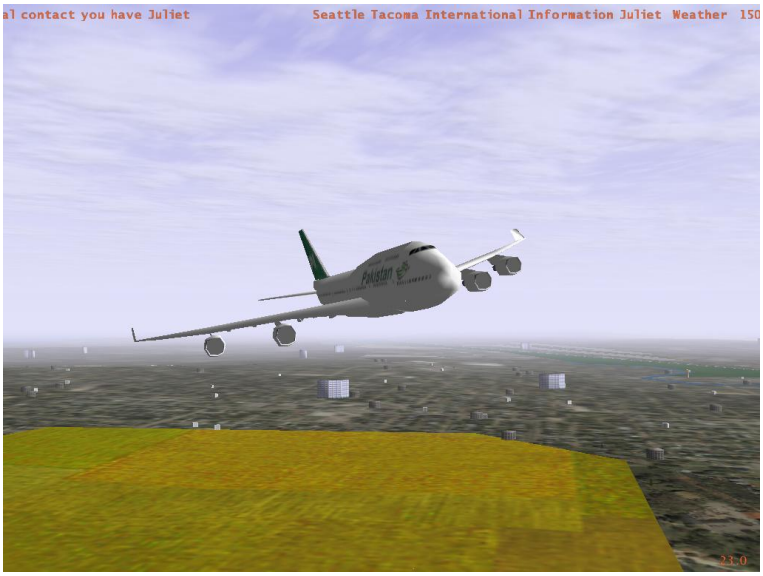
FlightGear 手册

Michael Basler, Martin Spott,
Stuart Buchanan, Jon Berndt,
Bernhard Buckel, Cameron Moore,
Curt Olson, Dave Perry,
Michael Selig, Darrell Walisser,
佟辉（中文翻译）



FlightGear 手册
2016 年 8 月 15 日
适用于 *FlightGear* 2016.3 “里约热内卢” 版

目录

第一章
序言

FlightGear 是一个自由的飞行模拟器，由互联网上一群对飞行模拟和编程狂热爱好的人开发。这本《FlightGear 手册》希望介绍给初学者如何快速上手 *FlightGear*，并享受飞行的乐趣。这里不会提供介绍 *FlightGear* 全部特性和插件的文档，然而会让新手了解如何开始和探索 *FlightGear* 所提供的各种体验。

此版本的文档是为 *FlightGear* 2016.3 “里约热内卢” 所写。使用早期版本 *FlightGear* 的用户也能在此文档中找到有用的信息，然而很多文档中提到的新特性，可能无法使用。

此文档分成三个部分，各部分的内容和结构罗列如下。

第一部分：安装

第 二 章，想要自由的飞翔？来玩 *FlightGear* 吧，介绍 *FlightGear*，并讲解其中的背景知识和哲学，还有系统要求。

第 三 章，飞行前的准备：安装 *FlightGear*，这里将提供如何安装二进制发行版和附加地景及飞行器的介绍。

第二部分：在 *FlightGear* 中飞行

第 四 章，起飞：如何启动程序，描述了如何实际启动已经安装的程序。还包括概述一些命令行参数和配置文件。

第 五 章，飞行中：仪表板、键位和菜单，描述了如何操作这个程序，比如如何在 *FlightGear* 中实际飞行。这里包括了一份完整的（希望是）预定义键盘命令列表，一个菜单项概览，还详细讲述了仪表板和 HUD（Head Up Display，平视显示器）的操作，当然还有使用鼠标的一些提示。

第 六 章，特性，描述了一些 *FlightGear* 中特别为高级用户提供的特性。

第三部分：教程

第 七 章，教程，提供了为新飞行员准备的一些教程。

第 八 章，基础飞行模拟教程，提供了一份很基础的飞行教程，以插图的形式举例讲解在 *FlightGear* 中如何飞行。

第 九 章，短途转场飞行教程，从默认安装的旧金山地区出发，做一次短途旅行。

第 十 章，*IFR* 转场飞行教程，讲述如何使用仪表飞行规则（IFR），完成同样的短途飞行。

附录

附录 A，复飞：如果有什么拒绝工作，我们尝试帮助你解决一些使用 *FlightGear* 时的常见问题。

最后的附录 B，降落：离开飞机之前的一些想法，我们想给那些希望贡献到 *FlightGear* 开发中的人一些信息。告诉大家还有哪些需要完成。

1.1 简明阅读

为那些不喜欢从头至尾阅读文档的人准备的，我们建议以下章节，可以让你更简单的飞上蓝天。

安装:	第 三 章
启动模拟器	第 四 章
使用模拟器	第 五 章

1.2 为特别不耐烦的人

我们知道大多数人都讨厌阅读手册。如果你确定你的显卡支持 OpenGL（请阅读相关文档，比如大多数 NVIDIA 显卡）并且使用 Windows、Mac OS-X 或 Linux，那么你至少可以直接跳过这本手册的第一部分，并使用已经编译好的二进制程序。相关下载和说明可以前往

http://www.flightgear.org/download/.

如果你使用 Linux，会发现 *FlightGear* 已经在你的发行版包列表里了。当你安装好二进制程序以后，可以看第 四 章来获取启动模拟器的细节。

1.3 延伸阅读

虽然这种入门指南，就是要自成一体，不过我们依然强烈建议你阅读一下这些文档，特别是遇到问题的时候:

- 一张手边备查的**快速参考**可以在基本包的 `/FlightGear/Docs` 目录下找到并打印下来，也可以从这里下载

  http://www.flightgear.org/Docs/FGShortRef.pdf.

- 额外有关特定特性和功能的**用户文档**可以在基本包的 `/Docs` 目录找到。

- 还有一个官方的 *FlightGear* **wiki**，可以前往 http://wiki.flightgear.org。

第一部分

安装

第二章
想自由的飞翔？玩 *FlightGear* 吧！

2.1 另一个飞行模拟器?

你是否想过自己开飞机？但囿于金钱或者能力而望天兴叹？或许你是个真实飞行员希望在地面提高自己的技术？或许你想挑战一些危险的动作，而不用担心生命安全？亦或你只是想体验一个没有暴力的严肃游戏？如果你符合以上这些诉求，那么 PC 上的飞行模拟器对你就再适合不过了。

你也许已经用过 Microsoft© Flight Simulator（微软飞行模拟）或者其他商业级 PC 飞行模拟器。这些模拟器大多 50 美元左右，买一个并不是什么大问题，然而为了运行这些飞行模拟器则需要电脑超强的硬件性能，即便现在硬件价格持续下跌，购置这些仍大概需要 1500 美元左右。

既然已经有了这么多商业级的飞行模拟器，为什么我们还要花费数千小时编程和设计，开发一款自由的飞行模拟器呢？好吧，这里有很多原因，大体来说：

- 所有商业飞行模拟器都有一个通病：由一群开发者根据自身的私有需要来定义什么对他们是重要的，并限制终端用户的接口。尝试联系一个商业开发者让他听你的声音，在这种情况下是一个巨大的挑战。然而 *FlightGear* 是为所有人设计的，所有的一切都是开放的。

- 商业模拟器通常要在实用性和特性之间做出权衡。大多数商业开发者希望服务于更广泛的用户群，包括严肃的飞行员、初学者，甚至是休闲玩家。而事实上，最终都会向期限和资金妥协。由于 *FlightGear* 是自由且开放，也就没有了这些妥协。不会有发行商扼住我们的脖子，我们都是志愿者，大家一起制定自己的发行期限。我们也支持那些商业开发者没兴趣的领域，比如科研院所。

- 因为它们天然闭源，商业模拟器开发者将自己的天才想法和技能封闭的贡献给了他们的产品。在 *FlightGear* 中，开发者的所有技术水平和想法对这个项目有巨大的潜在影响。贡献到一个如 *FlightGear* 这样复

杂的项目中是有非常丰厚报偿的，开发者将会因此获得巨大知名度，因为我们正打造一款伟大的模拟器。

- 除此之外，纯粹就是好玩！你也许拿我们和那些制作飞行套件或飞机图纸的真飞行员比较，当然我们可以去外面买一台预先造好的飞行器，然而自己做却有特殊的体验。

以上所列都是我们为什么要创建 *FlightGear*。正是因为有了这些想法，我们想创建一个高质量的飞行模拟器。打造一个针对民用航空，跨平台的，开放的，用户支持的，以及用户可扩展的，一个飞行模拟平台。

- **民用航空：**这个项目主要针对民用航空模拟。它适用于通用航空和民用客机。我们的长远打算是希望 *FlightGear* 能成为 FAA 许可的飞行训练设备。让某些用户比较失望的是，*FlightGear* 目前不是一个空战模拟器；不过相应的特性并非明确排除在外。我们只是没有开发者对空战模拟的系统性需求有严肃的兴趣。

- **跨平台：**开发者尝试让代码尽可能做到平台无关性。根据我们的观察，对飞行模拟感兴趣的人运行了各种计算机硬件和操作系统，本项目代码支持如下操作系统：

  - Linux（任何发行版和平台），
  - Windows NT/2000/XP/7（Intel/AMD 平台），
  - Windows 95/98/ME
  - BSD UNIX
  - Sun-OS
  - Mac OS X

  目前为止，还没有任何一个飞行模拟器——无论是商业的或者自由的——能够支持如此广泛的平台。

- **开放：**此项目不受限于任何一个具体或精英的开发者。欢迎任何觉得有能力的人贡献其中。源代码（包括文档）都是以 GNU 公共许可协议（GNU General Public License）也就是 GPL 授权的。

  GPL 也经常被误解。简单来说 GPL 授权你可以自由的拷贝和再发布程序。你可以修改或者用修改以后和原始的程序谋利，然而，当发布软件的时候，必须保证再发布的软件源代码与原始的使用相同授权许可。一言以蔽之：

“你可以用这个软件做任何事情，只要不把它变成非自由的就可以”。

完整版 GPL 协议文本可以从 *FlightGear* 源代码或从以下渠道获得 http://www.gnu.org/copyleft/gpl.html.

- **用户支持及用户可扩展性：** 不像大多数商业模拟器，*FlightGear* 中的地景和飞行器的格式，内部变量，API，以及其他所有都是用户可以访问的，并配有从基础开始的文档。即使没有任何明确的开发文档（自然以后肯定会写），任何人都可以参考源代码来学习其背后原理。开发者的目标是建构一个基础的引擎，它可以让地景设计者、仪表板工程师、专家或者 ATC 系统设计者，声音艺术师、以及其他人共建其中。这正是我们的希望，让这个项目的开发者和终端用户，都能从全世界数百“模拟器达人”的创造性和想法中获益。

毋庸置疑，Linux 项目的成功，肇始于 Linus Torvalds 并影响了无数的开发者。Linux 不仅证明了通过互联网分布式开发尖端软件是可能的，同时也证明这样的成果可以超越同等质量的商业产品。

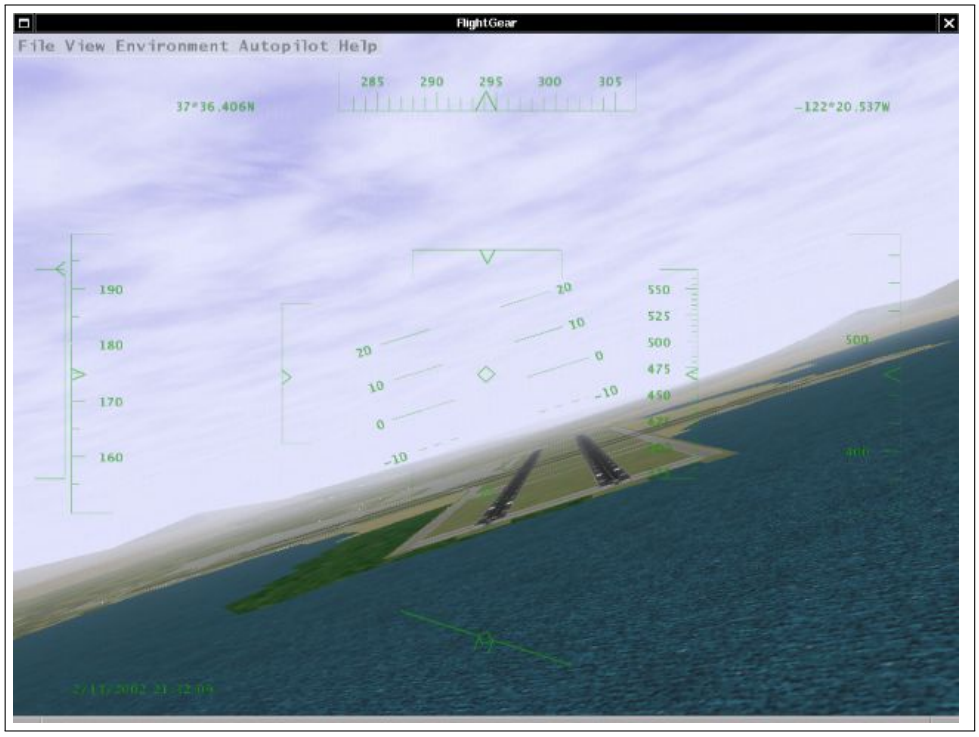


图 1：UNIX 系统下的 *FlightGear*：旧金山国际机场的糟糕进近——来自本手册一位作者的截图...

2.2 系统需求

与其他飞行模拟器相比，*FlightGear* 的系统需求并不奢侈。一个中等速度的 AMD Athlon64 或 Intel P4 就可以，甚至是低端的 AMD Athlon/K7 或 Intel PIII 也都可以让 *FlightGear* 做的非常好，前提是你需要有一款适合的 3D 显卡。

运行 *FlightGear* 有一个很重要的条件，就是需要一款支持 OpenGL 的显卡。如果你不知道什么是 OpenGL 可以参考 OpenGL 网站的简介

http://www.opengl.org

简单说："OpenGL 起步于 1992 年，现在已经成为支持 2D 和 3D 的工业级应用程序接口..."

