harmless.
- In flight, especially at high speed, the rudder is an in-efficient way to turn the aircraft:
  - It causes the airplane to present its flank to the airstream, increasing drag.
  - The airplane turns very slowly.
  - You will lack control while turning.
  - 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.
- The rudder can be vital when the wings are stalled. Indeed, during a stall the wing ailerons become less effective or even useless. (Note that some airplanes can go in a very dangerous stall if you overdo the rudder control at low speed.)

When you turn in flight, using the ailerons, you still need the rudder a little bit. You add a little bit of rudder. This allows you to compensate for the adverse yaw created when you roll using the ailerons. In a real aircraft, you can feel this sideways motion. In the simulator, you can check this visually on the turn coordinator. In 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. You can compensate this by pushing the right rudder pedal (type the keypad **`Enter`** key a few times). In normal flight you should use the rudder to keep the little ball centered.



So, in normal flight use the ailerons to turn, while close to the ground at low speed use the rudder. However, 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.

Even when taxiing, you should use the ailerons. Otherwise, strong winds can blow the aircraft onto its side. To counteract this, your should turn the ailerons into the wind. This raises the aileron in the wind, helping to keep the wing down.

You should avoid making quick and agressive movements of the rudder. On the ground at high speed this can make the airplane turn too sharply. In flight at low speed it can cause a very dangerous type of stall. In flight at high speed it can cause all kinds of aerodynamic and physical discomfort. Instead, make gentle movements of the rudder.

I recommend you practise turning 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. Use the rudder to turn towards a ground feature and maintain a heading, then turn the aircraft towards a new heading. 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 comes from the German phrase "Wie heißt Er" – "What's that name". This section is about gauges, switches and controls of the aircraft. While in the simulator you can take off and land a basic airplane with just the engine throttle and the yoke, but you will need all the controls to perform securely and efficiently.

7.7.1 Engine control

An airplane engine is designed for simplicity, reliability and efficiency. Rather than use advanced electronic ignition and fuel injection systems found in modern cars, they instead use older technology that doesn't rely on electrical power. That way, the plane can still fly even if it suffers complete electrical failure.

Magneto

On the bottom left, below the instrument panel you will find the magneto switch and 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).

You can move the switch with the **{** and **}** keys (use the **Alt Gr** key on Azerty keyboards).

You are probably aware that the fuel inside a car engine is ignited by electric sparks. Modern car engines use electronic ignition. An airplane engine uses a more old-fashioned (but more reliable) magneto ignition instead. For redundancy, it contains two such magnetos: the "left" one and the "right" one. When you change the magneto switch on OFF, both magnetos are switched off and the engine will not run. With 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 will use BOTH.

Given that you use both magnetos in flight, why have the switch? The reason is that during your pre-flight checks you will verify that each of the magnetos is working correctly. To do this, increase the RPM to about 1500 then switch the magneto switch to L and observe the tachometer. You should observe a slight drop in RPM. If the engine cuts out, the left magneto is broken. If you do not see an RPM drop, then the switch may be faulty, as both magnetos are still switched on. You can then perform the same test on the right magneto. Of course, in the simulator, the magnetos are unlikely to fail!

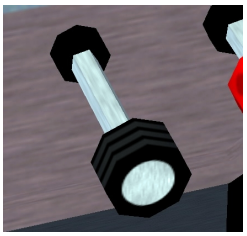
Should one of the two magnetos fail in flight, the other one will keep the engine running. The failure of one magneto is rare, the failure of both simultaneously is almost unheard of.

You may have typed **{** to shut the engine down. To start the engine again after doing so, type **}** three times in order to put the magneto switch on BOTH. Then use the starter motor by pressing the **Space Bar** for 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 in normal mouse mod). 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 as the propeller is still turning. If you wait for the propeller to stop, 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.)

Throttle

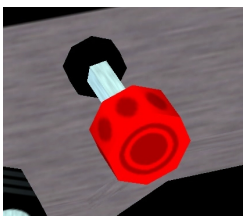


You already know that you increase the engine power by pushing that throttle rod in (**`Page Up`** key). You decrease the power by pulling the control out (**`Page Down`** key). You can also click left and right of the lever (middle mouse button for quicker moves, **`Ctrl-c`** to highlight the left and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

What does "increase the power" actually mean? 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 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.

Mixture

The amount of air compared to the amount of fuel is critical. 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, pulled out far too much.



When the mixture lever is fully pushed in, you feed the engine with an lots of fuel and little air. This is known as a "rich" mixture. When the lever is pulled out completely, there is an excess of air, known as a "lean" mixture. The correct position to produce maximum power is in between these two extremes, 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 should be 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 wasted and pushed out the exhaust. 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 normal flight, you have to pull the mixture lever a little, to get a more 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 should now be 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 waste no fuel 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. For thermodynamic reasons, the best mixture isn't exactly at maximum power - it is better for the engine to be running very slight richer or leaner than maximum power. This also avoids the possibility of the fuel detonating explosively damaging the engine. You can find the maximum power point by the fact you get the highest RPM. (Another method is to check the engine exhaust temperature. Roughly, this is the point at which you get the highest temperature.)

The mixture control allows you to burn less fuel for the same speed and distance, and therefore fly farther and pollute less. However, if you mis-manage it, it can cause serious problems. Suppose you go flying at high altitude and pull out the mixture lever accordingly. At high altitude there is less oxygen available so the correct mixture will be quite lean - i.e. with little fuel being used. Then you descend back in order to land. If you forget to push the mixture lever in as you descend, The fuel/air mixture will become far too lean and the engine will simply halt.

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 lever. 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 choked by waste fuel residues. Finally, turn the magneto switch to OFF to ensure the engine won't start again accidentally.

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 significantly as does your airspeed. It rises while diving and decreases while climbing.

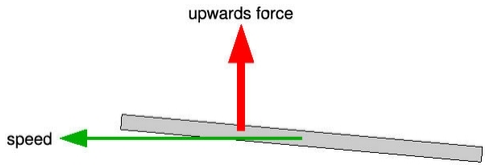
One pitfall of this is when you intend to tune the engine power in for landing. Suppose you're descending towards the airport, flying fast. You know the ideal RPM for landing is around 1,900 RPM. So you pull the throttle till you get 1,900

RPM. You think you tuned in the appropriate RPM. You think you shouldn't bother any more about it. But when you level off, the plane's speed starts to decrease, along with the RPM. A few minutes later, you get the low flight speed you wanted. You don't see the RPM is now far too slow. You will either lose altitude or stall. Or both. 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

Say you are flying with full engine power. Dropping the nose a little makes you lose altitude and raising the nose a little makes you gain altitude. You may think this is quite straightforward. The plane travels in the direction it is heading; the direction the propeller is heading. This is not the best way to think about it. This model 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. This is the angle of attack.

I wrote above that when the wings travel through the air with no angle of attack, they don't produce 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. This makes them create lift even when traveling through the air at no angle of attack.

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!

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. Move it at different speeds through the air. The faster you move the cardboard, the more upward 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 tune both the engine power and the yoke. However, during a normal flight you can simplify this by simply choosing a comfortable engine power level then relying on the yoke and trim for altitude.

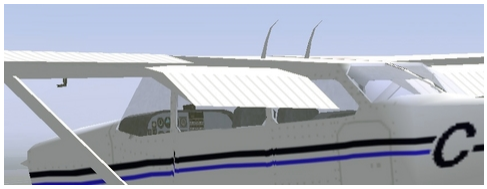
A very interesting exercise you can perform with the simulator is to fly straight with full engine power. Get maximum speed while keeping in horizontal flight. Then decrease the engine power to minimum. Pull steadily on the yoke to keep the plane at constant altitude. The plane slows down steadily, meanwhile you have pull more and more on the yoke to stay level. Since the speed decreases the lift from the wing will 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 flying very slowly, and the nose is very high, you may hear a siren yell. That's the stall warning (see below). This indicates that the angle of attack is too high for the airfoil to produce lift. The wings are no longer producing lift and the plane quickly loses altitude. The only way to correct this is push the yoke forwards to reduce the angle of attach, making the nose drop, then apply full power to gain speed and finally bring the yoke carefully back to level flight.

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

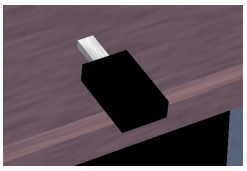
and landing the procedures are quite strict about the use of yoke and throttle. You do the opposite: control the speed with the yoke and trim, control the altitude and descent speed with the engine throttle. This will be discussed further below.

7.7.3 The flaps

The flaps are situated at the rear of the wings, either side of the aircraft fuselage.



You deploy the flaps and retract 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 at a time. 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.)

In the Cessna 172P. there are four flaps settings:

- 0° - for normal flight and normal take off.
- 10° - for short field take off, when you want to gain altitude while flying slowly. Or during the first stange of an approach to land.
- 20° - to slow the aircraft and lose altitude quickly, for example when descending towards the runway to land.
- 30° - To lose altitude even more quickly.

The flaps are somewhat delicate. Do not deploy the first step of flaps above 110 knots. Do not deploy the second or third stage of flaps above 85 knots.

The flaps create large amounts of drag on the aircraft and 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.

To check the flaps position visually, either use the mouse view mode to look at the back of the wing, or type **Shift-right arrow** to shift the view to the right and then quickly **Shift-up arrow** to get back to front view.

Flaps increase wing lift by altering the shape of the airfoil. The wing lifts more at a given speed with the first stage of flaps set. Hence you will get in the air a little sooner during take off. It also has the effect to make the plane fly with a lower nose attitude. This is useful as it provides a better view of the runway when taking off or landing.

The flaps also increase drag on the aircraft. The second and third stage of flaps produce much more drag than lift, so they are used to brake the plane. This is particularly useful when landing, because the airplane glides very well. If you cut down the engine power completely, the plane will descend, yet but 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 that it is better to have a strongly braking plane and lots of engine power, as the plane reacts faster to your commands. Should the engine fail, then just retract flaps as needed and glide to the runway.

What can you do if you have full flaps extended and need to increase your rate of descent further? 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 known as side-slipping, and is a very effective way to lose height progressively as it is easy to stop at any point.

7.7.4 The stall

An aircraft relies on the smooth flow of air over the surface of the wing to produce lift. However, if the wing is at too high an angle of attack, this flow is broken, and the wing no-longer produces lift. With no lift, the aircraft cannot fly, and quickly drops back to earth. This is known as a stall.

A stall is an emergency situation, whatever the While it can happen at any speed, it commonly occurs in slow flight. A given aircraft has a specific stall speed, at which no angle of attack can produce enough lift. You should always keep your aircraft well above the stall speed. To help, aircraft are equipped with stall sirens that sound when the angle of attack is approached.

If you encounter a stall, the remedial action is to immediately drop the nose, and apply full power, bringing the nose level when flying speed has been attained again. However, doing so will cause the aircraft to lose altitude, which you may not have to spare when landing or taking off!

A spin occurs when one wing stalls before the other, which can occur in a steep turn at low speed. As one wing is still flying, the aircraft turns around the stalled wing, spinning tighter and tighter. To get out of a spin, you need to apply rudder to straighten out the spin into a normal stall, then recover as above.

Aircraft like the Cessna 172 and Piper Cub, have benign stalls, and are unlikely to enter a spin. High performance jets, such as the F16 have much more agressive

stalls, and can easily enter a spin.

To practise this in the simulator, do the following:

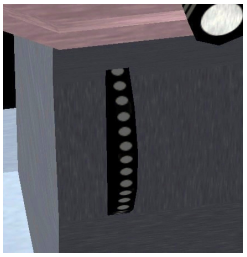
- Fly at constant altitude and attitude.
- Reduce engine power, raising the nose to avoid entering a descent.
- Continue to reduce power until the stall begins.
- Try to control the plane while it stalls and descends to the ground.
- Keep the yoke pulled to the maximum and the plane in a steady attitude, the wings parallel with the horizon. Try to change direction.
- Recover by lowering the nose, applying full power, and correcting the attitude once flying speed has been regained.

You can also experiment with stalls with different flap settings, and high speed stalls by making abrupt attitude changes.

Experiment with different aircraft. Compared with the Cessna 172 the Cessna Citation jet, stalls much more agressively and with little warning..

7.7.5 The trim

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



On *FlightGear*, the keys **Home** and **End** adjust the trim. **Home** rolls the wheel upwards while the **End** rolls the wheel downwards. You can also click on the upper or lower half of the trim wheel.

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 the yoke. The trim remains in position after you make a change, while the yoke only continues to affect the elevator while you apply pressure and returns the elevator to neutral when you release it.

During cruise flight, the required elevator position to keep the aircraft at constand altitude will not be completely neutral - it will vary depending on the air outside the aircraft, the current fuel level, and the payload. Obviously, holding the

yoke continually to retain a constant attitude would quickly become tiring. By using the trim to "trim out" the elevator force required for cruise flight, the yoke can be kept neutral.

During take off the trim should be neutral. Otherwise you may find that it either refuses to take-off with the normal level of yoke control, or takes off too quickly.

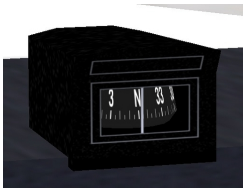
During landing, try to get the yoke/mouse/elevator towards neutral position by tuning the trim. This makes making small adjustments to your attitude and position easier. On the Cessna 172p this means trim on neutral. On the Cherokee Warrior II this means the trim a little "pulled".

The trim wheel movement is much slower than the yoke, allowing for delicate changes in trim. Be patient.