FlightGear 不会（且永远不会）运行于只支持 Direct3D/DirectX 的显卡。相比于 OpenGL，Direct3D 是私有接口，并限制在 Windows 操作系统。

你可能在一个 3D 显卡不支持 OpenGL 硬件加速的电脑上运行 *FlightGear* ——甚至系统都没有 3D 显卡。然而没有硬件加速的电脑，即便是最快速的也会崩溃。没有硬件加速时的典型情况是帧速率会低于每秒 1 帧。

任何现代的 3D 显卡都会支持 OpenGL。如果 Windows 显卡驱动不能支持 OpenGL，请访问你的显卡厂家的网站。你需要注意的是有时候 OpenGL 的驱动是由主板的厂商提供的，而不是显卡厂商。如果你想购买一款运行 *FlightGear* 的显卡，推荐 NVIDIA GeForce 显卡，因为比 AMD/ATI 显卡对 OpenGL 有更好的支持。256MB 的独立显存就已经绰绰有余了——即便较少的显存也能让很多人玩 *FlightGear* 很开心呢。

要安装和执行基本的地景，你需要大概 500MB 硬盘空间。如果你想自己编译程序，你还需要另外的 500MB 来放源代码和编译过程产生的临时文件。这还不包括开发环境，这取决于操作系统以及环境使用。Windows 用户大概需要 300MB 的额外空间来做开发环境。Linux 和其他 UNIX 机器所需的开发环境大多都已经安装好了，因此这些平台仅需很少的额外空间。

关于声音效果，任何适合的声卡都足够了。因为设计灵活，在 Linux 和 Windows 平台下，*FlightGear* 可以支持大部分的飞行摇杆和驾驶盘，以及脚踏板。*FlightGear* 甚至还可以支持全动式的飞行座椅。

FlightGear 主要是在 Linux 下开发的，Linux 是一个自由的 UNIX 克隆（包括了大量的 GNU 组件），与 *FlightGear* 有着一样的精神，也是由一群互联网上广泛的合作开发而成。*FlightGear* 也可以在 Windows 下运行，一部分开发也是在 Windows 平台下完成。在 Mac OS X 上构建 *FlightGear* 也是可行的，与 UNIX/X11 工作站有一些区别。如果你已经安装了合适的编译器，*FlightGear* 都可以在以上这些平台完成构建。主要的全平台支持的编译器是自由的 GNU C++ 编译器（Win32 系统下则是 Cygnus Cygwin 编译器）。

如果你想在 Mac OS X 系统下运行 *FlightGear*，你需要至少 Mac OS X 10.4。最低硬件需求是 Power PC G4 1.0GHz 或一台 Intel Mac。然而若想飞行的更舒服我们推荐 MacBook Pro，Intel iMac，Mac Pro 或 Power Mac （Power PC G5）。

2.3 选择版本

建议你运行官方发布的最新版本，每年都会发布，且可以通过以下方式获取已经编译好的二进制文件

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

如果你想获得最新的、最完整的（当然同时也是 bug 最多的）代码，可以从以下地址获取

http://www.gitorious.org/fg. [1]
想获取这些的代码，下面这个链接介绍如何获取 *FlightGear* 的代码仓库

http://wiki.flightgear.org/GIT.

2.4 飞行动态模型

历史上，*FlightGear* 是遗传自 LaRCsim 并加入了 Navion 飞机。这带来了很多限制（最重要的是很多特性是硬编码实现的，而不是配置文件），多次尝试开发或者加入额外的飞行模型。现在导致的结果就是，*FlightGear* 可以支持多种不同的飞行模型，可以在运行时选择。

- 或许最重要的飞行模型是 JSB 飞行模型，由 Jon Berndt 开发。JSB 飞行模型是独立的 *JSBSim* 的一部分：

  http://jsbsim.sourceforge.net/.

- Andrew Ross 创建了另一个飞行模型 *Yet Another Simulator*（*YASim*）。*YASim* 使用和其他飞行动态模型（FDM）完全不同的方式，基于几何信息而不是基于空气动力学信息。*YASim* 还有特定的高级直升机 FDM。

- Christian Mayer 开发了一款热气球的飞行模型。后来 Curt Olson 整合了“UFO”瞬移模式，这样就可以让你快速从 A 点移动到 B 点。

- 最终，有了 UIUC 飞行模型，由伊利诺伊大学厄巴纳-香槟分校的团队研发。这个项目最开始是面向飞行器结冰建模，而现在则包括“非线性”空气动力学，也就是让飞机在极端姿态下变得更真实，比如失速和大迎角飞行。演示这些能力的两个好例子是 Airwave Xtreme 150 滑翔机 和 1903 年莱特兄弟的“飞行者”。更多有关 UIUC 飞行模型的细节可以参考

  http://www.ae.illinois.edu/m-selig/apasim/Aircraft-uiuc.html

[1] 译者注：因为 Gitorious 网站已经关闭，从 2015 年 3 月起代码已经转移到 https://sourceforge.net/projects/flightgear/

甚至可以在 FlightGear 的场景里使用运行在不同的计算机或通过本地计算机的具名管道运行的外部 FDM ——尽管对刚刚接触 *FlightGear* 的人来说也许这是非建议安装的。

2.5 关于本手册

还有一点，仅此一点，本手册中出现的材料仅仅在此出现。正如蒙田所说我们“只是集中别人手里的大把花束，没有装饰我自己，只是用带子将它们捆在了一起”。大多数（很不幸不是全部）出现在这里的信息也可以从 *FlightGear* 网站获取

http://www.flightgear.org/

《*FlightGear* 手册》是完善全部 *FlightGear* 文档的第一步。目标读者和终端用户是那些没有兴趣了解 OpenGL 内部机制或构建地景的人。我们希望有朝一日会有随同的《*FlightGear* 编程指南》，用来描述 *TerraGear* 软件包地景工具的《*FlightGear* 地景设计指南》；还有一个《*FlightGear* 飞行学校》的打包。

我们诚挚的请求你来帮助我们完善此文档，可以提交更正、提高、建议或翻译。所有用户都欢迎贡献替代的安装（显卡、操作系统等等）。我们将会非常高兴的将这些特性加入到之后版本的《*FlightGear* 手册》中（当然我们也会对贡献给予评价）。

第三章
飞行前：安装 *FlightGear*

若要运行 *FlightGear* 你需要安装二进制程序。之后如果你愿意，还可以安装额外的地景和飞行器。

最新发布的预先编译的二进制程序，支持如下平台

- Windows - 任何版本
- Mac OS X
- Linux

要下载请前往

http://www.flightgear.org/download/main-program/

根据网页上的指示操作。

3.1 安装地景

FlightGear 的详细地景可以覆盖整个世界，从世界之巅的喜马拉雅山脉到堪萨斯乡村，都可以任君自由飞行。*FlightGear* 的基本包包括了旧金山周边的一小片区域，所以要想飞到其他地方，就需要下载额外地景。

每一块地景都被打包成了一个压缩文件，或者一个 tarball，每经纬度 10 度为一块。每一个 tarball 以 10 x 10 经纬度块来命名，比如 w130n50.tgz。

你可以从下面这个可以点选的地图里下载:

http://www.flightgear.org/download/scenery/

另外，你可以通过购买覆盖全世界的完整地景集来支持 *FlightGear*:

http://shopping.flightgear.org/

你下载了 tarball 到你的电脑上以后，首先需要找到 *FlightGear* 的 `Scenery` 安装目录。

- 对 Windows 而言，可能在这样的文件夹下：

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

- 对各种 UNIX 系统，通常是：

  `/usr/local/share/FlightGear/data/Scenery`.

- 对 Mac OS X，可能是这种：

  `/Applications/FlightGear.app/Contents/Resources/data/Scenery`.

要安装地景，首先解压缩 tarball 到 `Scenery` 目录。大多数操作系统提供了解压缩 tarball 的工具。如果你不能解压缩 tarball，可以安装相应的程序比如 7-zip（http://www.7-zip.org/）。

注意不要解压缩 tarball 包里像 958402.gz 这样以数字命名的地景文件——这会由 *FlightGear* 在飞行时自动解压。

解压了 tarball 以后，`Terrain` 和 `Objects` 目录会包含额外的子目录，里面有新加的地景。

要使用新的地景，只需要选择地景所在区域的机场。如果你在使用 *FlightGear* 启动器（FlightGear Launcher），选择机场前请先按刷新按钮。

3.1.1 MS Windows Vista/7

如果你使用 Windows Vista 或 Windows 7，会发现 Windows 把安装下载的地景（和飞行器）放到你的 Virtual Store 目录下：

`c:\Users\(你的名字)\AppData\Local\VirtualStore\Program Files\FlightGear\Scenery`

如果是这样的话，你需要把 `Terrain` 和 `Objects` 目录手动拷贝到如上面所说的真实 *FlightGear* `Scenery` 目录。

3.1.2 Mac OS X

你可以使用图形化（GUI）启动器安装下载的地景数据和飞行器。在 *Advanced Features » Others* 选项卡上找到 *Install Add-On data* 按钮，将会打开一个文件浏览窗口。选择一个或多个地景数据文件，将会安装这些地景到`/Applications/FlightGear.app/Contents/Resources/data/Scenery`。可以接受的地景文件类型可以是 zip、tar.gz、tgz、tar 和已经解压缩的文件夹。如果使用图形化启动器安装时因为某些原因失败的话，你可以用替代方法安装数据。打开数据文件夹，按其他选项卡上“Open data folder”会弹出一个 Finder 窗口。拖拽一个地景文件夹到 data 文件夹下的 data/Scenery 文件夹（或一个飞行器文件夹到 data/Aircraft 文件夹），就可以了。

3.1.3 FG_SCENERY

如果你想将下载的地景放到与安装位置不同的地方,可以设置 `FG_SCENERY` 环境变量。

FlightGear 会到此去寻找地景文件，可以在此按顺序列出要搜索的目录。在 UNIX （包括 Mac OS X）下用 ":" 分隔，在 Windows 下则用 ";"。

例如，Linux 下的 `FG_SCENERY` 环境变量可以设置成

```
/home/jsmith/WorldScenery:/usr/local/share/Flightgear/data/Scenery
```

首先会在 `/home/jsmith/WorldScenery` 搜索地景文件，之后则会去 `/usr/local/share/Flightgear/data/Scenery`.

在 Windows 下 `FG_SCENERY` 环境变量设置成

```
c:\Program Files\FlightGear\data\Scenery;c:\data\WorldScenery
```

首先去 `c:\Program Files\FlightGear\data\Scenery` 搜索地景文件，之后 `c:\data\WorldScenery`

在不同的平台设置环境变量的方法已经超出了本文的范畴，在此不多赘述。

3.1.4 飞行时获取地景

如果你能保证网络的连续性，*FlightGear* 支持在飞行时获取地景。首先为 *TerraSync* 创建一个空的目录，并允许用户可写，并给 *FlightGear* 设置好 `FG_SCENERY` 变量（如上文所述）。请**不要**把 *TerraSync* 下载的地景放到预先安装的地景目录

在 *FlightGear* 里，到 Environment（环境）菜单下选择 Scenery Download（地景下载）选项。然后选择上面创建的目录并选择 Enable Automatic Scenery（自动地景下载）。

TerraSync 最主要的一个好处是总能从 FlightGear World Scenery 项目下载到最新且最强大的地景文件，因此允许你在 World Scenery 发布之外（通常与 *FlightGear* 的发布保持同步）进行地景的增量更新。

3.1.5 独立运行 *TerraSync*

可以将 *TerraSync* 作为一个外部工具来运行。

在 Mac OS X 或 Windows，只需要在图形化启动器上选择 "Download scenery on the fly"，这样就会以一个独立的进程启动 *TerraSync* 自动下载你飞机周围的地景，因此你根本不用指定 FG_SCENERY 的地图。

另外你可以直接运行 terrasync 程序。它会告诉 *FlightGear* 使用 "Atlas" 协议，因此可以这样调用 *FlightGear*:

```
--atlas=socket,out,1,localhost,5505,udp
```

用命令行参数告诉 *TerraSync* 使用的端口号，以及各自的目录:

```
terrasync -p 5505 -S -d /usr/local/share/TerraSync
```

请注意 *TerraSync* （当用“-S”命令行参数时）将会使用 Subversion 通过 HTTP 下载。因此如果你的互联网连接使用了 HTTP 代理，请先确保并配置好“libsvn” Subversion 客户端使用代理。如果你使用的是 Mac OS X 10.5，图形化的启动器会在 svn 可用时自动指定 -S 选项。

3.1.6 创建自己的地景

如果你有兴趣想要生成自己的地景，可以先看 TerraGear ——*FlightGear* 里用于生成地景的工具：

http://wiki.flightgear.org/TerraGear

维护最活跃的 TerraGear 工具链源码树，就是与 *FlightGear* 托管在一起的 Mapserver：

http://mapserver.flightgear.org/git/gitweb.pl.

3.2 安装飞行器

FlightGear 基本包里只有少数的飞行器可以在 *FlightGear* 里面玩。开发者已经创建了大量的飞行器，从二战时的“喷火”（Spitfile）战斗机到现在的大型客机比如波音 747。

可以从这里下载这些飞行器：

http://www.flightgear.org/download/aircraft/

只需要将文件解压缩到你安装位置的 `data/Aircraft` 子目录下即可。飞行器是以 `.zip` 格式下载的。解压缩以后，将会在 `data/Aircraft` 目录下有一个包含此飞行器的子目录。下次你运行 *FlightGear*，新的飞行器就可以用了。

在 Mac OS X，可以使用图形化启动器来安装下载的飞行器文件，可以参考 3.1.2 节。

3.3 安装文档

上面提到的软件包都会有一份 PDF 版的《*FlightGear* 手册》，这样可以通过 Adobe Reader 打印下来。下载 Adobe Reader: [1] http://get.adobe.com/reader/

http://get.adobe.com/reader/

[1] 在 Linux 下可以安装自由开源的 Evince 软件 https://wiki.gnome.org/Apps/Evince。——译者注

此外，如果安装得当，HTML 版本的文档可以通过 *FlightGear* 的 `help` 菜单找到。

除此之外，源代码里也包括了一个 `docs-mini` 目录，里面有大量想法和解决方案来处理特殊的问题。这里也是一个很好的延伸阅读。

第二部分

在 *FlightGear* 中飞行

第四章
起飞：如何启动程序

4.1 启动模拟器

图 3：准备起飞：在旧金山国际机场（*KSFO*）的默认启动点等待起飞

大多数 FlightGear 的发布版本集成了完整的 FlightGear 启动器以便启动 FlightGear。只需要在开始菜单双击 `FlightGear Launcher`，或者桌面图标即可，还可以在命令行执行 `fgfs --launcher`。启动器可以让你方便的选择飞行器、起始位置（你甚至可以从距跑道 10 英里开外的地方开始进近，或者穿过一个特定的导航点）、时刻、启用或禁用 TerraSync 或实时天气，以及大量的其他设定。

第一次打开启动器的时候，你会看到对话框要求设置你的 `FG_ROOT` 变量，正常情况下应该会是 `c:\Program Files\FlightGear\data` 或者 `c:\Program Files\FlightGear 2016.1.1\data`。设置好以后你会看到如下图这样的界面：

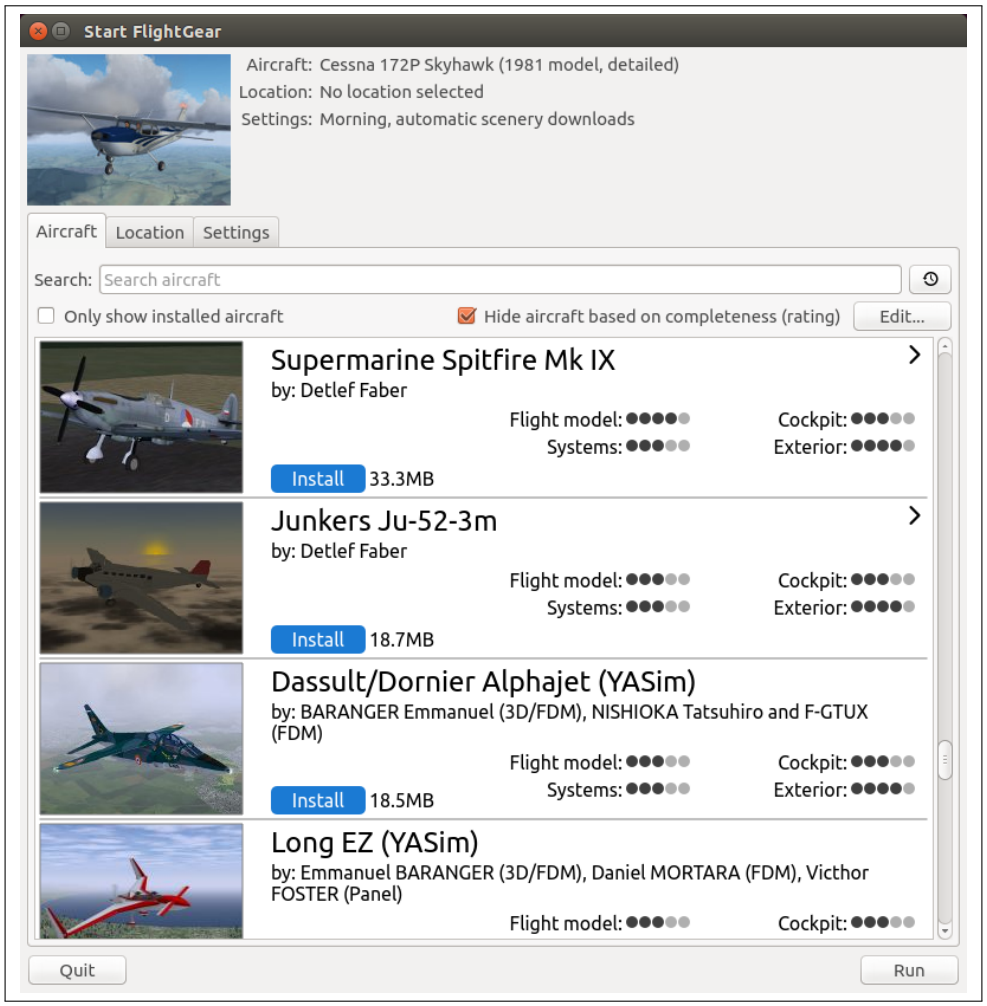


图 4：选择飞行器：选择飞行器并可以自动下载。

FlightGear 默认会预装塞斯纳 172P 和 UFO。你可以在列表里按 “Install” 按钮下载并安装其他任何飞行器。你也可以从官方网站或私人源下载飞行器。

启动器上有三个选项卡。之后的称为位置。只需在搜索框里输入你想启动时所处的位置，并按回车键。你可以从跑道、停机位开始，或者从跑道之外 10 英里的位置开始进近，或者以特定的高度、速度和航向穿过导航点。我们建议你从旧金山机场（KSFO）的任何一个停机位开始，也许是 A1 闸口？

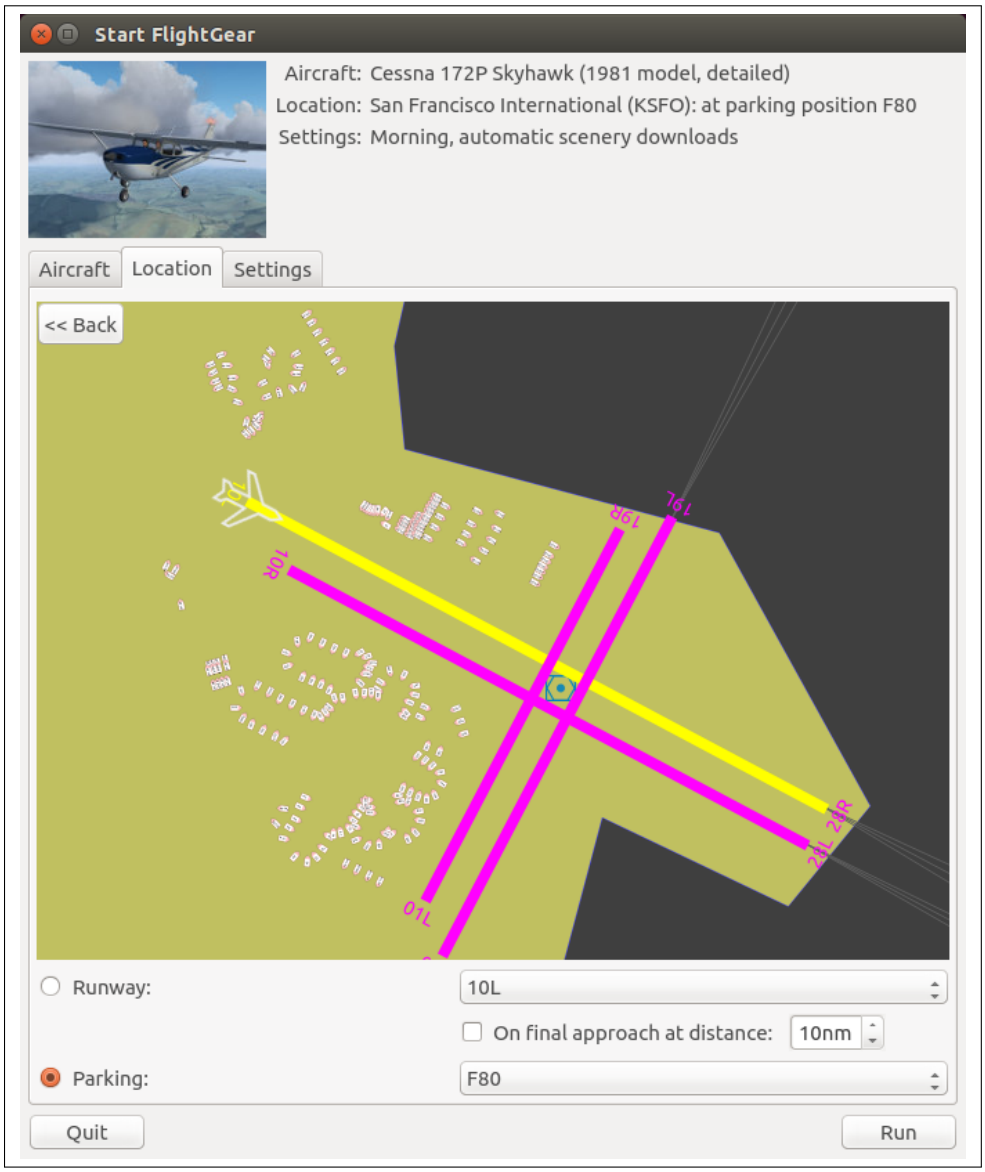


图 5：起始位置：选择起始位置在地面或者在空中。

最后一个选项卡是各种其他设置。你能设置时刻、季节，以及配置其他很多选项。在选项卡底部，有一个“额外选项”输入框，在此可以指定其他选项。如果你想使用它，请点击此输入框下面的链接。你可以在此页上设置，已下载的地景或飞行器的位置。

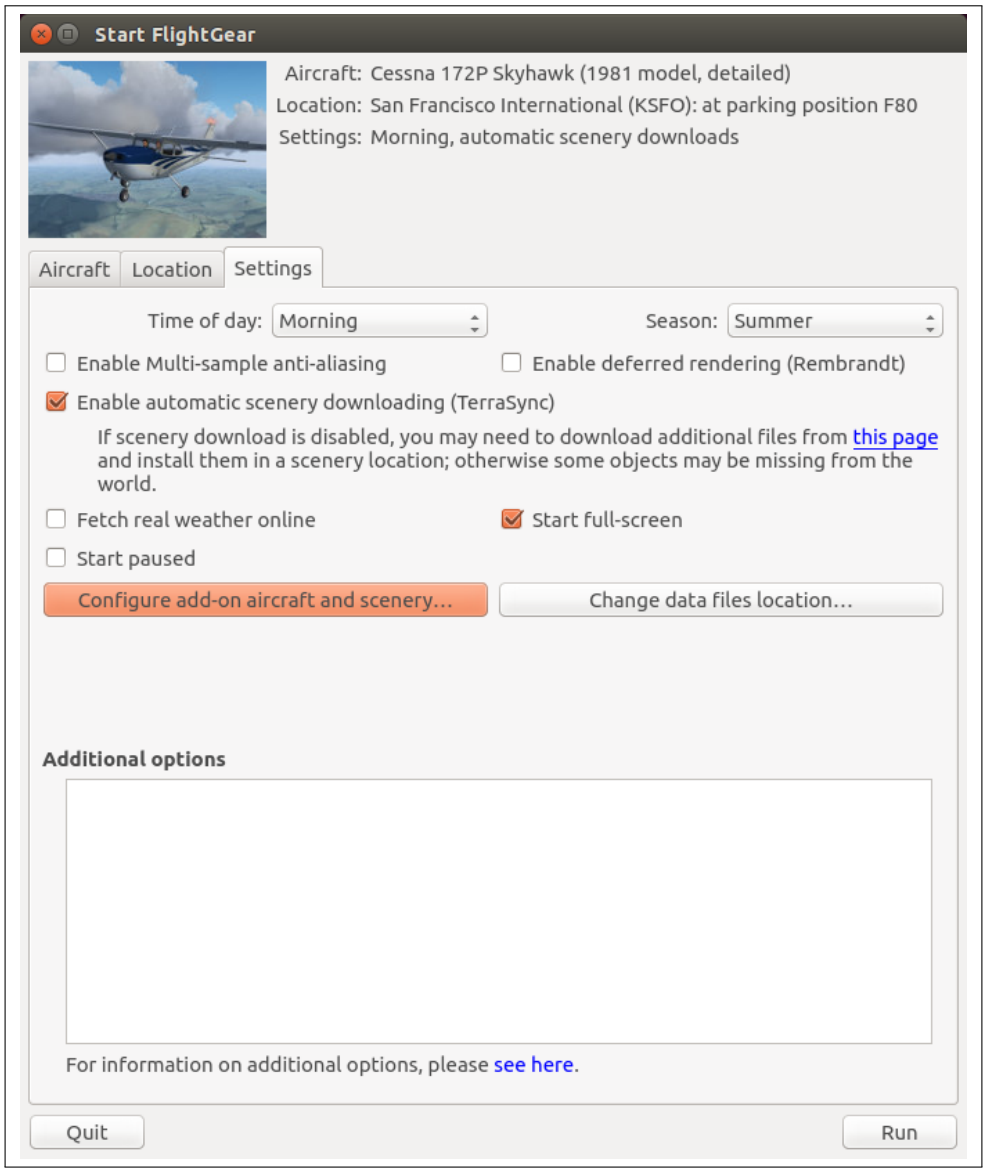


图 6：设置：设置其他模拟器选项。

当所有都设置好以后，就可以按“Run”按钮启动模拟器啦！

4.2 从命令行启动

另外，你可以从命令行启动 FlightGear。不过首先需要手动设置 `FG_ROOT` 和 `FG_SCENERY` 环境变量。

4.2.1 FG_ROOT

这里放置 *FlightGear* 将去哪里寻找数据文件，比如飞行器、导航台的位置、机场频率。在你安装 *FlightGear* 的 `data` 子目录下。比如：
`/usr/local/share/FlightGear/data` 或者 `c:\Program Files\FlightGear\data`.

4.2.2 FG_SCENERY

这里告诉 *FlightGear* 去哪里找地景文件。可以在此按顺序列出要搜索的目录。在 UNIX （包括 Mac OS X）下用 “:” 分隔，在 Windows 下则用 “;”。

```
/home/joebloggs/WorldScenery:/usr/local/share/FlightGear/data/Scenery
```

或者

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

4.2.3 在 Windows 下启动模拟器

打开一个命令行终端,切换到你安装的二进制目录下(一般是 `c:\Program Files\FlightGear\bin\Win32`），键入如下命令修改环境变量

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

并调用 *FlightGear*（在同一个命令行终端，环境设定只对当前的命令行有效），可以键入

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

命令行选项在 4.3 小节有详细介绍。当然你也可以用 Windows 文本编辑器（比如记事本）创建一个批处理文件，文件里输入上面这些命令。从最佳的性能考虑，运行 *FlightGear* 时建议最小化文本输出窗口。

4.2.4 在 UNIX/Linux 下启动模拟器

运行 *FlightGear* 之前，你需要设置一系列环境变量：

- 你需要添加 `/usr/local/share/FlightGear/lib` 到你的 `LD_LIBRARY_PATH`
- `FG_ROOT` 必须设置为 *FlightGear* 的数据安装目录。比如 `/usr/local/share/FlightGear/data`。
- `FG_SCENERY` 必须是一个包含地景的目录列表，用";" 分隔。这个变量的功效如同系统的 `PATH` 变量。
  比如：`$FG_ROOT/Scenery:$FG_ROOT/WorldScenery`。

在 Bourne shell（及兼容型）里添加这些变量：

```
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
```

或者在 C Shell (其兼容型)

```
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
```

当你设置好这些环境变量以后，只需要这样简单的启动 *FlightGear*

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

关于详细的命令行选项将会在 4.3 小节介绍。

4.2.5 在 Mac OS X 下启动模拟器

你也可以在 Mac OS X 的命令行里启动模拟器。打开 Terminal.app（在 `/Applications/Utilities`）并输入如下的命令：

```
cd /Applications/FlightGear.app/Contents/Resources
./fgfs --option1 --option2 ....
```

可以通过查看 4.3 小节获取具体的命令行选项。不像其他平台，若你使用预编译的二进制包，你不需要手动指定 FG_ROOT 和 FG_SCENERY 这样的环境变量。

4.3 命令行参数

下面列出的是 *FlightGear* 中可用的完整命令行选项和其简介。

如果你想连续多次使用这些配置，可以创建一个包含这些命令行参数的首选项文件，这将会在启动时自动加载这些配置。你可以用任何文本编辑器（记事本、Emacs、vi 任何你喜欢的）来创建这个配置文件。

- 大多数 UNIX 系统（包括 Mac OS X），将这个命令行选项的配置文件命名为 `.fgfsrc`，并放到你的 HOME 目录下。
- 在 Windows 系统下，将命令行配置文件命名为 `system.fgfsrc` 然后放到 `FG_ROOT` 所在的文件夹下(比如:`c:\Program Files\FlightGear\data`)

4.3.1 通用选项

- `--launcher`

  打开启动器（见前文）。

- `--help`

  显示相关的命令行选项。

- `--help --verbose`

  显示全部命令行选项。

- `--version`

  显示当前 *FlightGear* 的版本。

- `--fg-root=`*path*

  当你没有使用默认设置编译时，告诉 *FlightGear* 去哪里找到程序的根目录。

- `--fg-scenery=`*path*

  允许指定基础地景的路径,仅当地景没有安装到默认的 `$FG_ROOT/Scenery` 目录；在你使用 CD-ROM 上的地景时这样做非常有效。

- `--fg-aircraft=`*path*

  用来指定飞行器所在的目录，默认是 `$FG_ROOT/Aircraft`。

- `--language=`*code*

  用来指定会话的语言。比如 pl, nl, it, fr, en, de。

- `--restore-defaults`

  重置所有用户设置到默认

- `--enable-save-on-exit`, `--disable-save-on-exit`

  是否启用模拟器退出时自动保存用户首选项。

- `--enable-freeze`, `--disable-freeze`

  控制 *FlightGear* 启动时是否是暂停状态。默认是不暂停。

- `--enable-auto-coordination`, `--disable-auto-coordination`

  切换是否启用副翼与舵的协调配合。建议没有脚踏板或者游戏杆没有 Z 轴的用户打开自动配合。默认是关闭的。

- `--browser-app=`*path*

  指定你的网络浏览器的位置。例如：`--browser-app=`
  `"C:\Program Files\Internet Explorer\iexplore.exe"`
  （注意：“”之间的空格）