7.7.6 What direction am I flying?

Knowing the direction you are going is obviously a good idea. There are three basic ways to determine the direction you are flying:

- Look through the windows. If you are flying regularly from the same airport, you will learn to recognize the ground features such as roads, hills, bridges, cities, forests. In a simulator, you only have a narrow view of the virtual outside world. Several ways exist to allow you to pan your virtual head inside the airplane:
  - Use **`Shift`** and the four arrow keys to look to the front, rear, left and right.
  - Use **`Shift`** and the keypad keys to look in the four directions mentioned above and in four diagonal directions in-between.
  - Use the mouse in view mode (right button, ↔). This allows you to look in every direction, including up and down. Click the left mouse button to bring the view back to straight ahead.
- The magnetic compass. This is located above the instrument panel. The compass is simple, but is affected by the acceleration of the aircraft, and magnetic abnormalities on the ground. Also, the compass points towards magnetic north rather than true north. This deviation varies depending on your location.



- The directional gyro (picture below) or "heading indicator". The gyro is powered by a vacuum system. The gyro is set to match the magnetic compass, and is not affected by magnetic issues, or aircraft movement. However, due to gyroscopic precession and friction in the instrument, over time it drifts and must be reset by reference to the magnetic compass on ocassion. To reset the HI, during cruise flight, use the black knob on 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").

7.8 Let's Fly

By now you will be able to keep on runway while taking off by using the rudder and you're able to fly straight, descend, climb and make gentle turns. This section will describe a slightly more realistic approach to taking off and landing, and introduce some of the more subtle concepts you should be aware of.

7.8.1 A realistic take off

The following general rules apply during a normal take-off:

- The nose-wheel should be lifted from the runway at approximately 40 knots.
- Immediately after take-off, you should accelerate to 70 knots, to stay well above stall speed, in case of a gust of wind, or engine failure.
- Don't fly much above 75 knots to ensure you gain height as quickly as possible.
- Follow the runway heading until at 500 feet. This way, if you suffer an engine failure, you can easily land back on the runway you left.
- Don't over-fly buildings until at least 1,000 ft.
- Close the the ground, turns should be gentle and well coordinated using the rudder.

So, you need to take off and rise in the air at a steady speed of around 75 knots. However, when you raise the nose slightly at 40 knots, the aircraft will probably take-off at around 55 knots. To accelerate quickly to 75 knots, lower the nose slightly immediately on take-off, then raise it once 75 knots has been achieved. You are using the yoke to control the speed of the aircraft.

Putting this all together with what you have learned previously, a normal take-off using the mouse will consist of the following:

1. Adjust the altimeter to the correct altitude, based on the airport altitude. For reference, KSFO is at sea level - 0ft.
2. Check aileron and elevator are neutral by observing the yoke position.
3. Change the mouse to control mode by pressing the right mouse button.
4. Hold the left mouse button down to control the rudder.
5. Apply full power (hold **PgUp** until the throttle is fully in).
6. As the aircraft accelerates down the runway, keep it in the center by making small adjustments using the mouse.
7. As 40kts is reached, release the left mouse button, and pull back slightly to raise the nose-wheel. You are now controlling the yoke with the mouse.
8. The aircraft will fly off the runway at approximately 55 knots.
9. Lower the nose slightly to accelerate to 70 knots
10. Keep alignment with the runway.
11. Use the yoke to keep the ASI at 70 knots as you climb. If the airspeed is dropping, lower the nose. If the airspeed is increasing, raise the nose slightly.
12. Once you reach 500 feet, turn to your required heading, staying away from buildings until you are over 1,000ft.

7.8.2 Landing

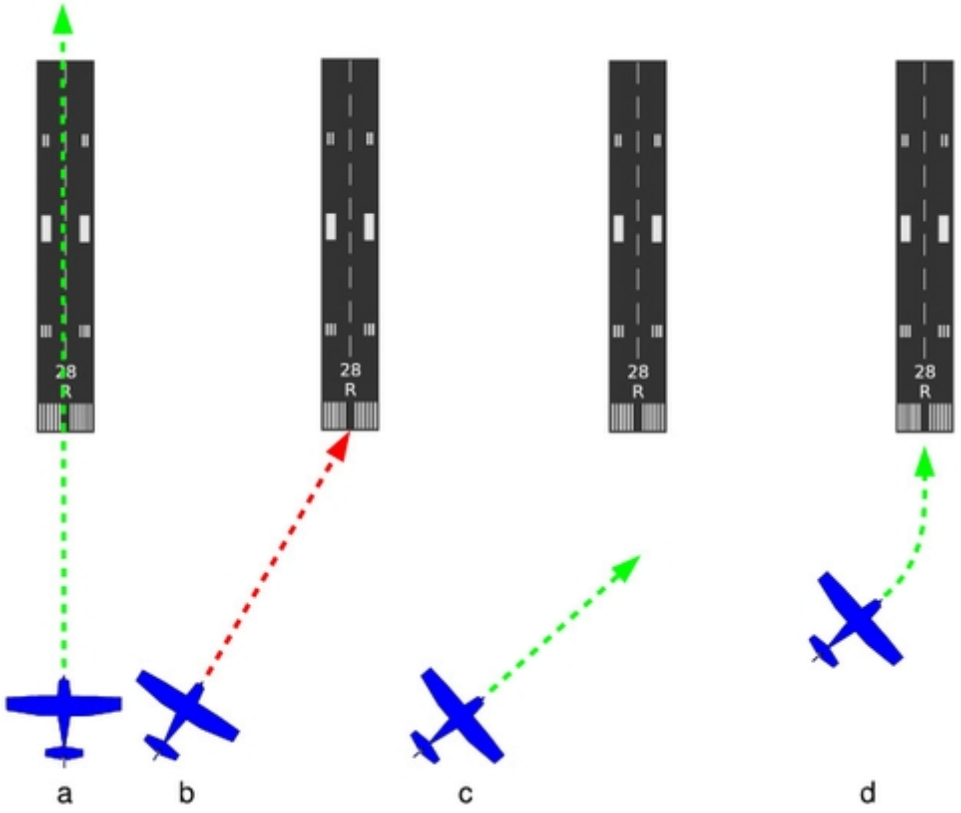
The rules for landing are almost identical to that of take-off, but in reverse order:

- Close the the ground, turns should be gentle and well coordinated using the rudder.
- Stay above 500ft until on final approach to the runway
- Approach the runway at approximately 70 knots.
- Touch down on the two main wheels at 55kts.

- Let the nosewheel touch down at 40kts.

Landings are much easier if you have an aiming point picked out on the runway. By observing the aiming point, you can easily tell if you are descending too fast or too slowly. If the aiming point appears to move upwards, you are descending too fast,

Obviously, you need to be lined up 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 in front 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.



A straight in landing using the mouse would consist of the following:

1. 1,500ft above the airport, and a couple of miles out, an in-line with the runway, reduce power to approximately 1500rpm. This will slow you down somewhat and start a gradual descent.

2. Once below 115 knots, apply one step of flaps (`]`). This will increase lift, but also slow you down.

3. Re-trim the aircraft so you continue to descend.

4. At around 1,000 feet, apply another step of flaps (`]`). This increase drag significantly, but also improve the view over the nose.

5. Tune the speed using the elevator and trim: push the yoke if you are flying below 70 knots, pull the yoke if you are flying above 70 knots. If using a joystick, use the trim to relieve any pressure on the elevator.

6. Tune the altitude using the engine throttle. Add power if you are descending too fast, reduce power if you are too high. It is much easier to work out if you are too high or too low by observing the numbers on the runway. If they are moving up the screen, you are descending too fast - increase power. If they are moving down, you are too high and need to reduce power.

7. Make minor adjustments to heading to keep aligned with the runway.

8. at about 500ft, apply the final step of flaps. (`]`). This increase drag significantly, so be prepared to increase power to keep your descent constant.

9. When you are just above the runway, reduce power to idle, and use the yoke to gently pull back the aircraft to horizontal. This is the "round-out" and should result in the aircraft flying level with the runway a couple of feet above the surface. Performing the round-out at the correct height is a difficult task to master. To make it easier, observe the horizon rather than getting fixated on the aiming point.

10. Keep the wings level using small inputs with the yoke. We want both wheels to touch down at the same time.

11. Continue pulling back on the yoke. The main wheels should touch down at about 55 knots. This is the "flare".

12. As you touch down, be ready to use the rudder to keep the aircraft straight (keypad **`0`** and keypad **`Enter`**)

13. Once you are below 40 knots, lower the nose-wheel to the ground.

14. Hold down the left mouse button to control the nosewheel/rudder using the mouse.

15. Once below 30 knots, use the brakes **`b`** to slow the aircraft further.

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

7.8.3 Engine Shutdown

To shut the engine down:

- Set the parking brake, type **B**.
- 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 **{**).

7.8.4 Aborted Landing

You must be mentally prepared to abort landing any time the landing doesn't look good, or due to external factors such as:

- an order from the control tower
- incorrect speed or landing angle when there is insufficient time to correct it.
- a strong gust blow of wind
- birds flying over the runway.

To abort the landing, apply full power (hold **PgUp**), raise the nose to climb, and once you are climbing, retract the flaps (key[).

Landing is much harder than taking off. Landing on a large runway, such as KSFO (San Francisco, the default) is much easier than smaller runways such as KHAF (Half Moon Bay, about 10 miles to the south west of KSFO).

To practise landings, use the command line below in a terminal window to start the simulator in flight and heading for the runway. The airplane is placed 5 miles ahead of the runway, at an altitude of **1500** feet and a speed of about 120 knots.

```
fgfs -offset-distance=5 -altitude=1500 -vc=120 -timeofday=noon
```

Approaching to land at 65 knots instead of 70 knots allows to use a much shorter runway length. However, this requires better control, particularly as it is much closer to the stall speed. It is quite different from landing at 70 knots.

7.9 Dealing with the Wind

Consider 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, but the balloon itself is completely static in the middle of the cube. Whatever the wind speed, persons aboard a hot air balloon experience not a breath of wind.

In the same way, an aircraft flies in the middle of a gigantic cube of air and flies relative to that air mass. The motion of the cube of air relative to the ground has no effect on the aircraft.

You, the pilot, on the contrary, are interested in 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.

When the wind blows in the opposite direction (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.

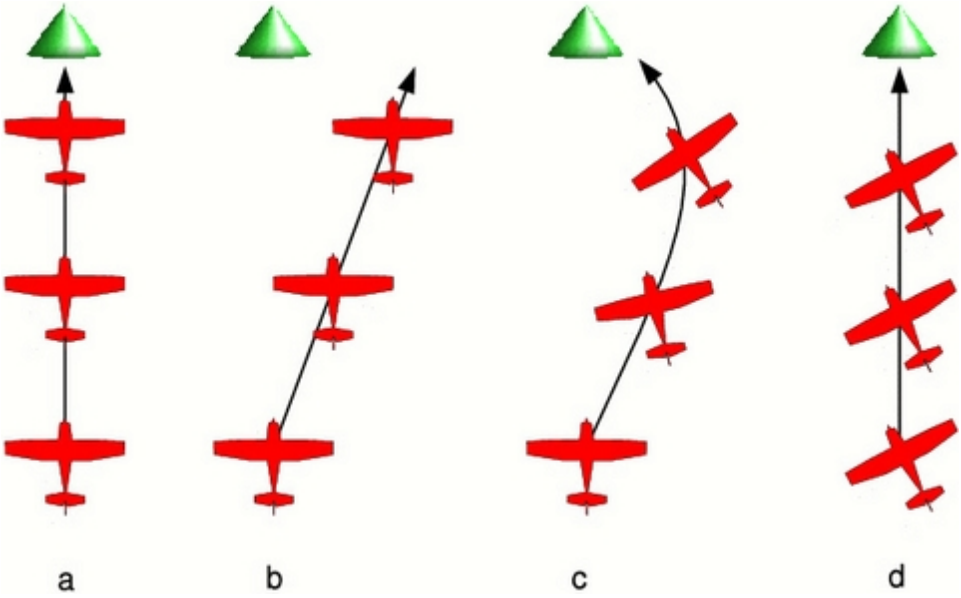
The two cases above are quite simple. More complex is when the wind blows towards the side of the airplane. Consider the diagram below.

- 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 lose time to get to the hill. Such a curved path is awful when you need to make precise navigation.

- 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 and the wind. That way it compensates for the wind and remains on a straight path towards the hill.



How much to the left or to the right of the object must you head? At what angle? Serious pilots use geometry and trigonometric computations to calculate the correct angle. You need no computations at all to fly roughly straight. The trick is to choose an aiming point in the direction you wish to fly, then observe how it moves. You will become aware if 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. To begin with, you may need to think about 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 faster the flight airspeed compared to the wind speed, the less correction you will need.

7.9.1 Crosswind Take Off

Taking off when the wind is coming from the side is tricky. Airport designers avoid this by placing runways so that they face into the prevailing wind. Often airports have multiple runways, placed such that there will be a runway facing straight into wind as much of the time as possible.

Taking off with a wind blowing straight towards the nose of the aircraft makes life easier as it is the speed of the wing relative to the air that causes lift. When there is no wind, the aircraft must accelerate to 55 knots to take off. However, if there is a 10 knot head-wind, the aircraft has an airspeed of 10 knots standing still and only has to accelerate to 45 knots relative to the ground to take off. This shortens take-off distances.

Just as a headwind shortens take-off, a tail-wind increases take-off length. Anything more than a knot or two makes a huge difference to take-off distance. As

(most) runways can be flown from either end, you can easily take off from the other end of the runway and benefit from the headwing.

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:

Unfortunately, sometimes there isn't a runway facing the wind, and you have to take off when the wind is blowing from the side.

The technique is as for a normal take-off with two changes:

- During the take-off roll, the aircraft will try to "weather-cock" into wind. You must react by using the rudder to keep the aircraft running straight. You will have to apply the rudder at quite a strong angle to stay aligned with the runway. You will need to keep applying rudder throughout the take-off.

- As you take off, the aircraft will react to the rudder and try to turn. You will need to correct for this using the ailerons. Once the aircraft is in the air, you can reduce the rudder pressure and aileron, then correct for the wind, to keep aligned with the runway as described above.

7.9.2 Crosswind Landing

Landing in a crosswind is very similar to the take-off:

- Stay aligned with the runway by compensating for the crosswind.

- As you begin to round-out, use the yoke to straighten the aircraft so it is pointed down the runway. Apply rudder to stop the aircraft turning.