- `--config=`*path*

  载入外部路径的选项。例如：
  `--config=./Aircraft/X15-set.xml`

- `--units-feet`

  使用英制度量衡。

- `--units-meters`

  使用米制度量衡。

4.3.2 特性

- `--enable-ai-models`, `--disable-ai-models`

  启用或禁用模拟器里的其他飞行器/AI 模型。

- `--ai-scenario=`*name*

  启用一个特定的 AI 场景（例如 `--ai-scenario=vinson-demo`）。可以启用多个。

4.3.3 声音

- `--enable-sound`, `--disable-sound`

  启用或禁用声音。

- `--show-sound-devices`

  显示可用的声音设备。

- `--sound-device=`*device*

  指定音频设备。

- `--enable-intro-music`, `--disable-intro-music`

  启用或禁用 *FlightGear* 启动时播放一段音频示例。

4.3.4 飞行器

- `--aircraft=` 飞行器

  载入特定的飞行器。比如:`--aircraft=c172p`。可以查看 `$FG_ROOT/Aircraft` 目录来获取可用的飞行器，找到文件名结尾是 "`-set.xml`" 的文件。指定飞行器时，去掉文件名后面的 "`-set.xml`" 即可。另外，用 `--show-aircraft` 选项可以打印可用的飞行器的列表。关于如何下载和安装飞行器，请前往 3.2 节。

- `--show-aircraft`

  打印一个可用飞行器的列表。

- `--min-status=` 状态

  按照声明的开发状态来显示飞行器列表，可选的值有 `alpha`, `beta`, `early-production`, `production`。需要配合 `--show-aircraft` 选项。

- `--aircraft-dir=`*PATH*

  相对于执行位置的飞行器文件所在路径。默认 `$FG_ROOT/Aircraft`。

- `--vehicle=` 飞行定义文件

  与 `--aircraft` 相同。

- `--livery=`*Name*

  设置飞行器涂装。

4.3.5 飞行模型

- `--fdm=`*abcd*

  选择核心的飞行模型。可用的选项 `jsb`, `larcsim`, `yasim`, `magic`, `balloon`, `external`, `pipe`, `ada`, `null`。现在这个选项一般可以忽略，`--aircraft` 选项已经可以很好的设定 FDM。

- `--aero=`*aircraft*

  指定加载的飞行器空气动力学模型。现在这个选项一般可以忽略，`--aircraft` 选项已经可以很好的设定航空器模型。

- `--model-hz=`*n*

  以此速率（每秒迭代数）运行飞行动态模型。

- `--speed=`*n*

  让飞行动态模型运行速度快于真实时间。

- `--trim`, `--notrim`

  当初始化 JSBSim 时是否修正，默认会启用修正。

- `--on-ground`, `--in-air`

  指定启动时在地面（默认），或者在天空中。如果指定 `--in-air` 你必须同时使用 `--altitude` 来设定初始高度，你可以用 `--vc` 设定初始速度。请注意有些飞行器（特别是 X15）必须默认从空中启动。

- `--enable-fuel-freeze`, `--disable-fuel-freeze`

  控制燃油模式是否恒定（比如冻结）或者正常消耗（默认）。

4.3.6 初始位置和方位

- `--airport=`*ABCD*

  从一个特定的机场启动。机场用 ICAO 四字代码指定，比如 `--airport=KJFK` 表示纽约的 JFK 机场[1]。美国机场没有 ICAO 代码的，尝试在三字代码前加上字母“K”[2]。

- `--parking-id=`*ABCD*

  从一个特定的停机位开始。

- `--runway=`*NNN*

  从指定跑道的起始点开始（比如 28L）。如果没有指定跑道或者停机位 ID，会分配迎风起飞的跑道。

- `--vor=`*ABCD*, `--ndb=`*ABCD*, `--fix=`*ABCD*

  设定起始位置与 VOR、NDB 或 FIX（定位点）的相对位置。可以用来练习进近。

- `--carrier=`*NAME*

  从航空母舰上开始。查看 6.2 一节获取航母相关的选项。

- `--parkpos=`*NAME*

  指定航母上的起始位置。必须使用 `--carrier` 选项。默认在弹射起飞位置。

[1]纽约肯尼迪国际机场——译者注

[2]机场三字代码就是 IATA 代码，即“国际航运协会代码”，如纽约肯尼迪机场（KJFK）的 IATA 代码即为 JFK。——译者注

- `--offset-distance=`*nm*, `--offset-azimuth=`*deg*

  设定相对于特定 `--airport`, `--vor`, `--ndb`, `--fix`, `--carrier` 的距离偏移量和机头朝向偏移量。

- `--lon=`*degrees*, `--lat=`*degrees*

  起始于特定的经度和纬度，以十进制角度表示（西经和南纬用负数）。

- `--altitude=`*feet*

  起始于特定的高度。隐含表示了 `--in-air`。高度以英尺为单位，除非你设定了 `--units-meters`，则会使用米制高度。你也可以同时指定起始速度 `--vc` 以防启动后立刻失速。

- `--heading=`*degrees*, `--roll=`*degrees*, `--pitch=`*degrees*

  设定飞行器的起始方位（orientation）。所有值默认都是 0 ——航向正北平飞。

- `--uBody=`*X*, `--vBody=`*Y*, `--wBody=`*Z*

  设定飞机在 X、Y 和 Z 轴上的起始速度。速度以英尺每秒来表示，除非你还选择了 `--units-meters`，在这种情况下将会用米每秒来表示。

- `--vNorth=`*N*, `--vEast=`*E*, `--vDown=`*D*

  设定沿着南北、东西和垂直方向的初始速度。速度是以英尺为单位，除非你设定了 `--units-meters`，在这种情况下将会用米每秒来表示。

- `--vc=`*knots*, `--mach=`*num*

  设定起始空速以节或马赫数为单位。用于在 `--altitude` 选项时设定速度，除非你想一启动就立刻失速。

- `--glideslope=`*gradi*, `--roc=`*fpm*

  设定下滑道的起始角度以度为单位，或者用爬升率的英尺每分钟作单位。可以是正数也可以是负数。

4.3.7 环境选项

- `--ceiling=`*FT_ASL[:THICKNESS_FT]*

  设定阴云在特定高度，以及可选的厚度（默认 2000 英尺）。

- `--enable-real-weather-fetch`, `--disable-real-weather-fetch`

  控制是否启用实时获取天气信息。

- `--metar=`*METAR STRING*

  指定特定的 METAR 字符串。比如 `--metar="XXXX 012345Z 00000KT 99SM CLR 19/M01 A2992"` METAR 支持大多数格式（美国，欧洲）。不可与 `--enable-real-weather-fetch` 合用。

- `--random-wind`

  设定随机的风向和强度。

- `--turbulence=`*n*

  设定乱流的强度，从完全静稳（0.0）到暴虐（1.0）。

- `--wind=`*DIR@SPEED*

  指定地面风。风向用度表示，速度用节表示。可以在数值中用冒号表示范围；如：`--wind=180:220@10:15`。

- `--season=`*param*

  设定模拟的季节。可选的取值有 `summer`（默认），`winter`。

- `--visibility=`*meters*, `--visibility-miles=`*miles*

  设定能见度，用米或英尺表示。

4.3.8 渲染选项

- `--aspect-ratio-multiplier=`*N*

  设定显示器高宽比的倍数。

- `--bpp=`*depth*

  指定每个像素的位数。

- `--enable-clouds`, `--disable-clouds`

  启用（默认）或者禁用云层。

- `--enable-clouds3d`, `--disable-clouds3d`

  启用（默认）或者禁用 3D 云层。非常漂亮，但取决于你的显卡对 GLSL 着色器的支持，一些老显卡或不那么强的显卡可能不支持。

- `--enable-distance-attenuation`, `--disable-distance-attenuation`

  启用或禁用更加真实的跑道灯和进近灯衰减。

- `--enable-enhanced-lighting`, `--disable-enhanced-lighting`

  启用或禁用更加真实的跑道灯和进近灯。

- `--enable-fullscreen`, `--disable-fullscreen`
  启用，禁用（默认）全屏模式。
- `--enable-game-mode`, `--disable-game-mode`
  为 3DFX 显卡启用或禁用全屏显示。
- `--enable-horizon-effect`, `--disable-horizon-effect`
  启用或禁用地平线附近天体的拉长效果。
- `--enable-mouse-pointer`, `--disable-mouse-pointer`
  启用或禁用（默认）额外的鼠标指针。对老旧 Voodoo 显卡在全屏模式时很好用。
- `--enable-panel`, `--disable-panel`
  启用（默认）或禁用仪表板。
- `--enable-random-buildings`, `--disable-random-building`
  启用或禁用（默认）随机建筑物。注意随机建筑物会消耗更多内存。
- `--enable-random-objects`, `--disable-random-objects`
  启用（默认）或禁用随机地景物体。
- `--enable-random-vegetation`, `--disable-random-vegetation`
  启用（默认）或禁用随机的植物，比如树。需要支持 GLSL 着色器的显卡。在老旧的或者不强大的显卡下不要启用。
- `--enable-rembrandt`, `--disable-rembrandt`
  启用或禁用（默认）一些专家级特效包括增强的灯光和实时阴影。
- `--enable-skyblend`, `--disable-skyblend`
  启用（默认）或禁用雾/霾。
- `--enable-specular-highlight`, `--disable-specular-highlight`
  启用（默认）或禁用高光反射效果。
- `--enable-splash-screen`, `--disable-splash-screen`
  启用或禁用（默认）加速板初始化时旋转 3DFX 图标（只用于 3DFX）。
- `--enable-textures`, `--disable-textures`
  启用（默认）或禁用材质纹理。

- `--enable-wireframe`, `--disable-wireframe`
  启用或禁用（默认）线帧。如果你想知道 *FlightGear* 的内部运作机制，可以试试这个！
- `--fog-disable`, `--fog-fastest`, `--fog-nicest`
  设定雾的级别，默认会将雾里的渲染效果切断。如果禁用雾将可以看到更远，但是帧速率将会下降。使用 `--fog-fastest` 可以体验不那么真实的雾，可以提升帧速率。默认是 `--fog-nicest`。
- `--fov=`*degrees*
  设定视野角度。默认是 55.0 度。
- `--materials-file=`*file*
  指定渲染地景用的材质图文件。默认是:
  `Materials/regions/materials.xml`
- `--geometry=`*WWWxHHH*
  定义窗口/屏幕的分辨率。例如 `--geometry=1024x768`。
- `--shading-smooth`, `--shading-flat`
  使用平滑着色（默认），或者使用更快速但不那么漂亮的平面着色。
- `--texture-filtering=`*N*
  配置各向异性纹理过滤。取值有 1 （默认）、2、4、8 或 16。
- `--view-offset=`*xxx*
  允许基于默认的正前方视野角度增加偏移角度。可取的值有 `LEFT`, `RIGHT`, `CENTER`，或者角度数值。用于多窗口显示时。

4.3.9 HUD 选项

- `--enable-anti-alias-hud`, `--disable-anti-alias-hud`
  控制是否对 HUD（**H**ead **U**p **D**isplay，平视显示）启用防锯齿。
- `--enable-hud`, `--disable-hud`
  控制是否显示 HUD，默认为禁用。
- `--enable-hud-3d`, `--disable-hud-3d`
  控制是否显示 3D HUD。默认为禁用。
- `--hud-culled`, `--hud-tris`
  显示三角形剔除的百分比，或者 HUD 中三角形渲染的数量。主要方便图形开发者。

4.3.10 飞行器系统选项

- `--adf=`*[radial:]frequency*
  设置 ADF 频率和径向。
- `--com1=`*frequency*, `--com2=`*frequency*
  设置 COM1/COM2 无线电频率。
- `--dme=`*nav1|nav2|frequency*
  设置 DME 以 NAV1、NAV2 或指定特定的频率和径向。
- `--failure=`*system*
  设定飞行器的失效系统。可用的系统有 `pitot`, `static`, `vacuum`, `electrical`。使用多次可以设置多个失效故障。
- `--nav1=`*[radial:]frequency*, `--nav2=`*[radial:]frequency*
  设定 NAV1/NAV2 的无线电频率和径向。

4.3.11 时间选项

- `--enable-clock-freeze`, `--disable-clock-freeze`
  控制时间推移是正常或者冻结。
- `--start-date-gmt=`*yyyy:mm:dd:hh:mm:ss*,
  `--start-date-lat=`*yyyy:mm:dd:hh:mm:ss*,
  `--start-date-sys=`*yyyy:mm:dd:hh:mm:ss*
  指定精确的起始时间/日期。这三种功能的区别是指定不同的参考点，分别是格林威治时间，虚拟飞行时的本地时间，或者你的电脑的本地系统时间。
  不可与 `--time-match-local`, `--time-match-real` 选项同时启用。
- `--time-match-local`, `--time-match-real`
  默认是 --time-match-real：模拟器会读取系统时钟并使用。当你的虚拟飞行与你电脑在同一个时区时，这会非常合适，因为时钟是同步的。然而，当你飞到世界的另一端，就不一样了，因为和虚拟飞行和你实际时间之间有几个小时的时差。
  --time-match-local 选项可以解决这个系统本地时间与虚拟飞行的时差问题，将你的本地系统时间与虚拟飞行的时间同步。
  不可与 `--start-date-gmt`, `--start-date-lat`, `--start-date-sys` 选项同时启用。

- `--time-offset=`*[+-]hh:mm:ss*
  指定一个相对于之前时间选项的时间偏移量。

- `--timeofday=`*param*
  设定一天中的时段。可以用的值有 `real`, `dawn`, `morning`, `noon`, `afternoon`, `dusk`, `evening`, `midnight`。

4.3.12 网络选项

- `--multiplay=`*dir,Hz,host,port*, `--callsign=`*ABCD*
  多人游戏的选项和飞行器呼号。查看 6.1 节。

- `--httpd=`*port*, `--telnet=`*port*
  在特定的端口启用 HTTP 服务器或者 Telnet 服务器，以便访问属性树。

- `--jpg-httpd=`*port*
  在特定的端口上启用屏幕截图 HTTP 服务器。

- `--proxy=`*[user:password@]host:port*
  指定要使用的代理服务器。

4.3.13 航路/导航点选项

- `--wp=`*ID[@alt]*
  允许给 GC 自动驾驶指定一个导航点；也可以通过此选项的多个实例来指定多个导航点（比如一条航路）。

- `--flight-plan=`*[file]*
  如果你有多个导航点这个更加合适。你可以指定一个文件来载入。

4.3.14 IO 选项

注意：这些选项面向高级用户，明白自己做什么。

更多有关 IO 参数的细节描述在你 *FlightGear* 安装的 Docs 目录的 README.IO 文件里能找到。

- `--atlas=`*params*
  使用 Atlas 协议打开连接（用于 Atlas 和 TerraSync）

- `--atcsim=`*params*
  使用 ATC Sim 协议打开连接（atc610x）。

- `--AV400=`*params*
  打开连接以驱动一个 Garmin 196/296 系列 GPS。
- `--AV400Sim=`*params*
  打开连接以驱动一个 Garmin 400 系列 GPS。
- `--generic=`*params*
  使用通用（XML 定义）的协议打开连接。
- `--garmin=`*params*
  使用 Garmin GPS 协议打开连接。
- `--joyclient=`*params*
  连接 Agwagon 游戏杆。
- `--jsclient=`*params*
  连接远程游戏杆。
- `--native-ctrls=`*params*
  使用 FG 原生控制协议打开连接。
- `--native-fdm=`*params*
  使用 FG 原生 FDM 协议打开连接。
- `--native-gui=`*params*
  使用 FG 原生图形协议打开连接。
- `--native=`*params*
  使用 FG 原生协议打开连接。
- `--nmea=`*params*
  使用 NMEA 协议打开连接。
- `--opengc=`*params*
  使用 OpenGC 协议打开连接。
- `--props=`*params*
  连接非活动的属性管理器。
- `--pve=`*params*
  使用 PVE 打开连接。