- The aircraft will land on one wheel first. Use the rudder to keep the aircraft pointed straight down the runway as the other wheel touches down.

The technique described here is the slip landing. Another crosswind landing technique is the crab landing.

7.9.3 Taxiing in the 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, configure the simulator 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 means that the elevator is almost neutral, while 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.

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 tasks for the pilot. You still are the pilot aboard and have to keep aware of everything. Be prepared to shut the autopilot down as they often go wrong, both in real life, and in the simulator.

The autopilot is mounted to the right of the yoke:

Switch it on by pressing the **AP** button. The autopilot then controls the roll of the aircraft. It keeps the wings level with the horizon. This is displayed in the picture below by the "ROL" marking. To switch the autopilot off press **AP** again.

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 section 7.7.6). "HDG" stands for "heading". Press again on the **HDG** button to get back to roll control mode (or **AP** to switch the autopilot off).

The buttons **ALT**, **UP** and **DN** are used to tell the autopilot either to control the vertical speed VS or the altitude ALT.

For more advanced use of the autopilot, see the reference document for the autopilot modeled in the Cessna 172:

https://www3.bendixking.com/servlet/com.honeywell.aes.utility
.PDFDownLoadServlet?FileName=/TechPubs/repository
/006-18034-0000_2.pdf

7.11 What Next?

This tutorial has introduced you to the basics fo flight in the Cessna 172. From here you can explore the many features that *FlightGear* has to offer.

Once you have mastered the content of this tutorial, you may want to look at the other tutorials in this Manual, covering flying to other airports, flying using instruments when clouds obscure the ground, and flying helicopters.

This tutorial has skipped over a number of topics that a real-life pilot would have to consider:

- How to follow real checklists.
- How to make emergency landing on very short fields, after and engine failure.
- How to navigate with regard to the laws of the air, charts, laws, radio beacons and weather conditions
- How to create a flight plan and fly it accurately.
- How to place people, fuel and baggage 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 tanks and their systems.
- How to deal with the failure of every possible part of the plane.

This tutorial has also not covered features of more advanced aircraft, including:

- retractable landing gear
- variable pitch propellers
- multiple engines
- jet engines.

7.12 Thanks

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 for use in the manual.
- 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 Flying Other Aircraft

I cross-checked all the data about the Cessna 172p, a pilot friend verified I did not write too much rubbish and I made numerous virtual test flights. This section contains less reliable data about other airplanes based on my experience in the simulator. You may find it useful as an introduction to those airplanes but bear in mind my only goal was to make flights that seem OK and acquire basic knowledge.

7.13.1 How to land the Cherokee Warrior II

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 provide more braking and allow it 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 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 use two steps of flaps during landing.Don't decrease the engine throttle too much.
- If you remain at two flaps deployed during landing, the round-out and flar will be similar to the Cessna 172p. However, using the third set of flaps will slow the aircraft down dramatically. 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 descent 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 means 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

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 nose-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. The Piper J3 Cub is a good introduction to tail-wheel aircraft 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). Take-off is below 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 is quite different to that of 172, as the aircraft is very light, and has no flaps.

1. Fly at say 500 feet constant altitude and "exactly" 52 mph speed towards the runway. 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.

2. 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).

3. 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.

4. 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.)

5. 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.

6. 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

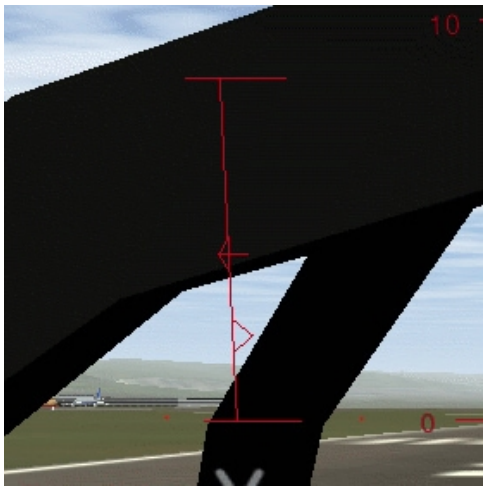
Take off on a jet is easy but you must have fast reflexes. My favorite jet on FlightGear is the A-4 Skyhawk. You get it with the `-aircraft=a4-uiuc` parameter on Linux, provided it is installed.

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).



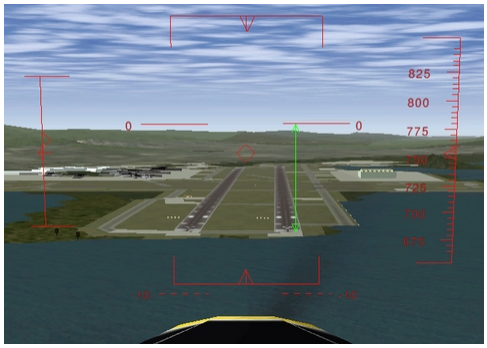
- It is not mandatory to use the rudder to keep on the runway. The airplane will take off before it drifts off the runway. (For sure it is better and more "secure" to keep in the middle of the runway. But using the rudder can make things hectic for a beginner.)
- Once above about 160 knots, the plane 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.

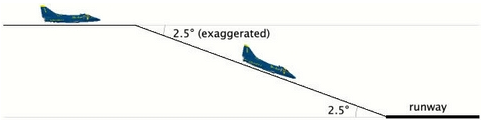
You don't land a jet the same way you land a little propeller airplane. My way to land the A-4, inspired by some texts I found on the Web, is this:

- 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° (that is 25% of 10°):
  - If the angle increases above 2,5°, 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°, 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 ∧ is on, the speed is too fast. The center ◯ 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

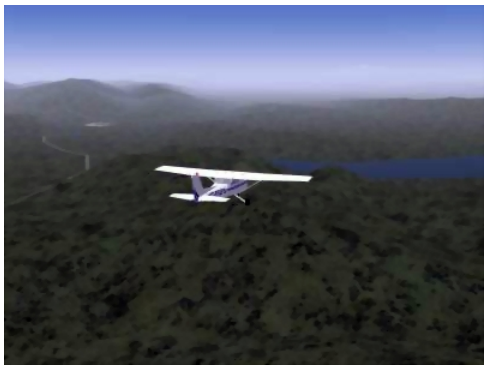


Figure 8.1: Flying over the San Antonio Dam to Livermore

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 fly microlights rather than Cessnas in real life. 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 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.

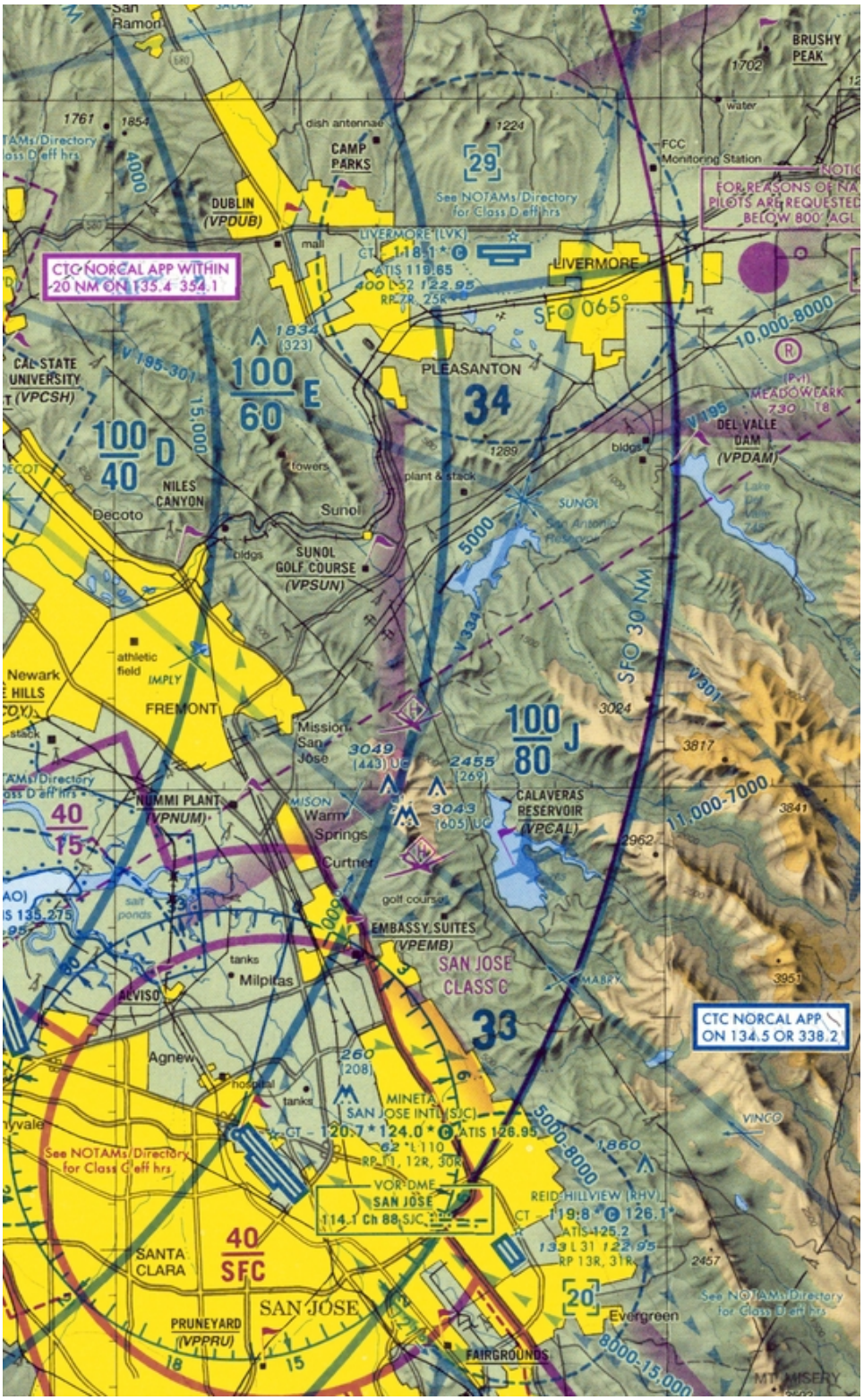


Figure 8.2: Sectional extract showing Reid-Hillview and Livermore airports

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.

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 8.5: COMM1 adjustment knob

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.

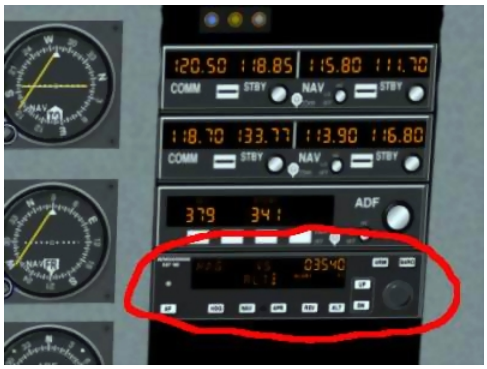


Figure 8.10: The C172 Autopilot

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 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 oxygen. This results in an inefficient fuel burn and a reduction in power because the

Figure 8.11: The Calaveras Reservoir

Figure 8.12: The Calaveras Reservoir

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.

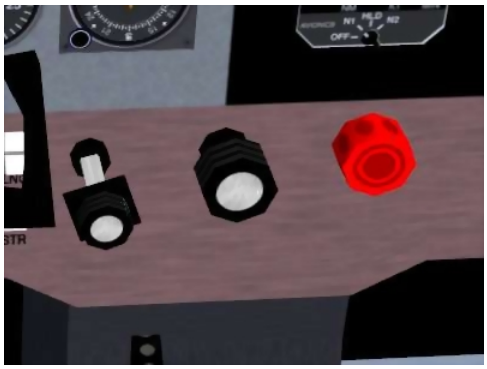


Figure 8.13: Mixture Control

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.

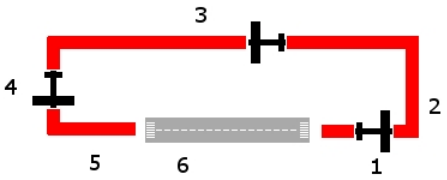


Figure 8.15: The Traffic Pattern

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

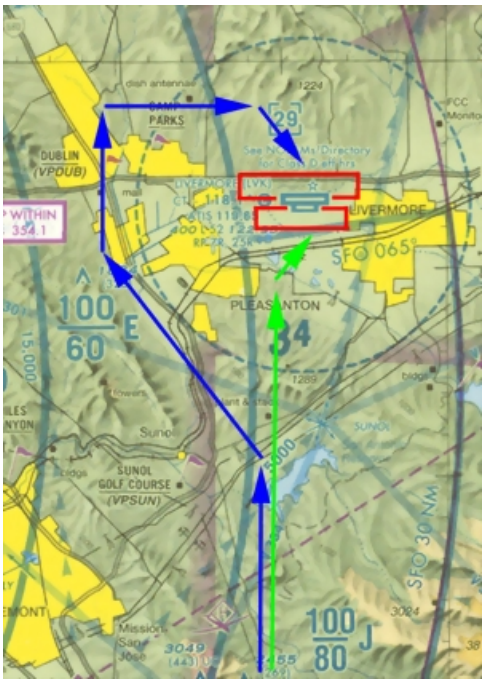


Figure 8.16: Sectional extract showing approaches to Livermore

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 :).

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

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

Figure 8.17: On Final at Livermore with VASI on the left

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

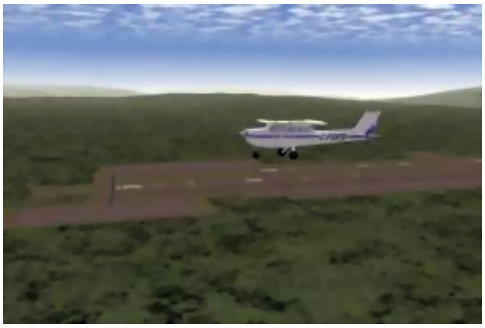


Figure 8.18: Missed approach at Livermore

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).
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

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.