- `--ray=`*params*

  使用 RayWoodworth 全动式座椅协议打开连接。

- `--rul=`*params*

  使用 RUL 协议打开连接。

4.3.15 调式选项

注意：这些选项仅限那些懂得自己在做什么的高级用户使用。

- `--enable-fpe`

  启用浮点数异常时中止。

- `--fgviewer`

  与载入整个模拟器不同，只载入轻量级的 OSG 查看器。用来检查飞行器建模。

- `--log-level=`*LEVEL*

  设定日志记录级别。可用的值有 `bulk`, `debug`, `info`, `warn`, `alert`。

- `--prop:[type:]name=value`

  设定属性，把值 `value` 赋给 `name`。

  例如：`--prop:/engines/engine[0]/running=true` 启动模拟器时引擎已经运转。

  其他例子：

  ```
  --aircraft=c172p
  --prop:/consumables/fuels/tank[0]/level-gal=10
  --prop:/consumables/fuels/tank[1]/level-gal=10
  ```

  为赛斯纳短途飞行加油。你可以指定属性类型 (`double`, `string`, `boolean`)。

- `--trace-read=`*params*

  追踪一个属性值的读取；可以使用多个选项。

- `--trace-write=`*params*

  追踪一个属性值的写入；可以使用多个选项。

4.4 游戏杆支持

你能想象一个飞行员只用键盘来控制塞斯纳飞机吗? 要想获得更好的飞行体验，你需要一个飞行游戏杆/驾驶盘，外加脚舵踏板。

FlightGear 已经集成了飞行游戏杆支持，会自动检测任何连接的游戏杆、驾驶盘或脚踏板。只需要连接上你的游戏杆并启动模拟器即可。

通过菜单 Help -> Joystick Configuration 你可以看到 *FlightGear* 已经配置好了你的游戏杆。这个对话框显示了游戏杆的名字以及每个按键和轴都设置到什么地方。你可以按一个按键或者移动游戏杆来查看具体的映射。

如果你有一个普通的游戏杆，也许已经有人已经在 FlightGear 里面为它设定好了，你进入游戏就可以飞了！ 如果你想改变某一个按键或轴的配置，只需要用游戏杆配置对话框来编辑即可。

如果你的游戏杆不太常见，那么 *FlightGear* 将默认使用一个简单的游戏杆配置。若你想改变配置，只需要用游戏杆配置对话框来编辑即可。配置会即刻起效，并一直保存到你下次飞行。

第五章
飞行中：仪表板，键位和菜单

本章将会讲述控制程序主系统并驾驶飞机。这里假设读者已经熟悉飞行，也许是从其他模拟器里习得。如果你对飞行是完全的新手，对你来说是第 七 章的教程是更好的材料，可以学习到如何在 *FlightGear* 里飞行。

以下这里可以找到一张包含标准键位的单页。

http://www.flightgear.org/Docs/FGShortRef.pdf.

大多数键盘控制可在模拟器的 Help 菜单下找到。

5.1 启动发动机

取决于飞行器的类型，你可能需要在飞行前启动发动机。下面的讲述是通用的。请查看飞行器的帮助或者飞行器教程来获取更多信息。

启动发动机以后，你需要检查停留刹车是不是开启，如果开启可以按“B”键释放刹车。

5.1.1 活塞式飞机

对活塞式飞机而言，磁电机点火系统用“{”和“}”键来控制。对大多数飞机而言，用“s”键来使用点火器。对有多台发动机的飞机而言你可以选择控制使用哪个发动机。使用“~”键选择所有发动机同时起动。大多数磁电机有四个档位——关断（OFF），左（LEFT），右（RIGHT）和双侧（BOTH）。因此要起动选择的发动机，按“}”键三次，然后再长按“s”键。

注意，二战时代的战斗机，其发动机起动过程一般比较复杂。请查询相应飞行器的帮助文档。

5.1.2 涡轮螺旋桨飞机

起动涡轮螺旋桨发动机只需要将油门杆从关断位（OFF）移动到慢车位（Idle）即可，使用"m" 键。

5.1.3 喷气式飞机

起动一台喷气式发动机显然更加复杂，而控制方法也与不同的飞行器有关。

1. 设置关断开关（cutoff）在 ON 位
2. 打开点火器
3. 当发动机的脱机转速达到 N1 的 5% 时，设置关断开关到 OFF 位。
4. 当发动机转速达到正常转速时，关闭起动器。

5.2 键盘控制

虽然 *FlightGear* 支持操纵杆、驾驶盘和方向舵踏板，你依然可以只用键盘搭配鼠标来飞行，下面会有详细介绍。然而为了让你更好的控制飞机，你需要至少使用键盘做一些控制。

下面的这些键位绑定并不是硬编码的，而是用户可调整的。查看和修改 *FlightGear* 主目录下 `keyboard.xml` 文件来配置键位设定。这是一个人类可读的 ASCII 文件。对初学来说，修改此文件并不明智，不过高级用户可以按照自己的想法修改键位绑定，比如适应其他模拟器的习惯。

5.2.1 飞行器控制

为了可以在飞行中完全控制飞机，你需要确保 `NumLock` 打开，且 *FlightGear* 窗口在焦点位置。下面这些按键控制飞行器的主要控制面。

按键	功能
9 / 3	油门
4 / 6	副翼
8 / 2	升降舵
0 / 回车	方向舵
5	副翼/升降舵/方向舵回中
7 / 1	升降舵调整片

表 1：飞行器主控制

以下键位控制发动机。

按键	操作
!	选择一号发动机
@	选择二号发动机
#	选择三号发动机
$	选择四号发动机
~	选择所有发动机
{	减少所选发动机的磁电机档位
}	增加所选发动机的磁电机档位
s	为所选的发动机点火启动
M / m	贫油/富油所选发动机的油气混合比
N / n	减少/增加所选发动机的螺旋桨转速

表 2: 发动机控制按键

以下按键控制次要的飞行器系统。

按键	操作
b	启用所有刹车
, / .	启用左右轮刹车
	(用于差动刹车)
l	切换尾轮锁定
B	切换停留刹车
g / G	收起/放下起落架
空格	按键通话（PTT）
- / _	多人飞行时文字聊天菜单/输入
[/]	收起/放下襟翼
j / k	收起/张开扰流板
CTRL-B	切换减速板

表 3: 飞行器次要系统控制

5.2.2 模拟器控制

要修改视觉角度，你需要先禁用 `NumLock`。以下是可用的控制方式:

数字键	视觉角度
Shift-8	前方
Shift-7	左前方
Shift-4	左侧
Shift-1	左后方
Shift-2	后方
Shift-3	右后方
Shift-6	右侧
Shift-9	右前方

表 4：视觉角度

另外，以下按键可以让你控制显示：

按键	操作
P	切换仪表板开关
c	切换 3D / 2D 驾驶舱（如果这两个都可用）
S	循环显示面板样式，完全/迷你
CTRL-C	切换仪表板/驾驶舱热点的可见度
h	切换 HUD
H	改编 HUD 亮度
i / I	最小化/最大化 HUD
x / X	放大/缩小
v / V	来回循环显示模式
CTRL-V	重置显示模式到飞行员视角
z / Z	增加/减少能见度（雾）
F10	切换菜单显示
Shift-F10	切换全屏模式

表 5：显示选项

最后，下面是通用的模拟器控制。

按键	操作
P	暂停模拟器
a / A	模拟器速度增加/减慢
t / T	时钟加速/减慢
CTRL-R	即时回放
F3	保存屏幕截图
ESC	退出程序

表 6：通用模拟器控制。

5.2.3 自动驾驶控制

FlightGear 支持两种类型的自动驾驶——一种是通用自动驾驶可以支持所有飞机（即便是通常没有安装自动驾驶的），另一种是飞机特有的自动驾驶，可以在驾驶舱里控制。

通用自动驾驶可以用如下按键来控制：

按键	操作
退格键	切换自动驾驶
CTRL-A	切换高度保持
CTRL-G	切换下滑道保持（NAV 1）
Ctrl-H	切换航向保持
Ctrl-N	切换 NAV 1 锁定
Ctrl-S	切换自动油门
Ctrl-T	切换地形跟踪锁定（AGL）
Ctrl-U	为你的高度增加 1000 ft（紧急）
F6	切换自动驾驶航向模式
F11	自动驾驶高度对话框
8 / 2	高度调节
4 / 6	航向调节
9 / 3	自动油门调节

表 7：自动驾驶控制。

CTRL + T 特别有趣，可以让你的飞机像巡航导弹一样依循地形飞行。CTRL + U 可以让你在感觉即将坠毁时很方便的接手。

5.3 鼠标控制动作

除了选择菜单项和点击驾驶舱里的控制装置，你的鼠标在 *FlightGear* 里还能有更多其他有用的功能。

有三种鼠标模式：正常模式（默认），控制模式和查看查模式，你可以通过 Tab 键来切换这些模式。

5.3.1 正常模式

在正常模式，你可以控制菜单和仪表板。这个模式用一个普通的鼠标箭头来表示。

要控制一个开关或者按钮，只需要使用左键或右键点击即可。

要旋转一个无线电频率旋钮或者线性控制，比如油门杆，点击左侧来减少数值，点击右侧来增加数值。点鼠标左键可做小幅度调整，点鼠标中键可做大幅调整。有些控制，比如无线电也可以使用鼠标滚轮，而另一些比如油门，支持鼠标左键拖拽。

使用 CTRL-C 来高亮显示物体上可点击的热点。

5.3.2 控制模式

在控制模式下，你通过移动鼠标来控制飞行器，此模式使用一个十字作为鼠标指针。

在此模式下，移动鼠标向左或向右可控制副翼让飞行器滚转。移动鼠标向前或向右控制升降舵可改变飞行器的俯仰姿态。

按住左键可以改变控制行为，这样左右移动鼠标就可以控制方向舵。按住鼠标中键并前后移动鼠标就可以控制油门。

最后，鼠标滚轮则可以用来设置升降舵调整片。

若你没有游戏杆的时候使用此模式非常有用，可以提供比键盘更好的飞行器控制。如果你打算经常使用鼠标控制飞行器，建议打开自动协调选项，这样副翼和方向舵会一致控制。可以在启动模拟器时指定 `--enable-auto-coordination` 选项来启用这个功能，或者在启动器里勾选自动协调（auto-coordination）选项。

5.3.3 查看模式

在查看模式，你可以使用鼠标四处围观。此模式使用一个双箭头当鼠标指针。

移动鼠标可以平移和倾斜当前位置的视图。此模式非常有用，可以在驾驶舱里方便观察，或者从侧窗向外看。鼠标滚轮可以控制放大和缩小。点左键可以回到初始位置，通常是平视前方。

按住中键并移动鼠标可以让你移动视点本身，向左向右向上向下。按住 CTRL 和鼠标中键并移动鼠标，可以让视点前后移动。

5.4 菜单选项

菜单栏提供了访问模拟器和飞行器各种功能的入口。很多飞行器也有自己的菜单项，比如修改注册号和自动起动发动机。这些都可以在菜单栏的末尾看到。

要显示或者隐藏菜单栏，请按 F10 键。也可以移动鼠标到屏幕的最上边缘，菜单栏会自动显示出来。

菜单栏提供了如下这些菜单和功能。

- **File**
  - **Reset (Shift-Esc)** 重置到选定的起始位置。可以让你在发生问题的时候快速上手。
  - **Load Flight Recorder Tape (Shift-F1)** 载入一段提前录制好的飞行回放。

- **Save Flight Recorder Tape (Shift-F2)** 保存当前飞行，方便以后回放。
- **Screenshot (F3)** 保存屏幕截图,文件名格式是 `fgfs-screen-XXX.jpg`。
- **Screenshot Directory** 设置屏幕截图存放的目录。
- **Sound Configuration** 配置各声音通道的音量，以及是否可以在飞机外听到。
- **Input Configuration** 配置鼠标的行为。
- **Scenery Download** 配置是否使用 *TerraSync* 特性自动从互联网下载地景。
- **Aircraft Center (Experimental)** 让你可以从模拟器界面下载和安装飞行器，试验性的，并不完善。
- **Quit (Esc)** 退出程序。

• **View**

- **Toggle Fullscreen (Shift-F10)** 切换模拟器的窗口模式或全屏显示模式。
- **Rendering Options** 配置各种图形特性。可以配置各种养眼的阴影、3D 云层以及镜面反射等等，会影响帧速率。为了帮你更好的平衡各个选项，在 Display 选项菜单启用 “Show Frame Rate’’（显示帧速率）选项，可以在画面右下方显示当前的每秒帧数。大多数人发现帧速率在 20fps 时就足够飞行了。帧速率取决于启用的图形特性、当前的能见度（用 z / Z 键来设置）、视野内的物体及其层次程度（Level of Detail，LOD）。
- **View Options** 配置各种视图选项，包括什么可以显示出来，2D 仪表板、帧速率和聊天信息等。
- **Cockpit View Options** 配置驾驶舱内的视图。飞行员头部移动、高 G 力时的黑障和反向 G 力时的红视现象[1]。
- **Adjust LOD Ranges** 设置物体的精细度显示范围，这会影响模拟器中材质和物体的显示效果。
- **Adjust View Position** 设置当前的视点偏移量。你可以通过拖动滑块来设置。除此之外，你也可以用鼠标来做小幅调整（见上文）。
- **Adjust HUD Properties** 设置 HUD（平视显示器）属性，比如透明度以及是否使用抗锯齿。

[1]因为急速爬升等机动飞行动作，产生较大的正向 G 力，导致飞行员的血液向下肢流动，会让头部缺血而产生视力障碍的现象称之为 “黑障”；反之当快速下降等反向 G 力较大的机动动作发生时，血液会向头部和眼部流动，造成视网膜充血，因此会产生 “红视” 现象。这两种生理现象对人体都有一定损害。——译者注

  - **Toggle Glide Slope Tunnel** 显示一条虚拟的下滑道，引导你沿着正常的进近路径降落到跑道上。可以在进近降落有困难时帮助你。
  - **Instant Replay (Ctrl-R)** 控制即时回放选项——检查你降落姿态的好工具！
  - **Earthview Orbital Rendering** 控制地球轨道视觉渲染，基于 NASA 可视化地球图片集。
  - **Stereoscopic View Options** 配置立体显示，使用红/绿眼镜或者其他显示方式。

- **Location**

  - **Position Aircraft In Air** 放置飞机于空中一个随意的点。你必须指定一个已知的地面点，比如一个机场、VOR、经纬度坐标，以及相对于该点的距离、位置和高度。你也可以设置起始速度和航向。练习进近时可以用这个。
  - **Select Airport** 将飞机放在一个机场。你可以搜索几乎所有你已安装的机场，查看当前 METAR，空域信息，空域外观以及选择跑道或停机位。
  - **Random Attitude** 设置飞机在一个随机的高度、航向和速度。可以用来练习从一个不寻常的高度改出。
  - **Tower position** 配置机场塔台的位置，用于配置塔台视点观察以及塔台锁定视点。

- **Autopilot** 此菜单仅在飞行器默认配备自动驾驶仪时可用。一些飞机的自动驾驶是通过其仪表板来配置的，在这种情况下此菜单不可用。

  - **Autopilot Settings (F11)** 配置飞行器的自动驾驶。你可以设置自动驾驶的各种方式——从简单的保持机翼水平到沿 ILS 进近。
  - **Route Manager** 设置自动导航的航路（航路点）列表。航路点可以是机场也可以是定位点。HUD 上会显示航向、距离和前往当前航路点的时间。
  - **Previous Waypoint** 从航路列表上选择前一个航路点。
  - **Next Waypoint** 从航路列表上选择下一个航路点。

- **Environment**

  - **Weather** 可以设置当前天气，选择天气场景，或者使用最近的气象情报点（通常是机场）的 METAR 天气报告。你要么使用基本天气，也就是会明确设置天气状况；或者使用高级天气，模拟更大的天气系统，不过会牺牲本地的精确性。

  - **Environment Settings** 配置季节材质贴图（夏季或冬季），以及地面状态，比如雪线和扬尘等。
  - **Time Settings** 允许你设置模拟器的当前时间，加速模拟，以及改变模拟器的时间变化率。也会显示 UTC 时间和本地时间。
  - **Wildfire Settings** 配置飞机坠毁时是否产生真实的火焰效果，这可能会波及，也可以用机上适当的灭火瓶扑灭。

- **Equipment**
  - **Map** 显示一个可移动的地图，会显示出机场、导航台等
  - **Map (Canvas)** 显示另一个可移动的地图，使用 Canvas
  - **Map (opens in browser)** 在网页浏览器里显示地图（需要运行内部 Web 服务器）
  - **Stopwatch** 显示一个简单的秒表，可用于仪表进近。
  - **Fuel and Payload** 可以设置燃油量和当前的飞行器载重。只在一部分飞行器上可用。
  - **Radio Settings (F12)** 设置无线电通话和导航设备的频率和径向。
  - **GPS Settings** 配置 GPS 导航点和显示的航道信息。
  - **Instrument Settings** 你可以设置高度表气压值以及航向指示器的偏移量。
  - **Random Failures** 配置随机的飞行器系统和仪表失效。此设置为平均故障间隔时间（MTBF）或平均故障间隔周期（MCBF）。
  - **System Failures** 配置随机的飞行器系统失效，比如真空系统。
  - **Instrument Failures** 配置特定仪表的随机失效。

- **AI**
  - **Traffic and Scenario Settings** 配置活动的 AI 场景。注意这只在模拟器重启以后才会起效。
  - **ATC Services in Range** 显示邻近机场的通讯频率。
  - **Wingman Controls** 控制 AI 僚机（取决于飞行器类型）。
  - **Tanker Controls** 如果飞行器支持，可以动态的创建一个空对空加油机，参考 6.8 节获取更多信息。
  - **Carrier Controls** 控制 AI 航母。
  - **Jetway Settings** 控制某些机场的廊桥。

- **Multiplayer**

  - **Multipler Settings** 启用多人游戏，设置呼号和服务器。
  - **FGCom Settings** 配置与其他多人游戏参与者的语音通话。
  - **Chat Dialog** 允许你在多人环境里与其他人聊天。
  - **Chat Menu (-)** 可以给其他飞机或 ATC 发送普通聊天信息。一些菜单含有子菜单选项。
  - **Pilot List** 显示当前范围里的其他玩家，包括他们的距离、航向和高度。
  - **MPCarrier Selection** 显示可用的 MPCarriers（多玩家航母）列表。

- **Debug** 此菜单包括了本指南范围以外的选项。

- **Help**

  - **Help** 在浏览器窗口打开帮助系统。
  - **Aircraft Help** 显示相应飞行器的信息。
  - **Aircraft Checklists** 如果可用，显示特定飞行器的检查单。
  - **Common Aircraft Keys** 显示控制飞行器的基本按键列表。
  - **Basic Simulator Keys** 显示控制模拟器的基本按键列表。
  - **Joystick Configuration** 在模拟器里配置游戏杆，包括轴和按键的分配。
  - **Tutorials** 可以在模拟器里启动有关当前飞机的教程。只对某些飞行器可用，请看下一章的教程以获取更多信息。
  - **About** 显示此版本 FlightGear 的各种信息。

5.5 仪表板

图 6：塞斯纳 *172* 的 *3D* 驾驶舱。

FlightGear 里的飞行器可以有二维仪表板和三维驾驶舱。3D 驾驶舱提供了更加真实的飞行员视角，但对小显示器来说不太容易获取信息。

默认的塞斯纳 172P（c172p）既有三维也有二维的驾驶舱。三维驾驶舱在启动 *FlightGear* 时默认开启。但你也可以从菜单 `View->Display Options->Show 2D Panel` 选择显示二维的仪表板，或者按“P”键。

所有仪表板上的控制杆和旋钮可以通过鼠标来操作，只需鼠标左键/右键点击相应的控制杆/旋钮即可。要想大幅改变，用鼠标中键。通常情况，在控制杆/旋钮的右侧点击会增加数值，左侧则会减少数值。

很多仪表（特别是无线电）也支持使用鼠标滚轮来改变数值。其他控制（比如油门）也可以使用鼠标左键拖动。

5.6 平视显示器

FlightGear 还提供了 HUD（**H**ead **U**p **D**isplay，平视显示器）。HUD 一般在军用飞机和非常先进的喷气式飞机上能看到。然而 *FlightGear* 可以在很多通用飞行器上使用 HUD。要启用 HUD，只需按“h”键。

图 7 中的 HUD 显示了飞机的主要飞行参数。中间你可以找到俯仰指示器（以度表示），其上方是副翼指示器而下方是方向舵指示器。相应的左侧滑块是升降舵指示器，旁边则是升降舵调整片指示器。而最下方则是一个简单的转弯指示器。

最左侧边缘有两个标尺：里侧表示速度（以节为单位）外侧表示油门位置。最右侧边缘的标尺表示高度——左侧的表示与地面之间的距离（高，Height），而右侧表示相对海平面的高度，两者都以英尺为单位。

除此之外，HUD 还提供了其他信息。左上方可以找到日期和时间，连同当前位置的纬度和经度。

你可以使用“H”键或“h”键来修改 **HUD** 的颜色。用“i / I”来最小化/最大化 HUD。

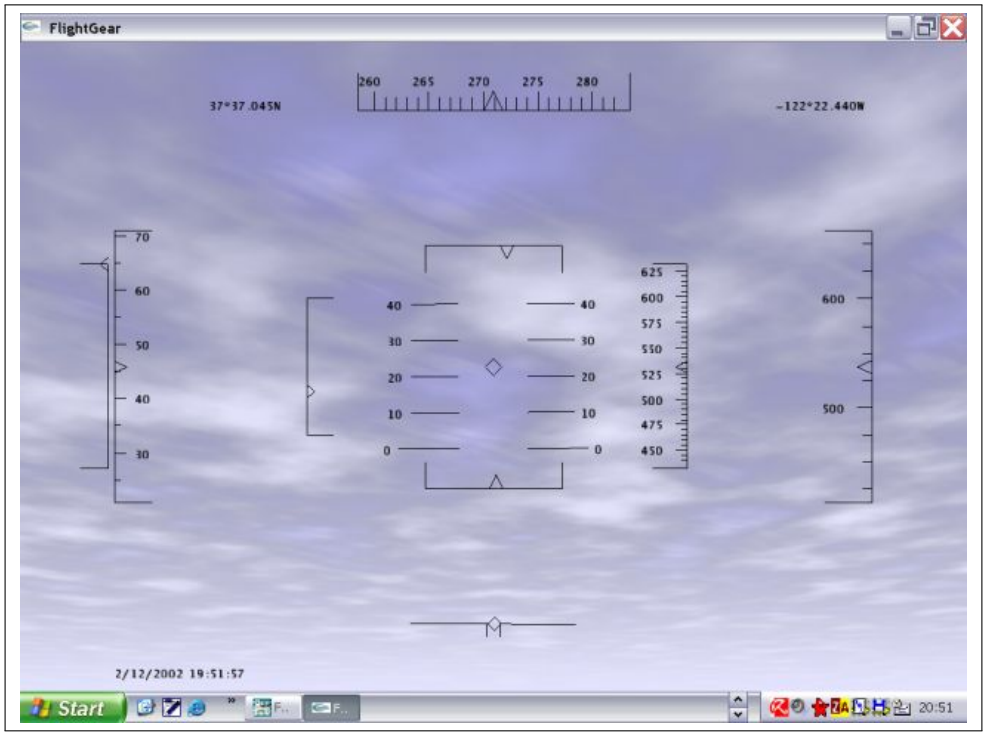


图 7：*HUD*，平视显示器。

第六章
特性

FlightGear 包括了很多特性，其中很多对新用户来说并不容易找到。本章将会讲解如何启用并使用这些高级特性。

很多特性依旧还在开发中，因此这里所列的很多信息并不一定及时。想获取最新的信息(和新特性),请访问 *FlightGear* 维基,http://wiki.flightgear.org/。

6.1 多人游戏

FlightGear 支持多人游戏环境，你可以与其他连线的飞友共享天空。有关服务器信息和在线用户（以及他们的飞行状态），可以查看多人游戏地图:
http://mpmap02.flightgear.org

点击“服务器”选项卡可以看到多人游戏服务器列表。

6.1.1 快速开始

你可以通过打开 Multiplayer（多人游戏）菜单下的多人游戏对话框进入一个多人游戏环境。只需要从列表里选择离你最近的服务器，输入呼号（这样其他人就可以看到你），并点击“Connect”（连接）即可。

要查看区域里的其他飞行员，查看 Multiplayer（多人游戏）菜单下的 Pilot List（飞行员列表）。

所有多人服务器都是互联互通的，因此没有必要连接到和其他人相同服务器里。

6.1.2 其他方法

如果你要连接一个非标准服务器，或者以上方法不管用，你可以使用下面的这些方法。

使用 FlightGear 启动器

FlightGear 启动器最后一个页面有可以选择多人游戏的选项。只需要勾选，并输入上面方法找到的主机名称和端口号，再输入你的呼号来称呼你自己。你的呼号最多 7 个字符。你必须在“Features”（特性）里勾选“AI 模型”这样可以看到其他飞友的飞机。

使用命令行

给 fgfs 传递多人游戏的基本参数有：

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

1. <portnumber> 是 TCP 端口号，比如 5000。
2. <server> 多人服务器的名称，比如 mpserver01.flightgear.org。
3. <client> 你电脑的名称，或者 IP 地址，也就是使用 FG 连接服务器的 IP 地址——即使是本地 192.168 这样的地址。比如 192.168.0.1
4. <callsign> 你自己的呼号，最多 7 个字符，比如 N-FGFS。

当模拟器载入完毕，你应该能在地图上看到你自己。如果不行，检查命令行的错误信息并查看下面的“故障排除”一节。

6.1.3 故障排除

多人游戏要想工作，需要知道我们电脑的 IP 地址并且能够连接到服务器。如何获取这些信息依赖于你的配置，如下：

使用 USB 调制解调器连接互联网的情况

首先你要知道运行 FlightGear 多人游戏网络的 IP 地址。如果是通过插在 USB 接口上的 ADSL 调制解调器连接互联网的话，你可以通过 http://www.whatismyip.com 网站找到自己的 IP 地址。请注意此 IP 地址并不固定，可能会经常改变——如果多人游戏不能用，首先检查这个[1]。

[1]中国大陆可以使用 http://ip138.com/ 来获取公网的 IP 地址。——译者注

使用以太网调制解调器连接互联网

或许，你是通过连接在你电脑 RJ-45 接口上的路由器连接，或者以太网连接器（类似西方电话的那种接头），或者使用无线连接。你需要知道相应网络接口的 IP 地址。

如果是 Linux 系统，可以用 root 用户登录并输入 “ifconfig”。你能看到列表里多于一个网络接口，以 “lo” 开头的可以忽略。你需要找到类似 “eth0” 或者 “wlan0” 这样的，仔细看并找到 “inet addr”，后面的数字就是要找的 IP 地址，比如 “inet addr:192.168.0.150” [1]。

在 Windows XP 下，点击 “开始”，然后选 “运行”，并输入 “cmd”。在命令行窗口里输入 “ipconfig”，就可以看到 IP 地址了，记下来。

在 Windows 98 下，点击 “开始”，然后选 “运行”，并输入 “cmd”。在命令行窗口里输入 “winpcfg”，就可以看到 IP 地址了。

如果依旧不行

你**必须**给出你本地的，路由器之后的 IP 地址，这样可以连接到多人服务器。请相信我！

你还要确保防火墙没有阻碍——临时关闭或者把 5000 端口加入允许入站例外。

如果你依旧不能连接，请前往 FlightGear IRC 频道，有人会愿意出手协助。

6.2 航空母舰

FlightGear 支持在尼米兹号（游曳在旧金山湾附近）、卡尔·文森号、圣安东尼奥号、福煦号和艾森豪威尔号航空母舰上的操作。航母配备有弹射起飞装置，阻拦索，升降机，TACAN 和 FLOLS。要启用这些航母，你需要使用文本编辑器编辑 $FG_ROOT 目录下的 preferences.xml 文件，找到 “nimitz” 这样的文字，你可能会看到这样的：

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

你需要删掉注释标记，变成这样：

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

同时确认相关的 AI 选项为 “true”。保存文件并退出编辑器。

[1] Linux 下还可以使用 `ip a` 命令，找到类似 “eth0” 这样的网络接口，并在其下找到形如 `inet 192.168.0.150` 这样的数字即可，主要用在无法使用 root 权限登录时。——译者注

6.2.1 从航母上开始

要想让飞机从航母上开始起飞，启动 FlightGear 前，在命令行使用如下参数：

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

注意“Nimitz”的首字母“N”要大写。

如果你使用 Windows 或者 OS X 启动器来运行 FG，你会发现图形界面有一个文本输入框允许你输入特定的命令，输入如上的选项即可。

有很多飞行器都适合在航母上弹射起飞，不过 seahawk 是最容易飞行的。

6.2.2 从弹射器起飞

当 FlightGear 启动以后，你要确保停留刹车为 OFF（关断）并按住“L”键来勾住弹射器挂钩。必须一直按住“L”键直到弹射器挂钩勾住飞机。你应该会观察到飞机被拉到对准弹射器的位置，并看到吊索出现并勾住飞机。这些只会发生在你的飞机已经很接近弹射器的时候；作为粗略的指南，默认的停机位 seahawk 的机鼻应该大致指向甲板上的观察者座舱。

要想让航母进入最佳弹射位置，选择“ATC / AI”菜单，然后选择“AI Carrier”（AI 航母）下面的“Turn into wind”（转入迎风面）选项。应该会看到航母开始加速并转入迎风面，随着航母转弯甲板自然也有所倾斜。你要等待航母转弯完毕甲板恢复水平，才能进入下一个步骤。