Chapter 9

An IFR Cross Country Flight Tutorial

9.1 Introduction

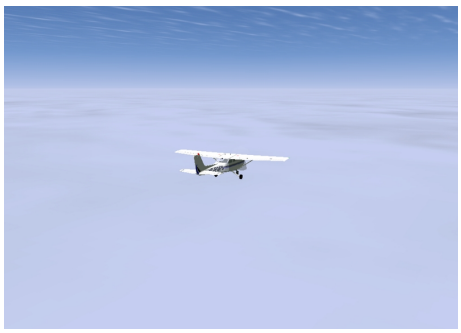


Figure 9.1: Flying over the San Antonio Dam to Livermore. I think.

In the cross country flight tutorial, you learned about VFR flight, and in the course of the flight you were introduced to most of the flight instruments in the C172p. Now we're going to do an Instrument Flight Rules (IFR) flight. In this flight you'll be introduced to the remaining instruments, learn a bit about IFR flight, and learn many, many TLAs (Three-Letter Acronyms).

We'll fly the same flight, from Reid-Hillview (KRHV), runway 31R, to Livermore (KLVK), runway 25R, only this time we'll do it in IFR conditions: a 750 foot ceiling, and 1 mile visibility. This tutorial assumes you've completed the cross country flight tutorial.

9.1.1 Disclaimer

This is not intended to teach you how to fly IFR. Rather, it is meant to give a flavour of what IFR flying is like, and remove the mystery of the panel instruments not covered by the cross country flight tutorial.

I'm not a pilot. Like the previous tutorial, this information has been gleaned from various non-authoritative sources. If you find an error or misunderstanding, please let me know. Mail me at bschack-cocoa -at- usa -dot- net.

9.2 Before Takeoff

We need to tell FlightGear about our flight conditions. Depending on which version of FlightGear you're using there are different ways to set our "desired" weather. You need to check your specific version to see what you need to do. In the command-line version, you need to add the following options when you start up:

```
--ceiling=750:3000 --visibility-miles=1
```

The first option asks for a 750 foot ceiling, with a thickness of 3000 feet (ie, the cloud tops are at 3750), the second should be obvious. I strongly recommend also setting the following, admittedly hairy, option:

```
--prop:/environment/config/aloft/entry/visibility-m[0]=30000
```

The option's not pretty,[1] but the result is — when you top out at 4000 feet, you'll be able to see for miles (30,000 metres, in fact), including a few peaks poking through the clouds (without this option, you'd have 1 mile visibility above the clouds as well as below them). It's a small thing, and not strictly necessary, but there's something very uplifting about rising above the clouds into a bright sunny world, with blue sky above and a carpet of white below, like Figure 9.1.

9.2.1 Flight Planning

Unfortunately, when you start, you won't see a carpet of white. You'll see something more like Figure 9.2. Those clouds don't look very friendly, and it's hard to even see past the end of the runway. Maybe we should just *drive* there in the Cessna. We had been planning to practice ground steering anyway ...

So how do you get from A to B when you can't see? There are a variety of ways that have evolved over the years, with various advantages and disadvantages. Our flight will use all of the navigation instruments the standard Cessna C172p has, just to give a taste of what's possible.

[1] In fact, you might have to make it even uglier, if you use csh or tcsh. They'll complain about the square brackets, so you'll need to escape them: `...visibility-m\[0\]...`. If you really don't like this, you can also set this option from the FlightGear menu: click **Weather** ⇒ **Weather Conditions**, and set the visibility in the 3000 foot layer to 30000.

Figure 9.2: On the runway at KRHV

Our entire route, and the aids we'll be using, are shown in Figure 9.3. Our route is in green, the navigational aids blue and red. The route looks a bit crazy — in fact, you might wonder if we're *more* lost using our fancy equipment than just flying by the seat of our pants — but there is a method to the madness. Rather than overwhelming you with details by explaining it all now, I'll explain it bit by bit as we go along.

9.2.2 VHF Omnidirectional Range

The first bit will involve VOR[2] (VHF (Very High Frequency) Omnidirectional Range) navigation, and will get us to a point about 5 nm (nautical miles) south of Livermore.

VOR stations are indicated on the sectional by a big bluish-green circle with compass markings around the outside. I've helped you by marking their centers with a big blue dot as well. Reid-Hillview is very close to one, San Jose, which you can see in Figure 9.3. Near the centre of the circle, in a bluish-green rectangle, is the station information. According to the station information, it's is a VOR-DME station (I'll explain DME later), its name is San Jose, its frequency is 114.1 MHz (or Channel 88, which is an alternative way to say the same thing), and its identifier, or "ident", is SJC (which in Morse code is ··· ·––– –·–·.

To tune into a VOR station, we use one of the NAV receivers, which are paired with the COMM receivers (see Figure 9.4). And we navigate using the corresponding VOR gauge. We'll choose the NAV1 receiver (and VOR1 gauge) in this case (NAV2 would have worked just as well). Before setting the frequency, check out the VOR1 gauge. It should look like VOR1 in Figure 9.4. The important thing is the red and white flag. That means there's no VOR signal, so we can't trust the gauge.

The NAV receiver has an active frequency, a standby frequency, and a tuning knob, just like the COMM receiver. Tune it to 114.1, and press the swap button. If

[2] See http://en.wikipedia.org/wiki/VHF_omnidirectional_range for more information.

Figure 9.3: Green: our route, Blue: VORs and radials, Red: NDBs

Figure 9.4: IFR navigation Instruments

Figure 9.5: VOR1, after tuning

you look at VOR1, you should notice that the red and white flag has disappeared, to be replaced with a "TO" flag, as in Figure 9.5. That means we're receiving a signal. But is it the correct one? What if we accidentally set the wrong frequency? What if there's a different VOR nearby with the same frequency?

To confirm that we're tuned into the correct VOR, we listen for its ident. If you can't hear the ident, or if it doesn't match the chart, don't trust the needle. So far, you probably haven't heard a thing. That seems strange. After all, the red and white flag turned off. We must be in range of *some* station, right? Why can't we hear anything?

When in doubt, look for the simplest solution. In this case, check the volume. There's an extra dial on the NAV receivers that the COMM receivers don't have, labelled OFF, LO, and HI (see Figure 9.6). That's our ident volume control. So, is it currently off, or low, or high? Unfortunately, it gives no visual feedback about its current state, so we don't really know. The only way to be sure is to turn it up. Unfortunately, for the same reason, it's hard to tell if our mouse clicks are doing any good, so hit Ctrl-C to see the hot-spots. This control has two hot-spots, one on the top which increases the volume, the other on the left which decreases it. Increase the volume. *Now* you should hear the ident: ··· ·––– –·–·. Phew.

Back to VOR1. There's a knob labelled OBS (Omni Bearing Selector). As the name vaguely suggests, it is used to select a bearing. If you turn it, you should see the vertical needle, called the CDI (Course Deviation Indicator) move.[3] Try to center the needle. It should center when the little arrow at the top points to somewhere around 277. That number, and the TO flag means: "Flying at a heading of *277*° will lead you directly *to* the station".

That's great, except, according to our route, we don't want to go *to* the station. We actually want to intercept the light blue line labelled "009°" (the "9 degree

[3]The horizontal needle is used in ILS landings, which will be explained later.

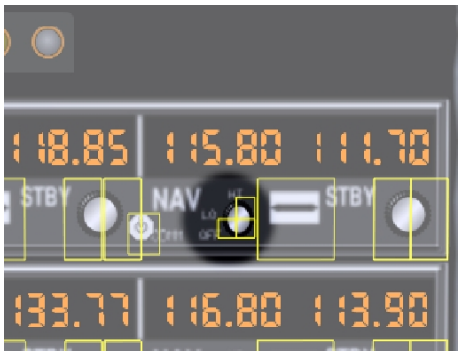


Figure 9.6: Ident volume knob and hot-spots

radial") coming *from* the station. How do we do that? Simple. Set the OBS to 9. When we fly across the radial, the needle will center, and the flag will say FROM. This tells us: "flying at a heading of 9° will lead you directly away *from* the station", which is what we want. At that point we'll turn right to a heading of 9°.

One final thing — set the heading bug on the directional gyro to our current heading (about 310°).

9.2.3 Takeoff

Now we're ready to take off. It took a long time to get ready, but in fact there's really more that we should have done. For example, there are all the things mentioned in the previous cross country flight tutorial. And there are other IFR preparations that we should have made. But again, in the interests of not overwhelming your brains, I'm feeding out the information in trickles. This brings us to the most important control you have — the 'p' key. Use this often, especially when a new concept is introduced.

Okay. Now take off, keeping a heading of 310° for now. Establish a steady rate of climb. We plan to climb to 4000 feet. There's just one problem though — those ugly looking clouds are standing in our way.

9.3 In the Air

If this is your first attempt at IFR flight, you will find it nearly impossible to fly once you enter the clouds. When you enter the clouds, you will be momentarily disconcerted by the lack of visual cues. "No matter," you then think. "I'll just keep things steady." In a few moments, though, you'll probably notice dials and needles spinning crazily, and without knowing it, you'll be flying upside down, or diving towards the ground, or stalling, or all three.

Figure 9.7: Autopilot after engaging

It takes practice to get used to flying without external visual clues, although it's a skill that you definitely *must* master if you want to fly IFR. For now though, we'll use "George", the autopilot, to make this part of flying easier.

9.3.1 George I

Once you've established a steady rate of climb and heading, engage the autopilot by pressing the AP button. You should see "ROL" displayed on the left to show that it's in "roll mode" — it is keeping the wings level. In the middle it will display "VS", to show it is in "vertical speed" mode — it is maintaining a constant vertical speed. On the right it will *momentarily* display that vertical speed (in feet per minute). Initially, the value is your vertical speed at the moment the autopilot is turned on. In the case of Figure 9.7, the autopilot has set the vertical speed to 300 feet per minute.

When you engage the autopilot, CHECK THIS CAREFULLY. Sometimes the autopilot gets a very funny idea about what your current rate of climb is, like 1800 feet per minute. Our little Cessna cannot sustain this, and if the autopilot tries to maintain this (and it will), you will stall before you can say "Icarus". This is a bug, to be sure, and a bit annoying, but it is also a useful cautionary lesson — don't put blind faith in your equipment. Things fail. You have to monitor and cross-check your equipment, and be prepared to deal with problems.

We want a vertical speed of around 500 to 700 feet per minute. Hit the up and down (UP and DN) buttons to adjust the vertical speed to a nice value. Take into account the airspeed as well. We want a sustainable rate of climb.

Finally, once you're climbing nicely, hit the heading (HDG) button. On the display, "ROL" will change to "HDG", and the autopilot will turn the airplane to track the heading bug. Since you set the heading bug to the runway heading, and you took off straight ahead (didn't you?), it shouldn't turn much.

9.3.2 MISON Impossible

It's around 8 nm to the 009 radial intercept, so we've got a bit of time. Since there's no scenery to admire, we might as well prepare for the next phase of the flight.

If you look along our route, just after we intercept the 009 radial and turn north, we pass by a point labelled MISON (see Figure 9.9 for a closeup of that section of the chart without my fat blue and green lines drawn on top. MISON is in the lower

Figure 9.8: Typical IFR scenery

right). Just above and to the left of MISON are two crossed arrows. MISON is an intersection. We're actually going to pass east of MISON, but the radial passing roughly from northwest to southeast through MISON (and our route) is of interest to us. We're going to use it to monitor our progress.

Noting our passage of that radial isn't strictly necessary — we can just keep flying along the 009 radial from San Jose until we need to turn. But it's useful for two reasons: First, it's nice to know exactly where we are. Second, it confirms we are where we think we are. If we fly and fly and never cross the radial, alarm bells should start going off.

Looking at the sectional, we see that the radial is the 114 radial from the Oakland VORTAC (VOR TACAN, where TACAN stands for Tactical Air Navigation). Oakland's frequency is 116.8, and its ident is OAK (−−− ·− −·−). NAV2 should already be tuned to Oakland, but if it isn't, do it now. Turn on NAV2's volume and make sure you're getting the correct ident.

When we were on the ground, VOR2's needle was flickering — VOR signals are line-of-sight, and Oakland was too far away (and behind some hills) for us to receive it clearly. Soon after taking off, however, it should have settled down, and now it should be holding steady.

We need to adjust the OBS, to tell VOR2 which radial we're interested in. Set the OBS to 114. See if you can guess whether the flag should read TO or FROM when we cross the 114 radial. And see if you can guess whether the needle will move from left to right or right to left as we cross the radial.

A final note: There's nothing magical about the 114 radial — we could have used 113, or 115, or 100, or 090. The reason I chose 114 is because there was a line on the map already drawn along the 114 radial, which saved me the trouble of drawing a line myself.

9.3.3 George II

As we continue towards the 009 radial intercept, let's look a bit more closely at the autopilot. First of all, if you aren't in the habit of trimming the airplane, you'll

Figure 9.9: Oakland VOR and 114 radial to MISON intersection

Figure 9.10: Autopilot with altitude armed

probably notice a flashing "PT" with an arrow on the autopilot. The autopilot is telling you to adjust the pitch trim. I tend to ignore it because, flying with a mouse, trimming is more trouble than it's worth. Those of you lucky people with yokes and joysticks and who find flashing lights annoying might want to trim to get rid of it.

Also, on the right there's a big knob, the altitude select knob, which we can use to dial in a target altitude. We're going to use it. Turn it until you see our desired cruising altitude, 4000 feet, displayed on the right. When you started turning it, "ALT ARM" should have appeared in the autopilot display (as in Figure 9.10). This indicates that you've selected a target altitude. The autopilot will maintain the current rate of climb until reaching that altitude, at which point it will level off and change from vertical speed (VS) mode to altitude hold (ALT) mode. In altitude hold mode it maintains an altitude (in this case our target altitude of 4000 feet).[4]

Don't forget that the autopilot won't adjust the throttle, so when it levels out, the airplane (and engine) will speed up to potentially dangerous levels. You'll need to adjust the throttle to get a proper cruise.