连接到弹射器以后，你需要将发动机推到全速，确保停留刹车在关断位，所有飞行控制符合弹射条件（seahawk 会紧贴在右后侧）。准备好以后，按“C”键释放弹射器。你的飞机将会被弹射离开甲板，此时你可以收起起落架并缓慢爬升，小心不要失速。

6.2.3 找到航母——使用 TACAN

在一片开放水域找到航母是一件很困难的事情，特别是当能见度比较差的时候。为了能够协助这项任务，尼米兹级航母配备有 TACAN（塔康）系统，帮助相应的飞机（包括 Seahawk）来获取并导航到当前范围内的航母。首先你需要设置相应的 TACAN 频率，此例中是 029Y，在无线电对话框（CTRL-R 或者从 FlightGear 菜单里选择 Equipment / Radio）。你必须观察 DME 上显示的航母到你飞机的距离，以及 ADF 仪表（Seahawk 上的 ADF 在 DME 的旁边）指示的航母方位。转到此方位，你将会看到 DME 上的距离显示你正在接近航母。

6.2.4 在航母上降落

在真实当中，降落操作最难。你能找到 Andy Ross 写的 A4 Skyhawk 教程来帮助你：
http://wiki.flightgear.org/A-4F_Skyhawk_Operations_Manual

当你用 TACAN 系统定位航母以后，你需要正对飞行甲板的后方。飞行甲板与船舷有一定的角度，你要经常调整姿态以正对飞行甲板。确保飞机已经设置好降落（Help / Aircraft 菜单里会有飞机相关的帮助信息）起落架和阻拦钩已经放下。

在进近到航母时你可以看到在甲板左侧，有一系列彩色的灯光——这就是所谓的“菲涅尔透镜光学助降系统”（Fresnel Lens Optical landing System，FLOLS）。这将会帮助飞机调整在正确的下滑道上。你可以看到横向有一排绿色的灯光，当处于下滑道中间时，橙色的灯（也就是所谓的“meatball”（肉球））会与绿色的灯光在一条直线上。若进近正确 meatball 会与绿色的灯光共线，如果高于下滑道，meatball 也会向上移动，反之则会向下移动，如果非常低 meatball 会变成红色。如果你一直保持 meatball 在正中位置，你将会钩住第三条阻拦索。

降落在航母上往往被说成是“可控坠毁”你不能像在陆地机场那样，浪费时间来调整飞机温柔降落。首要确保你能钩住阻拦索。

当你的机轮接触到甲板以后，你需要将油门推到最大动力，可以防止你错过所有阻拦索而采取“复飞”；阻拦索会钩住飞机即使飞机在最大动力的时候。

如果你愿意，然后你可以在 ATC / AI 菜单选择升起升降机，滑行到一个升降机上，降下此升降机（取消选中相应的菜单项即可）并滑行到机库。

如果你第一次没有成功不要灰心——毕竟这不是很容易掌握的动作。如果经过一些练习，你发现 Seahawk 太简单了，你可以选择 Seafire[1] 来获得更多挑战！

6.3 Atlas

Atlas 是 FlightGear 下的一个“可移动的地图”应用。用来显示飞机周边的地形、机场位置、导航点的无线电频率等信息。

关于 Atlas 的更多信息可以看其网站：
http://atlas.sourceforge.net

6.4 多显示器支持

FlightGear 支持多显示器。直接修改 XML 文件你可以配置多显示器，设

[1] Seafire “喷火”战斗机是英国 Supermarine （秀泼马林）公司在二战时研发的著名舰载战斗机。相关中文资料可见 http://www.afwing.com/intro/seafire/1.htm 。——译者注

置相对于当前主视角的从视角偏移量，这样就可以使用多显示器了。比如你可以使用一个显示器看正前方的画面，使用两个其他显示器来看两侧的画面。

有关配置多显示器的信息,可以查看 *FlightGear* 安装目录下的 README.multiscreen 文件。

6.5 多计算机支持

FlightGear 支持使用灵活的 I/O 子系统连接多个程序实例，显示完全不同的视角并由不同的计算机来控制。这可用来结合多显示器支持来创建更加复杂的环境，分立式的座舱仪表板显示器，甚至分离的控制站来设置仪表失效，修改天气等等。

关于此有个 747 座舱的实例：
http://www.flightgear.org/Projects/747-JW/

6.5.1 安装

每个 *FlightGear* 实例只支持一个显示器。因为飞行动态模型和图形的限制，*FlightGear* 对处理器的要求比较高，因此不推荐在一台机器上运行多个 *FlightGear* 实例。

因此你需要一些计算机来显示你要模拟的每一个视角，包括仪表板。这些计算机必须联网在一起，为了简单起见也最好在同一个子网里。

设计将一台计算作为主计算机。此计算机将会运行 FDM 和连接控制设备，比如游戏杆、驾驶盘和方向舵踏板。因为此机器运行 FDM，因此通常只显示简单的视图，比如主面板，以利性能最大化。

其他计算机都作为从机。这些机器主要用来显示并接收从主机发来的 FDM 信息。

6.5.2 基本配置

创建一个多显示器的基本配置是非常简单的。主机需要广播 FDM 信息和控制信息到从机。可用如下命令行选项来完成：

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

从机必须要接收这些信息，并将其自身的 FDM 关闭。

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

6.5.3 高级配置

上面列出的配置将会在两台计算机上显示相同的画面。你可能希望使用如下的这些命令行选项，来设置主机和从机：

```
--enable-game-mode （在 glut 系统上全屏显示）
--enable-full-screen （SDL 或视窗全屏）
--prop:/sim/menubar/visibility=false （隐藏菜单栏）
--prop:/sim/ai/enabled=false （禁用 AI 空中管制）
--prop:/sim/ai-traffic/enabled=false （禁用 AI 飞机）
--prop:/sim/rendering/bump-mapping=false
```

如果你想让主机只显示仪表板，可以创建一个飞行器的全屏仪表板（塞斯纳 172 已经配备），使用如下的选项：

```
--prop:/sim/rendering/draw-otw=false （只渲染仪表板））
--enable-panel
```

6.6 录制并回放

模拟器里已经提供了即时回放特性，你可以记录你的飞行以利后期分析，或者使用 I/O 系统来回放。有关如何录制特定 FDM 的技术细节可以在 $FG_ROOT/protocol/README.protocol 文件里找到。

要录制一段飞行，使用如下命令行选项：

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

此命令会以 20Hz（每秒 20 遍）的速度来记录 FDM，使用 playback 协议并将其写入到 flight.out 文件。

若要在之后回放，使用如下的命令行选项：

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

playback.xml 协议文件并不包括飞机类型、时刻等信息，你需要使用和录制时相同的命令行选项。

6.7 使用 Festival 文字转语音

FlightGear 支持文字转语音（TTS）方便 ATC 和教程消息，使用 Festival TTS 引擎（http://www.cstr.ed.ac.uk/projects/festival/）。此引擎已经在很多 Linux 发行版里包含了，也可以很容易的通过 Cygwin 安装到 Windows 系统上。写此指南时，尚不清楚是否支持其他平台。

6.7.1 安装 Festival 系统

1. 通过网站安装http://www.cstr.ed.ac.uk/projects/festival/
2. 检查 Festival 是否正常工作。Festival 提供了命令行界面。此处仅显示相关命令，注意命令里的括号。

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

3. 检查 MBROLA 是否已经安装，否则从这里下载

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

   在“Downloads”（下载）下面有一个“MBROLA binary and voices”（链接在底端，不太容易发现）。选择符合你平台的二进制下载。不幸的是，没有源代码可以下载。如果你不喜欢，你可以跳过整个 MBROLA 的安装。但是你不能使用更加真实的语音了。查看下面来找到更多语音的信息。打开 MBROLA 和 marvel 的帮助界面，主要是为了检查是否安装在正确路径并可以执行。

```
$ mbrola -h
```

6.7.2 为 FlightGear 启用语音支持

首先启用 Festival 服务器：

```
$ festival --server
```

现在启动 *FlightGear* 并加入语音支持。使用 /sim/sound/voices/enabled 属性。可以通过命令行选项：

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

当然，你可以将以上写入你的个人配置文件。这并不意味着以后必须运行 FlightGear 时搭配使用 Fesitval。如果没有搭配，你只会在终端窗口里看到一些错误信息而以。你不能在 FlightGear 运行时启用语音子系统。

为了检验一切是否正常，使用“”联系 KSFO 空中管制。首先会听到“自己的”声音（并可以在屏幕上端看到黄色的文本），接着你就能听到空中管制使用不同的语音回应（并看到浅绿色的文本）。

你可以在 preferences.xml 文件里编辑声音参数，在 $FG_ROOT/Nasal/voice.nas 文件里选择不同的文字颜色和声音。通话消息并不是直接写入到各自的 /sim/sound/voices/voice[*]/text 属性里，而是会缩写为 /sim/sound/voices/{atc,approach,ground,pilot,ai-plane}。

6.7.3 故障排除

在一些 Linux 发行版，访问 festival 是被限制的，你可能会收到类似的提示信息：

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

相关细节可以看：
http://www.cstr.ed.ac.uk/projects/festival/manual/festival_28.html#SEC130.
你可以禁用来自 localhost 和 localhost.localdomain 的访问限制，$HOME 下的.festivalrc 文件里添加如下：

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

或直接禁用所有访问列表：

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

这会允许任何人连接，对在防火墙之后的计算机应该是没问题的。

6.7.4 安装更多语音

这一节可能会有些乏味，如果你觉得默认语音还 OK 就可以直接跳过本节。首先找到 Festival 的安装目录。所有 Festival 文件都在同一个文件树下。在 UNIX 系统下可能是 /usr/local/share/festival/ 我们现在管这个目录叫 $FESTIVAL。

1. 检查哪些语音有效，可以在测试时在前面加上“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 语音和 MBROLA 安装包可以从这里下载：

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

   “don_diphone” 并不是最好的，但相对来说比较小也适合给 “AI 飞机” 使用。如果你安装了它，可以在 $FESTIVAL/voices/english/don_diphone/ 目录下找到。你也需要安装 “festlex_OALD.tar.gz”，可以在 $FESTIVAL/dicts/oald/ 目录下运行 Makefile 文件。（也许需要在 Makefile 里为 festival 加入 “–heap 10000000” 命令行参数。）

3. 较好的语音文件是 “us2_mbrola”、“us3_mbrola” 和 “en1_mbrola”。为了使用这些语音文件，你需要安装 MBROLA （见上文）和其安装包：festvox_us2.tar.gz、festvox_us3.tar.gz 和 festvox_en1.tar.gz。这些会创建这样的目录 $FESTIVAL/voices/english/us2_mbrola/。然而语音数据必须从其他网站分别下载：

4. MBROLA 语音可以从 MBROLA 的网站下载（见上文）。要想让语音文件标签为类似 “us2” 和 “us3”，解包这些文件到刚刚创建的目录 $FESTIVAL/voices/english/us2_mbrola/ 并改名为同样的 “us2” 和 “us3” 即可。

6.8 空中加油

正如名字所言，空中加油（Air-Air Refueling，AAR）让空中的飞机（一般是短途的喷气式飞机）近距离编队飞行从加油机上获取燃油。支持两种加油类型，KC135-E 加油机使用一根吊杆来连接到受油飞机上。小一点的 KA6-D 使用一根软管，受油机的加油口需要插入此软管里。

很多飞机支持空中加油，包括 T-38、闪电（Lightning）、A-4F、火神（Vulcan）、Victor（其自身也可做加油机）和 A-6E。你可以通过 AI/ATC 菜单查看飞行器是不是支持空中加油，如果 “Tanker”（加油机）菜单项可用，那么飞机就支持空中加油。

6.8.1 安装

要想设置空中加油，只需启动 FlightGear 时选择支持空中加油的飞机即可，起飞并爬升到 15000 英尺。在此高度巡航，选择 AI/ATC->Tanker 并选择 “Request”，会在和你相近高度的开放空域里分配一架新的加油机。

FlightGear 将会通报给你加油机的高度、速度和 TACAN 识别号。你需要在 TACAN 系统里输入加油机的 TACAN 识别号（可以从 Equipment->Radio Settings 对话框，或你的驾驶舱控制系统里）。取决于你的飞机类型，加油机也许会出现在你的雷达上。如果你想获得更多帮助来找到加油机，你可以选择 “Get Position”，让加油机报告相对于你的位置。

调整到合适的航向，由 TACAN 航道指引（你需要正对以便更加接近加油机）并在雷达或导航屏幕里寻找。距离 5 海里时，你需要减速到比加油机快 20 节的速度——所谓“慢速超过”。KC135 在 10 海里就可以发现，更小一些的 KA6-D 在 1 海里外可以看见。如果你发现自己可能会错过，放出减速板。

接近到 50 英尺左右的位置（不要距离过近，你可能会撞到加油机），驾驶舱上应该会显示正在加油。在 A4 仪表板的油量标尺上上会是一个绿色灯亮起，并会看到油量在增加。

当油箱加满，或加到你想要的油量，稍稍收油门，从加油机后撤离并继续你的飞行。

真实情况下，成功的空中加油并不简单，需要大量的练习。这里有一些提示：

1. 缓慢接近加油机。你会发现很容易就会超过加油机，进而发现加油机已经不知所踪。

2. 如果你发现自己不能很容易的靠近加油机，因为油门太敏感，可以尝试放出减速板。减速板会要求发动机输出更多动力以便维持速度，这样会降低油门的敏感性。

3. 为了减轻工作量，你也许会使用自动驾驶来修正高度和速度。虽然 NASA 最近演示了高级的自动驾驶系统可以完成空中加油而不需要飞行员干预，然而这不过是技术性作弊。

4. 还要记住随着加油的进行，飞机会变得越来越重，而重心也会移动，因此会影响飞机的配平。

5. 加油机会在空中沿顺时针方向飞行一个跑道形状的航线。虽然在转弯时能够保持连接，不过你也许会发现，等待加油机结束转弯准备进入平飞时更容易进入加油。加油机会在将要转弯时发出警告。

6.8.2 多人联机加油

空中加油可在一个多人联机会话中进行，使用 KC135 或 Victor。驾驶加油机的飞行员需要使用“MOBIL1”、“MOBIL2”或“MOBIL3”这样的呼号。其他数字也可以，但必须是只有这三个是分配空对空 TACAN 系统频道的，分别是 060X、061X 和 062X。

如果受油机的飞行动力学模型是 YASim，就没有其他的问题了，如果是 JSBSim，用户必须保证设置中没有开启 AI 加油机。这意味着在 aircraft-set.xml 和 preferences.xml 配置文件中取消（或注释掉）相应的加油场景（senario）。

多人联机的空中加油与 AI 加油是一样的，但有更多挑战。整个过程中一定要保持网络顺畅；多人游戏代码里包含了当网络出现短暂断联时，预测飞机的飞行轨迹，而恰恰这会让近距离编队变得很困难甚至无法进行。

第三部分

飞行教程

第七章
教程

如果你是个飞行新手，像 *FlightGear* 这样的高级模拟器会让你感到畏惧：因为要在没有学会如何飞行的时候就跳入飞机的驾驶舱。

在真实生活中，学习飞行时都会有一个飞行教员坐在你旁边教你如何飞行，并保证安全。

然而我们不可能为每一个虚拟飞行员都提供私人飞行教员，不过却有很多教程可以帮助你成为一名精通的虚拟飞行员。

7.1 飞行中教程

FlightGear 包含了一个飞行中教程系统，模拟教授一套虚拟的“课程”。这套教程从如何启动发动机到教会你第一次飞行。要访问这些教程，选择 Help 菜单里的 Start Tutorial。