9.3.4 Staying the Course

I assume by this point you've turned onto the 009 radial. Unless you're good or lucky, the needle probably won't be centered. We need to adjust our course. The CDI needle (the vertical needle on the VOR) tells us where to go. If it's to the left, that means the radial is to the left, so we need to go left. Ditto for right.

It's quite easy in theory, although in practice you may find that it's hard to keep the needle centered, and that you are slaloming down the radial. The key is to realize this: the *position* of the needle tells us where we *are*, the *motion* of the needle tells us what to *do*.

I'll explain. If the needle is to our left, then, yes, the radial is definitely to our left.[5] But if the needle is *moving* towards us, that means we're going to cross the radial, sooner or later, so our situation is improving, and we probably just need to wait for the needle to center. On the other hand, if the needle is *moving* away, we need to turn towards it to stop, and reverse, its motion.

[4] Of course, you don't really have to do this — you could just watch the altimeter, and when it gets to 4000 feet, reduce the vertical speed to 0, or press the ALT button to enter altitude hold mode. But by using the altitude select knob, we've demystified one more mystery button.

[5] Unless you're heading in the opposite direction, but that's another story.

Note that the amount we need to turn is difficult to guess correctly at first, so experiment. Try 10°. If the needle moves too fast, cut it down to 5° (ie, turn back 5°). If, on the other hand, the needle moves too slowly, double it to 20° (ie, add another 10°), and see what happens.

9.3.5 Yet More Cross-Checks

Cross-checking your position is always a good thing. The intersection with the Oakland 114 radial was one way (I assume that by now you've crossed it). Ahead of us lies the SUNOL intersection. If you look closely, 5 separate radials join at the point, so we have an embarrassment of choices with regards to the intersecting radial. Because it will come in useful later, we're going to use the one coming in from the upper right. Another check of the sectional reveals that this is the 229 radial of the Manteca VOR, 116.0 MHz, ident ECA (· −·−· ·−).

You should know the drill by now: Tune NAV2 to 116.0, set the OBS to 229, and check the ident to confirm the station.

Meanwhile, let's introduce another piece of gear on the panel that will cross-check the SUNOL passage. Some VOR stations have a distance capability, called DME[6] (Distance Measuring Equipment). For example, San Jose does (remember it's a VOR-DME station), as does Oakland (VORTAC also means it has DME capabilities). It turns out Manteca does as well.

Using DME, you can find out how far you are, in straight-line distance, from the VOR station. In our scenario, the DME isn't necessary, but we'll use it anyway, just to see how it works, and to reconfirm our position.

The DME is the instrument below the autopilot (refer to Figure 9.4). Probably the selector is set to N1. The N1 means "listen to NAV1", and since NAV1 is tuned to San Jose, it's telling us the distance to the San Jose VOR-DME. Now switch the DME to N2. It now shows us the distance to the Manteca VOR.

The DME shows you 3 things: the distance in nautical miles to the station, your speed towards or away from the station, and estimated time to the station at the current speed. Note that the distance is the direct distance from your plane to the station (called the "slant distance"), not the ground distance. Note as well that the speed is relative to the station, so unless you're flying directly to or from the station, it will probably be lower than your true groundspeed. For example, the speed from San Jose, which is directly behind us, should be greater than the speed towards Manteca, which is off to the right.

If we look up information about the SUNOL intersection,[7] it tells us that it is 33.35 nm (as measured by a DME receiver) from ECA on the 229.00 radial (that's what "ECAr229.00/33.35" means).

Now we have two ways to confirm the SUNOL intersection: The VOR2 needle will center, and the DME will read 33.4 or so. Note that the DME doesn't provide us with a very precise fix here because Manteca is at such an oblique angle. But it

[6] See http://en.wikipedia.org/wiki/Distance_Measuring_Equipment for more information.

[7] For example, from http://www.airnav.com/airspace/fix/SUNOL.

does give us a good warning of SUNOL's impending arrival. Moreover, if it has an unexpected value (like 30), it should raise a few alarm bells.

You may be wondering what "HLD" means (the setting between N1 and N2 on the DME). It stands for "hold", and means "retain the current frequency, regardless of whether NAV1 or NAV2 are retuned". For example, if we switch from N2 to HLD, the DME will continue to display (and update) information to Manteca. Even if we retune NAV2, the DME will remain tuned to Manteca. This is handy, because it basically represents a third independent receiver, and in IFR flight two receivers just never seem like enough.

9.4 Getting Down

We're getting close to SUNOL, flying along the 009 radial from San Jose, monitoring our position with the DME. At SUNOL we'll be less than 5 nm from Livermore, somewhere down there in the clouds. Perhaps if we just descended to 700 feet or so (Livermore is at 400, the ceiling is at 750) and headed more or less directly north after SUNOL, we'd get there? A recipe for disaster my friend, and you know it.

As you recall from the previous tutorial, when flying VFR, you don't just point your airplane to the nearest runway to land. You need to fly a pattern. This helps you line up, and helps prevent planes from crashing into one another, which is a Good Thing.

9.4.1 Instrument Approach Procedures

Similarly with IFR landings. There's a procedure to follow. In fact, there are *procedures* to follow. Because of the complexity of landing in IFR conditions, there's no single procedure for all airports. You need to check for your particular airport. In fact, you usually need to check for your particular airport, runway, and navigation equipment.

Our airport is Livermore (KLVK). Let's check the information for that airport. Go to http://www.airnav.com/airport/KLVK to see what they've got. Down near the bottom, we have IAPs (Instrument Approach Procedures). There are two listed for runway 25R. One is an ILS (Instrument Landing System) approach, the other a GPS (Global Positioning System) approach. Our plane has no GPS, but it does have ILS capabilities (I'll explain ILS later), so we'll choose that.

Although Livermore only has two different instrument approach procedures, big airports have many many more. If you look at nearby San Francisco, you'll see they have a *slew* of procedures. There are ILS procedures, GPS procedures, LDA procedures, VOR procedures, ... I wouldn't be surprised if they had a procedure for someone with a sextant and an hourglass in there. To learn IFR flight, you'll need to master all of them.

Back to Livermore. If you download the procedure, you'll see something like Figure 9.11 (except for the colour). It's pretty overwhelming at first — it com-

presses a lot of information in a small space. We'll ignore as much as we can, restricting ourselves to the three parts that have been coloured in. And we'll do those parts on a "need to know" basis — we'll only look at them when we really have to.

Where to start? At the beginning of course. An IAP will have one or more Initial Approach Fixes (IAFs). These are your entry points to the approach procedure and can be found in the "plan view", which I've coloured purple in Figure 9.11. Our IAP lists two, one in the middle and one on the right (see Figure 9.12 for a close-up).

An IAF is a *fix*, and a fix is an identifiable point in space. In fact, we've already encountered another kind of fix, namely a VOR intersection. Fixes are also usually named (eg, MISON, SUNOL). The IAF on the right is named TRACY, and consists of a radial, a distance, and an altitude. Specifically, it's 15 DME (15 nm as measured by a DME receiver) along the 229 radial from the ECA (ie, Manteca) VOR.

9.4.2 Nondirectional Beacons

However, we're not going to use TRACY as our IAF. We're going to use the IAF in the middle, which is a marker (LOM stands for "Locator Outer Marker"). We'll worry about what an outer marker is later. For now let's concentrate on the locator part. The locator in an LOM is an NDB[8] (nondirectional beacon). It's a bit like a VOR, in that it can be used to determine your heading and navigate from place to place. Like a VOR, it has a name (REIGA, in this case), a frequency (374 kHz), and an ident (LV, or ·–·· ···– in Morse). NDBs also appear on sectionals, as fuzzy red circles with a small circle in the middle, with their identification information placed in a red box nearby. (see Figure 9.13 for a closeup. Don't confuse the NDB, which is fuzzy, with the solid red circle on the left, nor the circle below with the "R" inside).

An NDB station basically broadcasts a signal that says "I'm over here", and the receiver on the plane can receive that signal and tell you, the pilot, "the station is over there". You just need to tune the receiver and monitor the correct instruments. The receiver, labelled ADF (Automatic Direction Finder), and the corresponding instrument, also labelled ADF, are shown in Figure 9.4.

To tune into REIGA, turn the tuning knob on the receiver until 374 is displayed as the standby (STDBY) frequency. As usual, use the middle mouse button for big changes (100 kHz in this case), and the left mouse button for small changes (1 kHz). Then hit the SWAP button. The 374 is now displayed as the selected (SEL) frequency. The needle on the ADF should swing around, eventually pointing ahead to the right. This shouldn't be too surprising, since that's about where the REIGA NDB is. Like VORs, to be sure we're really tuned into the right station, we need to hear the ident as well.

[8] See http://en.wikipedia.org/wiki/Non-directional_beacon for more information.

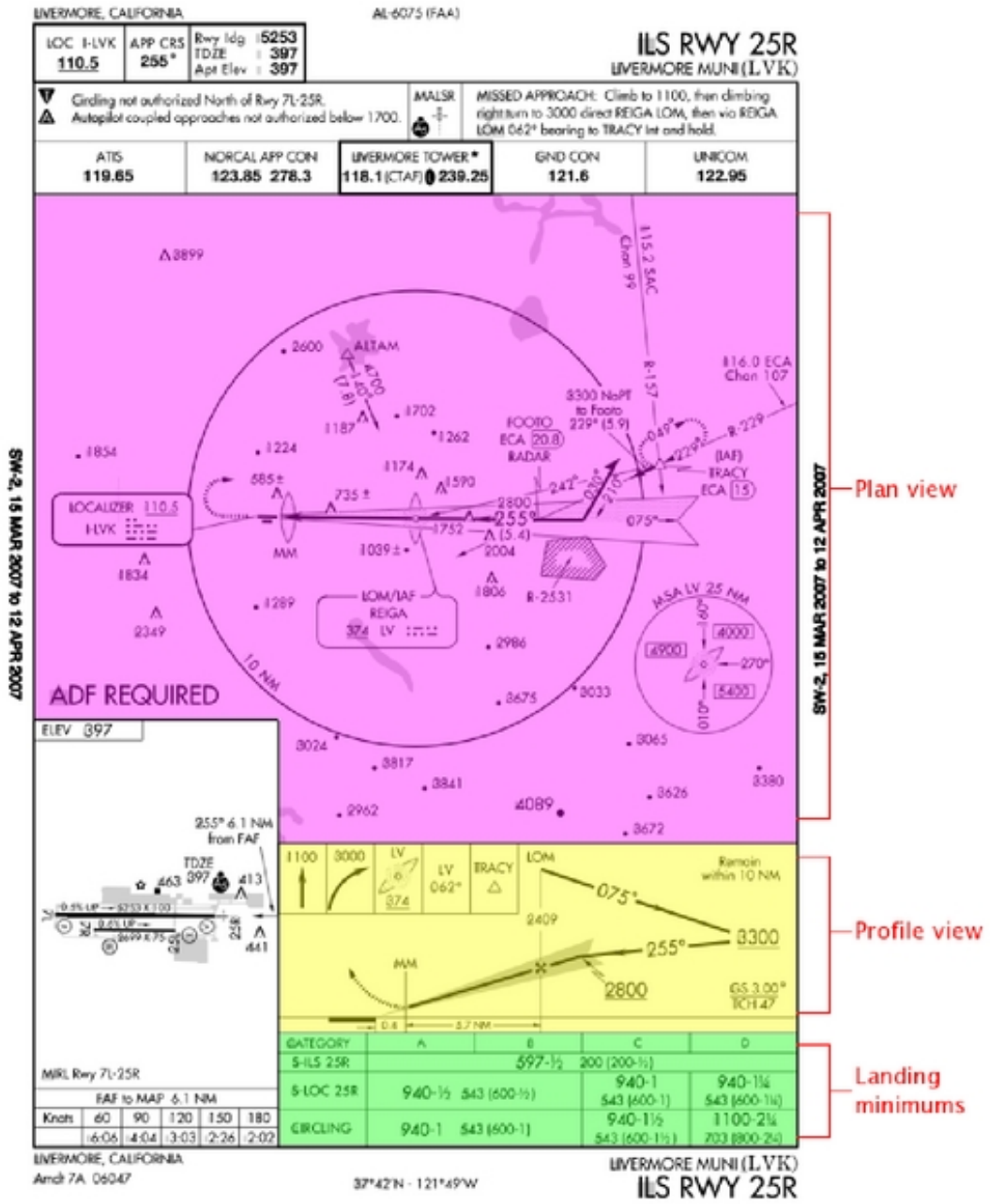


Figure 9.11: ILS approach plate for Livermore runway 25R

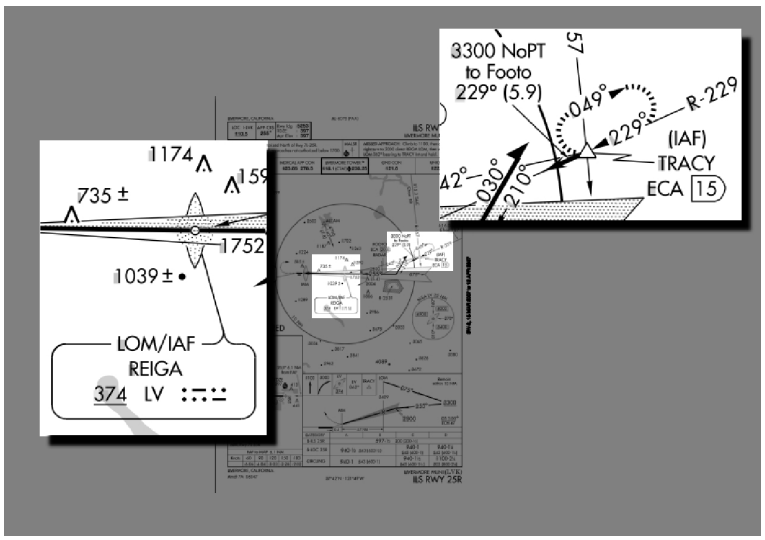


Figure 9.12: Initial approach fixes

Figure 9.13: REIGA nondirectional beacon

Unlike the NAV receivers, the ADF receiver has a dedicated IDENT switch. Turn it on, and wait a bit. You should hear something. If you don't, though, it's possible that the volume is turned off. Turn the ADF volume up to high (the ADF volume switch works the same as the NAV volume switch, with the same problems). If you still can't hear the ident, you've either dialled in the wrong frequency, you haven't waited long enough, or you're really, really lost.

Notice there's no OBS to set for an ADF — the needle just points to the station, which is nice. This leads us to our first rule for ADFs:

ADF Rule 1: The needle points to the station.

Pretty simple. In fact, you may not think it merits a "rule", but it's important to emphasize the difference between ADFs and VORs. A VOR, remember, tracks a single radial, which you specify by turning the OBS. An ADF has a knob, and a identical-looking compass card, so it's tempting to believe it acts the same way. It doesn't. Turn the ADF heading knob and see what happens. The compass card moves, but the arrow doesn't. It just *points to the station*.

In our current situation, where we just want to fly to REIGA, that's all we need to know to use the ADF. If the needle points "over there", then we'll fly "over there", and eventually we'll pass over REIGA. However, for the sake of practice, and because it will be necessary later, I'm going to give the second rule for ADFs, which explains what the compass card is there for:

ADF Rule 2: *If* the compass card reflects our current heading, then the needle gives the bearing *to* the station.

In other words, the compass card gives "over there" a number.

Now we're ready to head to REIGA. Rotate the ADF heading knob until our current heading is at the top (basically, the ADF should match the directional gyro). When we pass the SUNOL intersection, look at the ADF needle, and set the DG bug to that heading (I assume you're using the autopilot. If not, just turn to that heading). At the end of the turn, the ADF needle should point straight ahead. And if it doesn't, adjust your heading so that it does.[9]

By the way, the closer you get to REIGA, the more sensitive the needle becomes to changes in your heading. Don't go crazy trying to keep the needle centered as you get close. Maintain a steady heading, and get ready for the ...

9.4.3 Procedure Turn

So, once we hit REIGA, do we just turn left and head down to the runway? Ah, if only life were so simple. No, we turn right, *away* from the airport, and do a *procedure turn*. We know there's a procedure turn because of the barbed arrow in

[9] Which is actually bad technique in the presence of a crosswind, but I'm ignoring the wind to simplify the tutorial.

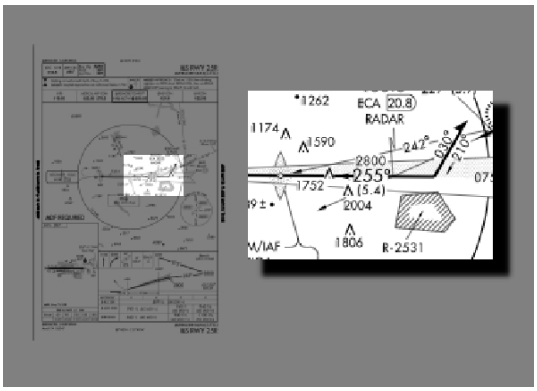


Figure 9.14: Livermore ILS procedure turn

the plan view (see Figure 9.14). As you can see if you follow the arrow, we need to fly away, on a heading of 075°, then turn left 45° to a heading of 030°. We do a U-turn (to the right, *away* from the airport — that's one of the rules about procedure turns) to come back at 210°, then a 45° right turn to 255°, heading straight towards the runway. All of this turning gives us time to set ourselves correctly on course, at the right altitude, to land on 25R.

Hmmm. I mentioned "right altitude", but how do we know that is? That's down below, in the profile view (the yellow part of Figure 9.11). You can see that at the top is the LOM, our IAF. Now follow the arrows. After the IAF, we head out at 075°. During the procedure turn we can descend to 3300 feet, but *no lower* (that's what the line *under* the 3300 means). After we finish our procedure turn and are heading back at 255°, we can descend to 2800 feet, but *no lower*, until we intercept the glide slope.

One thing the instrument approach procedure does *not* tell you is the length of the procedure turn. The only constraint is that you must not fly more than 10 nm away from the NDB. You'll notice there's a 10 nm circle drawn around it in the plan view, and a note in the profile view saying "Remain within 10 NM". They're not kidding. So, since we fly at around 110 knots, two minutes on each leg is reasonable — two minutes at 075°, and two minutes at 030°. On the way back we don't care about times — we just want to intercept 255°.

So, after we pass REIGA, turn right to 075° and start your timer.

9.4.4 Chasing the Needle

When we approached REIGA, we weren't particularly concerned about our course — we just aimed for REIGA. Now, however, our course is important. We want to be flying directly away from REIGA *on a course of 075°*.

Now, in an ideal world, after we turned to 075°, the ADF needle would be pointed directly behind you (ie, we'd be on course). Probably it isn't, so we need

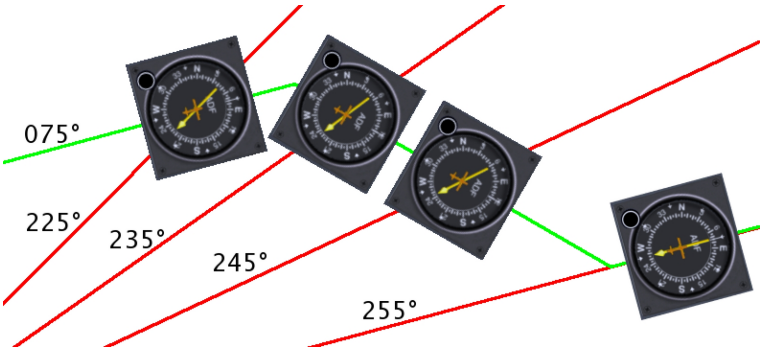


Figure 9.15: Getting back on course

to adjust our course. The key to adjusting our course is ADF Rule 2. If we've set the compass card correctly, then the needle shows us the current NDB "radial". If we turn and fly until we intercept the 255 "radial", then turn to 075°, we'll be right on course.

Figure 9.15 shows what I mean. In the figure, the plane, flying along the green line, is initially off course.[10] The heading is correct, 075°, but the station is at 225°, not 255°. To correct this, we turn right (remembering to adjust the ADF compass card). As we fly on this new heading, we get closer to the correct position, crossing the 235 and 245 "radials" (shown in red). Finally, when we the ADF needle points to 255°, we turn left to 075°, and readjust the ADF compass card.[11] We are now on course.

Of course, even when you get back on track, that won't be the end of the story. Your airplane drifts; your mind drifts; your compass drifts; the wind pushes you around. What you find is that you will be constantly making little corrections. That's okay, as long as we're close. And anyway, before long (2 minutes actually), we'll turn left 45° to 030° as part of our procedure turn, at which point we'll just ignore the NDB anyway. Sigh. All that effort for just 2 minutes. Hardly seems worth it.

9.4.5 FOOTO Time

While you're flying outbound, take an occasional look at VOR2, tuned to Manteca, and the DME. Assuming the OBS is still at 229, and the DME still tuned to N2, at some point the needle should center, meaning you've crossed the 229 radial, and, if you're on course, at the same time the DME should read 20.8. How do I know that? If you look at the approach plate (Figure 9.11), you'll notice an intersection,

[10] *Way* off course, actually. I've exaggerated the angles to make the explanation clearer.

[11] You might be thinking "Wouldn't it be nice if there was an ADF where the compass card rotated automatically?" Well, such an ADF does exist, and it comes with its own acronym — RMI (Radio Magnetic Indicator).

named FOOTO. FOOTO is on the approach, and is defined to be 20.8 DME from ECA. Although this intersection is not strictly necessary for us, it comes for free, and provides good confirmation of our position both outbound and, later, inbound.

Depending on how fast you're flying, you'll probably pass FOOTO close to the time your two minutes at 075° are up. At the end of two minutes, turn left 45° to 030°. We'll fly for another two minutes on this heading.

9.4.6 George III

This leg is relatively uneventful, so we'll take advantage of the lull in the action to descend to 3300. Assuming you're using the autopilot, you will need to do a few things:

1. If you're in altitude hold (ALT) mode, you need to get back into vertical speed (VS) mode. Press the ALT button — the "ALT" in the middle of the display should change to "VS", and your current vertical speed (probably 0) should be displayed momentarily on the right.

2. Click the DN button until you get a vertical speed of -500 feet per minute.

3. If you want to set the target altitude, like before, rotate the big knob on the right until "3300" shows up on the right side of the display. "ALT ARM" should appear on the bottom.

Note that if you're using the autopilot to descend, it will just push the nose down, like a bad pilot, so the airplane will speed up. We want to go down, but we don't want to speed up, so we need to reduce engine RPMs to keep the speed at 110 knots. Later, when you level off at 3300 feet, you'll have to increase power again.

If you're flying manually, then you just need to adjust the engine to get the descent rate you want — the plane should stay magically at 110 knots if it's already trimmed for 110.

9.4.7 ILS Landings

While descending, we also need to start considering how we're going to intercept 255° on the way back and follow it down to the runway. You might think we're going to use the NDB like we did on the outbound leg, but at this point, the NDB is not good enough. This is an ILS landing, a so-called "precision" landing, and NDB is just not precise enough. It can get us close to the runway, but not close enough.

So, we're going to switch over to our ILS system. It is much more accurate horizontally. As well, it offers vertical guidance, something which the NDB does not give at all. And hey, it also gives you something else to learn in our few remaining minutes so that you don't get bored.

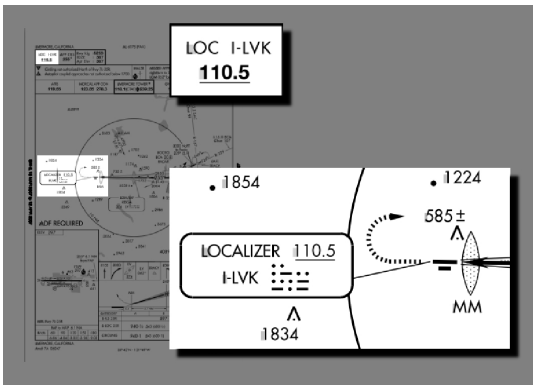


Figure 9.16: Livermore 25R localizer

As with NDB and VOR navigation, the ILS system[12] has a transmitter (or *transmitters* — a localizer *and* a glide slope) on the ground, and a receiver and a gauge in the aircraft. The receiver, it turns out, is just a NAV receiver, of which we have two. The gauge is like a VOR indicator, but it has an added glide slope indicator, which is a horizontal (you hope) needle. Like a VOR, the vertical needle shows whether you're left or right. The horizontal needle shows whether you're high or low. Our ILS gauge is our old friend VOR1.

As you might have guessed, the localizer has a frequency and ident associated with it (there's no need to tune the glide slope separately. If you tune the localizer, you've tuned the glide slope). This is shown on the approach plate in two places: at the top left corner, and in the plan view by the runway (see Figure 9.16). As we can see, the frequency is 110.5 MHz, and the ident is I-LVK (·· ·–·· ···– –·–).

Tune NAV1 to 110.5, and check for the ident. Sounds lovely, doesn't it? That localizer is going to save your bacon and get you out of this interminable soup. When you tuned into the localizer, you'll also have noticed the ILS needles move. And the OBS? Well, it's useless. Try moving it. No matter how you turn it, the needles don't move in response. That's by design. A localizer is basically a VOR with *one* radial, the approach heading. We don't care about any others, so we don't need an OBS to declare interest in any others. As a reminder, though, move the OBS to 255, our desired heading. This will remind us where to point our nose.

9.4.8 Intercepting the Localizer

We're now ready to intercept the ILS localizer. When the two minutes on the 030° leg have passed, make your U-turn to the right to 210°. Soon after you complete your turn, the vertical (localizer) needle on the ILS will begin to move. And it will move *fast*, much faster than the ADF and VOR needles did. A localizer is 4 times as sensitive as a VOR, relatively small movements of the aircraft make big

[12] See http://en.wikipedia.org/wiki/Instrument_Landing_System for more information.

changes in the needles. You'll probably overshoot, but don't worry, because we have around 5 or 10 minutes to get things straightened out.

Just remember: don't chase the needles. That mantra is now more important than ever. Those needles are sensitive — if you just turn left when the localizer needle is to the left and right when it's to the right, you'll be flying like a drunken sailor. If you're lucky, the runway will be passing underneath you as you swing across the track for the umpteenth time. Luck, though, is something we should not be relying on. Determine on how the needles are moving before making your move.

Now that you're heading back inbound at 255°, slow to 75 knots, drop a notch of flaps, and descend to 2800 feet (but no lower). And check for the inbound passage of FOOTO to confirm your position. And pat your head and rub your stomach.

9.4.9 Intercepting the Glide Slope

As we fly drunkenly towards the runway, cursing localizers and needles and resolving never, ever to fly in such crappy conditions ever again, don't forget to look at the horizontal needle, the glide-slope needle. At the start it was high above us, because we were actually under the glide slope. But as we levelled out at 2800, the glide slope started coming "down" to us. Eventually, you should see the needle start to move down. When the needle is horizontal, that means you're on the glide slope. You won't be for long, though, unless you start descending.

What's a good rate? It depends on our groundspeed. In our case, we're going at 75 knots (there's almost no wind, so our airspeed and groundspeed are the same), and it turns out that we need to descend at around 400 feet per minute. With the autopilot, that's pretty easy — just dial in -400, and you're set (but remember to reduce power to keep our speed at 75 knots, or you'll hit the runway going pretty fast, and be prepared to adjust things if you drift above or below the glide slope).

Without the autopilot, it's also pretty easy — just reduce power. How much? In this case, with our plane, to around 1600 RPM. Again, it depends on many things — plane, elevation, winds, weight, ..., so you'll have to adjust things if you see the glide-slope needle start to move up or down. Like the localizer needle though, ... (are you ready?) DON'T CHASE IT. Watch how it's moving, then make your adjustment.

Since we're on final approach, you might want to drop a second notch of flaps. This will affect your trim, and you'll have to adjust power a bit as well.

Soon after we intercept the glide slope, we should pass over the outer marker. Several things will happen more or less simultaneously, all of which confirm your position:

1. You'll hear a continuous series of dashes.

2. The blue light above COMM1 will flash.

3. The ADF needle will swing around.

9.4.10 Touchdown I

After all the excitement of the procedure turn, it will seem like a long way down to the runway from the outer marker. There's not much to do but stare at those needles. In fact, you'll probably stare at them like you've never stared at them before. Take a look around at the other gauges too, though — they have useful things to tell you. Is our airspeed okay? We don't want to stall. RPMs about right? If flying manually, you'll want to constantly check the attitude indicator and directional gyro. This being a simulator, we don't have to worry about oil pressure and engine temperature, but you might want to glance over there anyway, just to get into the habit. And I hope you've done things like set the mixture to full rich (you did lean it out while cruising, didn't you?).

9.4.11 Touchdown II

Although ILS approaches can get us close to the runway, closer than VFR, NDB, or VOR approaches can, we still need *some* visibility to land,[13] so we need a way to decide if landing is possible or not. That's what the landing minimums section of the procedure plate is for (coloured green in Figure 9.11). In the category labelled "S-ILS 25R" (that's us), you'll see "597-1/2 200(200-1/2)". This tells us that we can track the glide slope down to an altitude of 597 feet (200 feet above the runway). At 597 feet we make our decision — if we can't see the runway, then we have to execute a missed approach. 597 feet is our *decision height* (DH).

In addition to the altimeter, this particular approach also has another indication that we're close — a middle marker (MM). This marker will sound — in this case, a dot dash series — and the yellow light above COMM1 will flash. Passage over the middle marker should coincide with reaching decision height.

So, what if you can't see the runway at decision height? As you might have expected, just as you can't land willy-nilly, you can't just go around willy-nilly. There's a Procedure. A Missed Approach Procedure. This is shown in several places on the approach plate (see Figure 9.17): At the top, where it says "MISSED APPROACH", in the plan view, where you can see a dashed arrow coming off the end of the runway and a dashed oval on the right, and in the profile view, where a series of boxes shows graphically what to do. In our case, these all tell us to:

1. Climb straight ahead to 1100 feet
2. Make a climbing right turn to 3000 feet
3. Fly to REIGA
4. Fly outbound from REIGA at 062°

[13] Well, unless it's a Category III ILS approach.

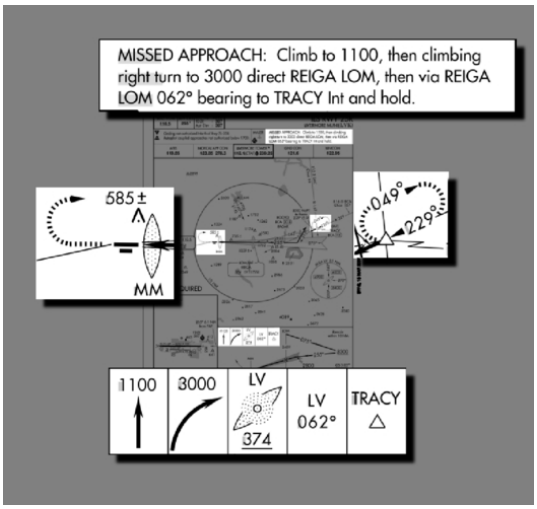


Figure 9.17: Missed approach procedure

5. Fly a holding pattern at the TRACY intersection

The holding pattern, as you might have guessed, is a place where you can "park" while sorting things out, and has its own set of procedures and techniques which we won't go into here, because ...

9.4.12 Touchdown III

In our ideal simulator world, you probably won't have to execute a missed approach. Assuming you stayed on the glide slope, you should have popped out of the murk at 750 feet, a whole 153 feet above the decision height, and with 1 mile visibility, the runway should have been in view soon after. With the runway in sight, you could then turn wildly to get on course[14] (it's very hard to be lined up perfectly) and land "normally" (which for me involves a lot of bouncing around and cursing). Park the plane, then stagger out of the cockpit and have another hamburger!

9.5 Epilogue

That was a lot of information in a short time, a rather brutal introduction to ILS flying. Hopefully, instead of turning you off, it has whetted your appetite for more, because there *is* more. Some of the major issues I've ignored are:

Wind This is a big one. Flying IFR in a crosswind affects everything you do, and you need to be aware of it or your navigation will suffer.

[14] Remembering, of course, to disengage the autopilot.

Figure 9.18: At decision height, runway in view. We're going to live!

Flying without the autopilot George tries his best, but he's not completely trustworthy. You have to be prepared to go it alone.

DG precession The directional gyro in the c172p is not perfect. Over time, the values it gives you are less and less reliable — it *precesses*. It needs to be periodically calibrated against the compass (using the OBS knob to adjust it).

IFR charts We used sectionals, which are really intended for VFR flight. There are a whole set of charts devoted exclusively to IFR flight.

ATC The other people out there need to know what you're doing.

If you want to learn more, try the following resources:

- *Flight Simulator Navigation*, written by Charles Wood. It covers everything from basic navigation to ILS approaches, with lots of examples and practice flights to improve your skills. Everything is linked together by an entertaining storyline in which you are the pilot for a fictional charter service.

  Two caveats, though. First, it is Microsoft Flight Simulator-based, so you'll have to translate into "FlightGear-ese" as appropriate. Second, it is a bit out of date, and things in the real world have changed since it was written. NDB beacons have been decommissioned, new approaches have replaced old ones — even an airport has disappeared (!). Treat this as a learning opportunity. You'll get better at finding more up to date information, and learn not to blindly trust your charts, just as you have learned not to blindly trust your instruments.

http://www.navfltsm.addr.com

- If you're *really* keen and want to hear it straight from the horse's mouth, there's the official *FAA Instrument Flying Handbook*. It's big and detailed, and there's *no* interesting storyline in which you're a pilot for a fictional charter service. It can be downloaded as two PDF files.

  http://www.faa.gov/library/manuals/aviation/instrument_flying_handbook

- If you'd like practice deciphering what the instruments are telling you, without the bother flying (or even virtual flying), you can try *luizmonteiro.com*, which has Flash tutorials of various instruments, including a VOR and an ADF.

  http://www.luizmonteiro.com/Learning.htm

- Another simulated instrument site is *Tim's Air Navigation Simulator*. It has a Java applet that simulates a plane flying in the vicinity of two VOR stations.

  http://www.visi.com/~mim/nav/

- If it's navigation information you're after, an excellent site is *AirNav.Com*, which I've used extensively in the course of this tutorial. It has detailed airport, navaid, and fix information, and links to IAPs. Unfortunately, the information is only for the USA.

  http://www.airnav.com

- Another source of airport and navaid information is *World Aero Data*. Its information isn't as detailed as AirNav's, but it is international.

  http://worldaerodata.com

Chapter 10

A Helicopter Tutorial

10.1 Preface

First: in principle everything that applies to real helicopters, applies to *FlightGear*. Fundamental maneuvers are well described here:
http://www.cybercom.net/~copters/pilot/maneuvers.html Some details are simplified in *FlightGear*, in particular the engine handling and some overstresses are not simulated or are without any consequence. In *FlightGear*it is (up to now) not possible to damage a helicopter in flight.

Since the release of version 0.9.10 some improvements have been made to the helicopter flight model and therefore the CVS version should be used. With these improvements the helicopter flight model of *FlightGear*should be quite realistic. The only exceptions are “vortex ring conditions”. These occur if you descend too fast and perpendicularly (without forward speed). The heli can get into its own rotor downwash causing the lift to be substantially reduced. Recovering from this condition is possible only at higher altitudes. On the Internet you can find a video of a Seaking helicopter, which got into this condition during a flight demonstration and touched down so hard afterwards that it was completely destroyed.

For all *FlightGear*helicopters the parameters are not completely optimized and thus the performance data between model and original can deviate slightly. On

the hardware side I recommend the use of a “good” joystick. A joystick without springs is recommended because it will not center by itself. You can either remove the spring from a normal joystick, or use a force feedback joystick, with a disconnected voltage supply. Further, the joystick should have a “thrust controller” (throttle). For controlling the tail rotor you should have pedals or at least a twistable joystick - using a keyboard is hard. Flightgear supports multiple joysticks attached at the same time.

10.2 Getting started

The number of available helicopters in *FlightGear*is limited. In my opinion the Bo105 is the easiest to fly, since it reacts substantially more directly than other helicopters. For flight behavior I can also recommend the as350, though there is no 3D-model - it uses the Bo105 3D-model. The As350 reacts more retarded than the Bo.

Once you have loaded *FlightGear*, take a moment to centralize the controls by moving them around. In particular the collective is often at maximum on startup.

The helicopter is controlled by four functions. The stick (joystick) controls two of them, the inclination of the rotor disc (and thus the inclination of the helicopter) to the right/left and forwards/back. Together these functions are called “cyclic blade control”. Next there is the “collective blade control”, which is controlled by the thrust controller. This causes a change of the thrust produced by the rotor. Since the powering of the main rotor transfers a torque to the fuselage, this must be compensated by the tail rotor. Since the torque is dependent on the collective and on the flight condition as well as the wind on the fuselage, the tail rotor is also controlled by the pilot using the pedals. If you push the right pedal, the helicopter turns to the right (!). The pedals are not a steering wheel. Using the pedals you can yaw helicopter around the vertical axis. The number of revolutions of the rotor is kept constant (if possible) by the aircraft.

10.3 Lift-Off

First reduce the collective to minimum. To increase the rotor thrust, you have to "pull" the collective. Therefore for minimum collective you have to push the control down (that is the full acceleration position (!) of the thrust controller). Equally, "full power" has the thrust controller at idle. Start the engine with **}**. After few seconds the rotor will start to turn and accelerates slowly. Keep the stick and the pedals approximately centered. Wait until the rotor has finished accelerating. For the Bo105 there is an instruments for engine and rotor speed on the left of the upper row.

Once rotor acceleration is complete, pull the collective very slowly. Keep your eye on the horizon. If the heli tilts or turns even slightly, stop increasing the collective and correct the position/movement with stick and pedals. If you are successful, continue pulling the collective (slowly!).

As the helicopter takes off, increase the collective a little bit more and try to keep the helicopter in a leveled position. The main challenge is reacting to the inadvertent rotating motion of the helicopter with the correct control inputs. Only three things can help you: practice, practice and practice. It is quite common for it to take hours of practice to achieve a halfway good looking hovering flight. Note: The stick position in a stable hover is not the center position of the joystick.

10.4 In the air

To avoid the continual frustration of trying to achieve level flight, you may want to try forward flight. After take off continue pulling the collective a short time and then lower the nose a slightly using the control stick. The helicopter will accelerate forward. With forward speed the tail rotor does not have to be controlled as precisely due to the relative wind coming from directly ahead. Altogether the flight behavior in forward flight is quite similar to that of an badly trimmed airplane. The "neutral" position of the stick will depend upon airspeed and collective.

Transitioning from forward flight to hovering is easiest if you reduce speed slowly by raising the nose of the helicopter. At the same time, reduce the collective

to stop the helicopter from climbing. As the helicopter slows, "translation lift" is reduced, and you will have to compensate by pulling the collective. When the speed is nearly zero, lower the nose to the position it was when hovering. Otherwise the helicopter will accelerate backwards!

10.5 Back to Earth I

To land the helicopter transition to a hover as described above while reducing the altitude using the collective. Briefly before hitting the ground reduce the rate of descent slowly. A perfect landing is achieved if you managed to zero the altitude, speed and descent rate at the same time (gently). However, such landing are extremely difficult. Most pilots perform a hover more or less near to the ground and then decent slowly to the ground. Landing with forward velocity is easier, however you must make sure you don't land with any lateral (sideways) component to avoid a rollover.

10.6 Back to Earth II

It is worth mentioning autoration briefly. This is a unpowered flight condition, where the flow of air through the rotors rotates the rotor itelf. At an appropriate altitude select a landing point (at first in the size of a larger airfield) and then switch the engine off by pressing `{`. Reduce collective to minimum, place the tail rotor to approximately 0° incidence (with the Bo push the right pedal about half, with As350 the left). Approach at approximately 80 knots. Don't allow the rotor speed to rise more than a few percent over 100%, otherwise the rotor will be damaged (though this is not currently simulated). As you reach the ground, reduce the airspeed by lifting the nose. The descent rate will drop at the same time, so you do not need to pull the collective. It may be the case that the rotor speed rises beyond the permitted range. Counteract this by raising the collective if required. Just above the ground, reduce the descent rate by pulling the collective. The goal is it to touch down with a very low descent rate and no forward speed. With forward speed it is easier, but there is a danger of a roll over if the skids are not aligned parallel to the flight direction. During the approach it is not necessary to adjust the tail rotor, since without power there is almost no torque. If you feel (after some practice),

that autorotation is too easy, try it with a more realistic payload via the `payload` menu.

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.ethe ***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-0.3
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/FlightGear-0
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 prebuilt 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 ongoing 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 mainboard 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 OpenGLṪhis 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 running 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. eàn audio library and some basic background engine sound. This was later integrated into the above-mentioned 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.

Stuart Buchanan
Updated various parts of the manual and wrote the initial tutorial subsystem.

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.

Maik Justus
Improved the YASim helicopter FDM, and added glider towing.

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.

Sam van der Mac
Contributed to The Manual by translating HTML tutorials to Latex.

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

.fgfsrc, 29, 215
2D cockpit, 54
3D cockpit, 54
3D panels, 214
3DFX, 189, 205
3dfx, 206

A1 Free Sounds, 216
A4, 16
abort, 117
additional scenery, 19
ADF, 62
Adler, Mark, 218
Aeronautical Information Manual, 84
AGP, 207
AGP Support, 206
AI, 59
aileron, 53, 61
aileron indicator, 64
Air Traffic Control, 146
air traffic facilities, 62
Air-Air Refuelling, 77
aircraft
    installation, 21
    selection, 32
    survey, 45
Aircraft Carrier, 67
aircraft model, 33
airport, 33, 224
airport code, 33, 66
airspeed indicator, 61
Airwave Xtreme 150, 17
Alonzo, Raul, 216
Altimeter, 141
altimeter, 61, 96
    tuning, 96
altitude
    absolute, 96
altitude hold, 54
America, Michele, 213, 216
anonymous cvs, 16
anti-aliaseded HUD lines, 32
artificial horizon, 61
astronomy code, 214
ATC, 59, 214, 224
ATI, 205, 206
ATIS, 62, 138
ATIS messages, 214
Atlas, 69, 215
attitude indicator, 61
audio library, 219
audio support, 215
auto coordination, 31, 61
autopilot, 54, 58, 63, 122, 142, 213, 218, 219
    modes
        heading mode, 122
        roll control mode, 122
        vertical speed mode, 122
autopilot controls, 54, 55
autothrottle, 54

Baker, Steve, 214
bank, 61
base package
    installation, 201
Basler, Michael, 216
Beech 99, 17
Bendix transponder, 214
Berndt, Jon, 225
Berndt, Jon, S., 213, 216, 224
binaries, 191
    directory, 197
    pre-compiled, 7
binaries, pre-compiled, 191
binary directory, 194
binary distribution, 5
bleeding edge snapshots, 201
Bleisch, Paul, 217
Boeing 747, 16
brakes, 56, 62, 98, 221
    left wheel [,] (colon), 98
    right wheel [.] (dot), 98
branch, developmental, 15
branch, stable, 15
Brennan, Jim, 217
Bright, Bernie, 217
BSD UNIX, 12
Buchanan, Stuart, 217

Buckel, Bernhard, 217, 224
Buckle, Gene, 217

callsign, 36
Carmichael, Ralph, 217
Cessna, 64, 222
Cessna 172, 16, 213
Cessna 182, 16
Cessna 310, 16
Chauveau, Didier, 217
Check, John, 45, 63, 214, 215, 217, 225
Cherokee flight model, 222
clock, 62
cloud layer, 34
clouds, 214, 222
cockpit, 54
CodeWarrior, 200
COM transceiver, 62
COMM1, 62
COMM2, 62
command line options, 29
communication radio, 62
compiler, 15
compiling, 191
    IRIX, 200
    Linux, 195
    Macintosh, 198
    other systems, 200
    Solaris, 200
    Windows, 195
configure, 197
contributors, 216
control device, 31
control surface, movable, 214
Cornish, Dave, 212, 217
cvs, anonymous, 16
Cygnus, 15, 224
    development tools, 192
Cygwin, 15, 190
    packages to install, 193
    setup, 192
    XFree86, 193

DC-3, 16
Debian, 192
debug, 59
default settings, 30
Delise, Oliver, 215, 216, 218, 225
Denker, John, 85
Detonator reference drivers, 204
development environment, 192, 194
differential braking, 56
Direct3D, 14
directory structure, 201
disk space, 15, 192
display options, 54
distribution
    binary, 191
documentation, 13
    installation, 21
DOS, 211
Doue, Jean-Francois, 218
DRI, 209

Eberly, Dave, 218
elevation indicator, 64
elevator trim, 53
engine, 51
    shut down, 106
    starting, 51
engine controls, 55
environmental variables, 25
equipment, 59
Evans, Francine, 218
Everitt, Oscar, 218
exit, 57, 66

FAA, 84
FAA Training Book, 84
FAQ, 6, 7, 187
FDM, 213, 216
    external, 17
    pipe, 17
FG_ROOT, 25
FG_SCENERY, 25
field of view, 34
Finney, Bruce, 218
flaps, 56, 61, 63, 109
    control lever, 109
    steps, 109
flight dynamics model, 16, 32, 213
flight instrument, 61
flight model, 16, 32, 213
flight planner, 215
Flight simulator
    civilian, 12
    free, 211
    multi-platform, 12
    open, 12, 13
    user-extensible, 12, 13
    user-sported, 12
    user-supported, 13
FlightGear, 215
    directory structure, 201
    versions, 15
FlightGear documentation, 17
FlightGear Flight School, 18
FlightGear Programmer's Guide, 17

FlightGear Scenery Design Guide, 18
FlightGear Website, 17, 216
flightmodels, 16
Foad, Julian, 225
fog, 34
fog effects, 219
frame rate, 15, 34, 212
Franz, Melchior, 218, 225
FreeBSD, 218
FreeGLUT, 200
frozen state, 31
FS98, 218
fuel indicator, 62
full screen display, 30
full screen mode, 34, 54

Gailly, Jean-loup, 218
GameGLUT, 222
Garg, Mohit, 218, 225
gauge, 61
gear, 56
Geforce, 7
Gellekum, Thomas, 218
geographic features, 212
Girish, Neetha, 218
GLIDE, 189
GNU C++, 15
GNU General Public License, 13
Go Around, 150
Goeke-Smith, Jeff, 213, 218
Gold, Michael, I., 218
GPL, 13
graphics card, 14
graphics library, 203
graphics routines, 211
GSHHS data, 213
gyro compass, 61

Habibe, 219
hang glider, 17
hangar, 45
Harrier, 16
haze, 34, 35
head up display, 64, 213
heading hold, 54
height, 64
help, 60
Hill, Mike, 219
History, 211
history
    aircraft, 213
    environment, 214
    scenery, 212
    user interface, 214
Hofman, Eric, 200
Hofman, Erik, 212, 219, 225
hot air balloon, 220
Hotchkiss, Charlie, 213, 219
HTTP server, 215
http server, 36
HUD, 32, 35, 64, 65, 213, 216, 219
HUD (Head Up Display
    full mode), 126
HUD (Head Up Display), 126, 128

icing
    modelling, 16
IFR, 63, 85
ignition switch, 51, 62
inclinometer, 61
initial heading, 33
install directory, 195
installing aircraft, 21
instrument flight rules, 63
instrument panel, 32, 54, 60, 213
Internet, 215
IRIX, 200

Jackson, Bruce, 211, 219
joystick, 31, 38, 51, 52, 215
    .fgfsrc, 44
joystick settings, 215
joystick/self detection, 215
joysticks, 15
JSBSim, 32
Justus, Maik, 219

Kaaven, Ove, 219
Kaszeta, Richard, 219
keybindings
    configuration, 56
keyboard, 51, 91
    numeric, 91
    uppercase and lowercase keys, 91
keyboard controls, 51–53
    miscellaneous, 56
keyboard.xml, 56
Knienieder, Tom, 219
Koradi, Reto, 219
Korpela, Eric, 211
Kuehne, Bob, 219

Laird, Kyler B., 219, 225
landing, 114
    aid signals, 96
landing gear, 56
LaRCsim, 211–213, 219, 220, 222
latitude, 65

Launching Flightgear
    Linux, 26
    Mac OS X, 29
    Windows, 27
leaflet, 7
light textures, 212
Linux, 7, 12, 13, 15, 191, 204, 205, 211
Linux distributions, 192
Livermore, 136
load flight, 56
location, 58
longitude, 65
Luff, David, 214, 220

Mac OS 9.x, 209
Mac OSX, 209
Macintosh, 7, 200
magnetic compass, 61
magneto, 103
magneto switch, 214
mailing lists, 187, 216
map, clickable, 212, 213
Marchetti S-211, 17
marker, inner, 63
marker, middle, 63
marker, outer, 63
Mayer, Christian, 214, 216, 220
Megginson, David, 84, 213–215, 220, 225
menu, 214
menu entries, 56
menu system, 224
MetaKit, 223
Metro Works, 223
Microsoft, 11
Mitchell, Eric, 212, 220
mixture, 63, 105, 145
    lever, 105
    optimisation, 106
mixture lever, 51
Moore Cameron, 187
Moore, Cameron, 220, 225
Morken, Anders, 220
mouse, 51, 65
    normal mode, 93
    rudder control, 99
    view mode, 93
    yoke mode, 93
mouse modes, 65
mouse pointer, 30
mouse, actions, 65
MS DevStudio, 190
MSVC, 190, 217
multi-engine support, 214
multi-lingual conversion tools, 220
Multiplayer, 70
multiplayer code, 215
Multiple Displays, 73
Murr, David, 211
Murta, Alan, 220

NAV, 62
navaids, 63
Navion, 213
NDB, 62
Nelson, Phil, 220
network, 215
network options, 36
networking code, 215, 218
networking support, 197
Novikov, Alexei, 220
NumLock, 52
NVIDIA, 7, 204–206
    drivers, 203
    Linux drivers, 204
    Windows drivers, 204

offset, 35
Olson, Curt, 20, 211, 212, 214–216, 220, 224
OpenAL, 196
OpenGL, 6, 7, 14, 15, 17, 188, 203–205, 209, 212, 215, 218
    drivers, 15
    libraries, 200
    Linux, 205
    Macintosh, 209
    runtime libraries, 208
OpenGL drivers, 203
OpenGL renderer string, 208
OpenGL Setup, 205
OpenSceneGraph, 196
Operating Systems, 12
options
    aircraft, 32
    debugging, 38
    features, 32
    flight model, 32
    general, 29
    HUD, 35
    initial position, 33
    IO, 37
    joystick, 38
    network, 36
    orientation, 33
    rendering, 34
    route, 37
    time, 35
    waypoint, 37
options, configure, 197

OS/2, 211

panel, 60, 220, 221
  reconfiguration, 63
parking brake, 52, 56
Paul, Brian, 221
pause, 56
PCI, 208
pedal, 38
Peden, Tony, 216, 221
Peel, Robin, 214, 221
permissions, 189
Perry, Alex, 221, 225
Perry, Dave, 225
PFE, 44
pitch, 61
pitch indicator, 64
places to discover, 66
Playback, 74
PLIB, 196, 214, 215
  header files, 196
preferences, 29
problem report, 187
problems, 187
  general, 188
  Linux, 189
  Windows, 189
programmers, 216
property manager, 215
proposal, 211

Quake, 203

radio, 139
radio stack, 62, 214
random ground objects, 213
README.xmlpanel, 63
Reid-Hillview, 136
Reinhard, Friedemann, 213, 221
Reinholdtsen, Petter, 221
reset flight, 57
Riley, William, 20, 212, 221
Ross, Andy, 214, 221, 225
RPM indicator, 61
rudder, 52, 53, 61
  keyboard control, 99
  mouse control, 99
rudder indicator, 64
rudder pedals, 15, 51
runway lighting, 212

save flight, 56
scenery, 212
  additional, 19
scenery directory
    path, 30
scenery subsystem, 220
Schlyter, Paul, 221
Schoenemann, Chris, 221
Schubert, Phil, 221
screenshot, 56, 57
Sectional, 136
See how it flies, 85
Selig, Michael, 214, 225
SGI IRIX, 12
SGI Irix, 7
Shewchuk, Jonathan, 222
shutdown, 117
side-slip, 110
Sikic, Gordan, 222
SimGear, 194, 196, 215
Smith, Michael, 222
snapshots, 201
Solaris, 200
sound card, 15
sound effects, 15
source code, 13
speed, 64
  units
    knot [nautical mile per hour], 100
spin, 110
Spott, Martin, 222, 224
SRTM, 19
stall, 110
starter, 51, 62
Starting Flightgear
  Linux, 26
  Mac OS X, 29
  Windows, 27
starting the engine, 62
startup latitude, 33
startup longitude, 33
startup pitch angle, 33
startup roll angle, 33
static objects, 213
Sun-OS, 12, 211
SuSE, 192, 205, 209
system requirements, 14
system.fgfsrc, 29, 215

tachometer, 98
tail-wheel lock, 56
Talsma, Durk, 214, 222
telnet server, 36
TerraGear, 215
terrain, 35
Text To Speech, 75
texture, 212

textures, 212, 220
throttle, 51, 53, 63, 64
throttle lever, 105
thunderstorms, 214
time offset, 56
time options, 35
TNT, 7
Torvalds, Linus, 13
Traffic Pattern, 147
triangle program, 222
triangles, 35
trim, 53, 111
Troester, Kai, 189, 225
troubles
    mouse speed, 94
    stuttered simulation, 89
Turbo 310, 16
turn coordinator, 97
turn indicator, 61, 64
tutorial, 84

U. S. Geological Survey, 212, 222
UIUC, 214, 222
UIUC airplanes
    3D models, 17
UIUC flight model, 16, 32
UNIX, 14, 191, 200, 211

Vallevand, Mark, 222
van der Mac, Sam, 220
van Sickle, Gary, R., 216, 222
VASI, 149
velocity rages, 61
vertical speed indicator, 61
VFR, 63, 85
video card, 203
view, 57
    changing, 91
    instant replay, 92
view directions, 53
view frustrum culling, 212
view modes, 54
views, 215
Vine, Norman, 213, 215, 216, 223, 224
visibility, 54
Visual C++, 200
visual flight rules, 63
VMap0, 19
VMap0 data, 213
Voegtli, Roland, 223
Volpe, Carmelo, 223
VOR, 62

Walisser, Darrell, 223, 225
waypoint, 58
weather, 58, 197, 220
wiki, 7
Williams, Ed, 223
Wilson, Jim, 223
window size, 35
Windows, 7, 15, 191, 204
Windows 95/98/ME, 12
Windows 95/NT, 211
Windows NT/2000/XP, 12
winds, 214
windsock, 120
Wippler, Jean-Claude, 223
wireframe, 35
Wood, Charles, 85
Woodsoup, 223
workstation, 14, 211
Wright Flyer, 17

X server, 207
X15, 16
XFree86, 189, 206, 209

YASim, 16
yoke, 31, 38, 51, 52, 62, 92
    mouse yoke mode, 93, 95
    pulling, 94
yokes, 15

Zeh, Allan, 224
ZLIB
    installation, 195
zlib library, 218