教程系统可以与 Festival TTS 系统一起工作（见上文）。

为简便起见，使用教程时建议将 AI 飞机关闭，可以从 AI/ATC 菜单找到。否则空中管制的信息会与教程信息混淆在一起，让你无法听清。

每一个教程都包括了大量你必须完成的步骤。你的飞行教员会告诉你如何完成每个步骤，并观察你的完成情况，需要时提供额外的指导。

在教程下，要求教员复述指令按“+”键。你可以在任何时候按“p”键暂停教程。要停止教程，可以在 Help 菜单选择 Stop Tutorial。

7.1.1 塞斯纳 172P 的教程

如果这是你第一次飞行，这里有一些为塞斯纳 172P 设计的教程，协助你学习基本飞行知识，就如同在真实的飞行学校一样。这些教程基于旧金山附近的半月湾机场（Half-Moon Bay，KHAF）和利弗莫尔市立机场（Livermore Municipal，KLVK）。两座机场都在基础软件包里提供了。要启动教程，选择塞斯纳 172P 飞机，并选择起飞机场为 KHAF 或 KLVK，使用向导或使用命令来启动 FlightGear：

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

当模拟器载入完毕，从 Help 菜单选择 Start Tutorial。你会发现有一系列可用的教程。选择其中之一并按 Next。一个教程的简介将会显示出来，按 Start 来启动此教程。

7.2 FlightGear 教程

下面的几章将会提供 *FlightGear* 相关的教程，可以让刚刚新手学会第一次飞行，并依靠自己的导航翱翔云端。如果你从来没有飞过小型飞机，下面的教程会教你如何飞行。

除了此指南，还有一个由 David Megginson （*FlightGear* 的主要开发者之一）写的非常棒的教程，教你如何用 *FlightGear* 飞一个机场起落航线（五边航线）。此教程有很多截图和大量资料，可在此处找到：

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

7.3 其他教程

还有大量非 *FlightGear* 相关的教程，其中有很多都适用于 *FlightGear*。这其中比较全面的则是由 FAA[1] 发布的航空信息手册（Aeronautical Information Manual，AIM）。可以在此下载：

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

这是由 FAA 官方发布的介绍基本飞行信息和空中管制流程的指南。里面包括了大量的飞行规则方面的信息，飞行安全，导航等等。如果你觉得这有些难度，可以看看 FAA 训练手册，

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

里面包含所有和飞行相关的信息，从飞行原理和飞机构成讲起，再到起飞降落和紧急情况处理。这非常适用于那些希望学习基本飞行知识，但又不想购买昂贵的纸质飞行员手册的人。

上面说的手册对 VFR（Visual Flight Rule，目视飞行规则）是个非常好的指导，然而并不包括 IFR（Instrument Flight Rule，仪表飞行规则）的相关介绍。不过一个由 Charles Wood 写的仪表飞行规则下的导航教程，可以在此找到：

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

[1] 美国联邦航空管理局，Federal Aviation Administration，缩写为 FAA。是美国运输部下属、负责民用航空管理的机构。——摘自中文维基百科，译者注

另一本全面但却很容易阅读的教程是 John Denker's 写的《飞机如何飞行》,

http://www.av8n.com/how/.

这是一本完全在线阅读的教程，从伯努力原理讲起，升力和动力等等，之后的章节则会讲到 VFR 和 IFR 飞行。

第八章
基础飞行模拟教程

8.1 序言

航空业充满了各种极端：

- 飞机非常脆弱却以高速飞行。然而却是最安全的交通工具。
- 飞行员需要紧跟各种规则和程序。然而飞机却是自由的象征。
- 只需要很少的一些训练，飞小型飞机是很容易的。然而一旦发生问题，需要在数秒内解决。
- 很多飞行教程都以幽默的口吻来写，然而若飞行时不够严肃，会让你快速坠地。

此教程使用的飞行器是塞斯纳 172P[1] 型。此机型已被用于很多真实飞行学校中，并且是一款非常好飞的机型。

下面的这些文章可以作为此教程的补充，并回答了很多你阅读中的一些问题。特别是第一篇介绍了飞机的主要部件和控制：

[1] 塞斯纳 172 对中国人来说并不陌生，在上世纪八十年代风靡全国的日本电影《追捕》中，杜秋冬人（高仓健饰）就是驾驶塞斯纳 172R 飞机从北海道飞回本州的。——译者注

- https://www.gleim.com/aviation/learn-to-fly/
- http://www.pilotfriend.com/training/flight_training/aero/principa.htm
- 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://flightsims.ig-wilson.com/index.php?f16land
- http://www.navfltsm.addr.com/

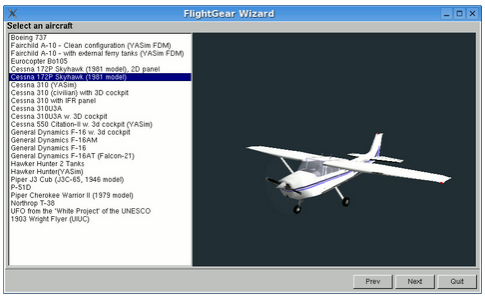
此教程基于我本人的知识，然而不可避免的可能会有一些错误。对此我非常抱歉。

8.2 启动

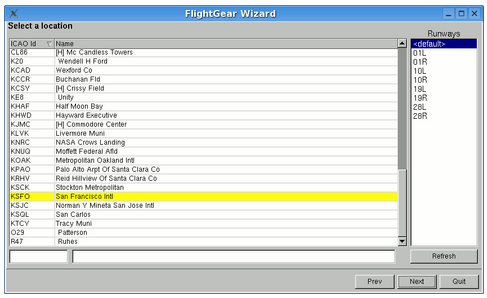
根据你的平台和发行版的不同，有很多种方式来启动 *FlightGear*。

8.2.1 MS Windows

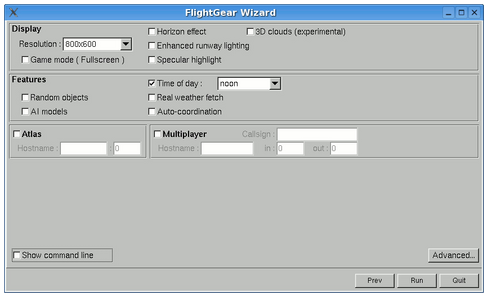
在微软 Windows 系统中，*FlightGear* 有一个图形化的向导，让你可以选择飞行器和起始位置。首先如下图所示选择 Cessna 172p 飞机。为了能够适应此教程，不要选择 2D 仪表板的版本（你也许会发现 2D 仪表板更适宜教学）。点击**下一步**按钮来选择机场。



你可以从任何机场启动教程，我假设你使用了 FlightGear 默认的旧金山国际机场（KSFO）：



选择好了 KSFO 之后就点**下一步**按钮，对模拟器来说你可以从任何时间点起飞。不过对于你的第一次飞行，我建议还是从午间起飞。另外建议你使用小一点的分辨率 800 × 600。之后你可以再使用更高的分辨率，然而这对性能会有较大影响。按**运行**按钮，则会应用你的选项来启动 *FlightGear*。



如果在你的 Windows 系统上运行最新版的 *FlightGear* 有问题的话，可以尝试使用旧版本（比如 0.9.8）这样可以减低对显卡的需求。你可从 *FlightGear* 下载页面找到 FTP 镜像来下载 。

如果你在 Windows Me 系统时飞行模拟器突然变得卡卡的，帧速率快速降低。可以尝试关掉除 Explorer 和 Systray 外的所有进程，之后再运行 *FlightGear*。如果你关掉的进程有杀毒软件，这会带来一定的安全风险。在一台 Windows Me 的机器上，*FlightGear* 使用 800 × 600 分辨率可以有较好效果，稍低一些的 640 × 480 会触发更低的帧速率。

8.2.2 Linux 和其他 UNIX 系统

在 Linux 和其他类 UNIX 系统上，也许你必须从命令行启动 *FlightGear*。如果你已经安装了 *FlightGear* 但在菜单里找不到的话，可以尝试：

- 在命令行窗口下输入命令 `fgrun`。如果安装的话，会启动一个和 Windows 下一样的图形化向导。
- 除此之外,还可以打开命令行窗口输入如下命令:`fgfs --timeofday=noon --geometry=800x600`.

8.2.3 黑夜里?

如果没有命令行选项 `--timeofday=noon`，*FlightGear* 会按旧金山当前时间（对欧洲来说通常是晚间）来启动。要改变模拟器的时刻，只需要在 **Environment->Time Settings** 菜单里选择 **Noon** 即可。

如果从菜单启动 *FlightGear*（比如 KDE 或 GNOME），你能编辑 FlightGear 的启动图标属性，并修改 `fgfs` 命令到类似 `fgfs --timeofday=noon --geometry=1024x768`，或者加入任何你想加入的命令行选项。有关命令行选项相关内容可以参考第 四 章起飞：如何启动程序。

8.3 第一个挑战——直线平飞

当 *FlightGear* 启动以后，你会看到如下的窗口并听到发动机的声音：

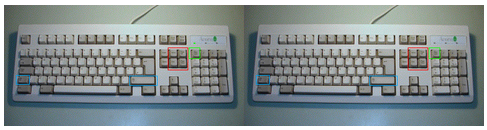
启动时，飞机停在跑道的起点且发动机维持低功率运转。飞机也许会轻微抖动，但不会移动。

关于键盘控制

- 在本教程中，小写字母表示你只需要按这个键即可，大写意味着你需要同时按 ⇑ **`Shift`** 键和这个键。也就是说，如果你看到“v”键表示

你只需要按字母 **v** 即可，如果你看到 “V” 则意味着你需要按下 **Shift** 键的同时按字母 **v**（简单来说，V 就代表 **Shift-v**）。

- 本教程假设你的 **NumLock** 是已经打开的状态。键盘右上角的绿灯亮起。如果没亮，按NumLock键直到绿灯亮起。



按 **v** 键来从外面查看飞行器。重复按 v 键可以循环各种视图模式，直到你回到驾驶舱中。按 **V** 将会反向循环。

在真实中，我们会围着飞机转一圈检查所有都正常，没有任何东西阻碍移动控制面，也没任何堵住仪器的开口处。而在模拟器里，这些都已经在启动前帮我们做好了。

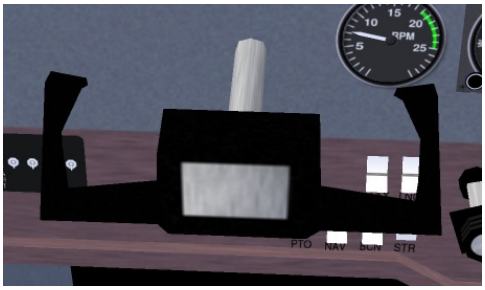
按住 **Page Up** 大概八秒，你能听到发动机的声音越来越大。

飞机开始在跑道上加速。在离地之前，飞机会被拉向左侧，并向左侧倾斜，并坠毁在地面上（也许）。

你可以通过 **View -> Instant Replay** 来检查整个坠毁过程的回放。按对话框底部的 **Replay** 按钮，下面这张图就是坠毁时的状态。你按 **F3** 可以截图保存。你可以用 **F10** 来切换菜单是不是显示。

之后，你可以用 **File->Quit** 退出 *FlightGear* 并用与之前同样的选项重新进入模拟器。

为了能够直线飞行，你需要控制飞机的驾驶盘:



你可以用游戏杆或者鼠标来控制。要使用鼠标你需要将鼠标置于控制模式，可以通过按 **Tab** 键直到指针变成 + 形。移动鼠标就可以看到驾驶盘也在随之移动。按 **v** 键可以从外面观察飞机。此时继续移动鼠标，可以看到机尾的升降舵和机翼后部的副翼也在移动。如果你的视点离飞机太远，看不清楚的话，可以按几次 **x** 键来放大，按 **X** 来缩小。**Ctrl-x** 可以恢复默认的缩放级别。此时按 **v** 回到驾驶舱里。

按 **Tab** 几次直到鼠标进入观察模式。在此模式下，鼠标指针会变成 ↔ 形。这可以让你更容易的移动鼠标环顾四周。按鼠标左键将会重新回到中间。你可以在控制模式临时进入观察模式，只需要按住鼠标右键并移动鼠标。最终再按 **Tab** 会回到正常鼠标模式。

简单来说，**Tab** 让鼠标在三个模式之间循环:

- 正常模式。此模式让你可以点击菜单或点击仪表板。
- 控制模式。此时鼠标控制驾驶盘（指针是 + 形）。
- 查看模式。此时鼠标控制观察的角度（指针是 ↔ 形）。

现在尝试在起飞，按 **Tab** 进入控制模式。按住 **Page Up** 键将发动机油门推到最大功率，不要强制通过鼠标让其在跑道上保持直线，就让它这样向左滑行直到自己升入空中。然后用鼠标尝试让飞机保持直线平飞。（如果你想控制在地面的飞机，请查看 8.5 小节）。

你会发现你需要防止飞机向左滚转:

……或者向右滚转：

……或者向地面扎去：

尽可能的沿直线飞行，让地平线稍稍高于机头：

无论你在电脑游戏或模拟游戏方面的能力如何，第一次飞行很可能不会成功。飞机可能会坠毁，也许只在起飞后不久。这也就是为什么很多人对真实性比较高的飞行模拟感到绝望。只需要多多尝试，也许你会找到一种很舒适的控制方式。

一个普遍的错误是向前移动鼠标，以使机头向上昂起。实际上，你需要向后移动鼠标这样才能让驾驶盘向后拉杆。

同样的，当你想要降低飞机的机头，你需要向前移动鼠标。也许这有些奇怪，不过所有飞机都是按照这种方式来控制的。随着时间的推移，你会明白这种方式的好处。你也许会发现鼠标的小幅移动会对飞机有很大的影响，所以也许降低鼠标的敏感度会有一些帮助。

如果你需要更形象的比喻，这么说也许会一些帮助：假设在你桌子上有一个美式橄榄球，而你的手必须“粘在”球的顶端。如果你要向前移动，则球就会向前滚动，而你的手指则会碰到桌面。如果你向后移动手指，则球会向后滚动，而你的手指就会指向天花板。你的手就是飞机：

另一个普遍的错误是假设驾驶盘的动作与飞机的滚转坡度一致。也就是说，你认为只要驾驶盘水平，飞机也会平飞。事实并非如此，驾驶盘只是控制飞机滚转的速率。如果飞机有向左 20° 的坡度，而驾驶盘是水平的，则意味着飞机会保持向左 20° 的坡度直到外力改变这个状况。如果想要让飞机恢复平飞，需要温柔的向右转动驾驶盘（温柔的向右移动鼠标），并稍稍维持

向右一段时间。飞机会向右慢慢滚转。当与地平线平齐时，再将驾驶盘慢慢转平。飞机会保持平飞（直到外力来改变其方位）。

还有一个错误是尝试找到驾驶盘/鼠标的“正确位置”。很自然的，你想找到一个完美的位置，可以让飞机保持直线平飞。事实上没有这种理想的驾驶盘位置。飞机在空中的运动本质上是不稳定的，你需要不断的用鼠标做小幅移动，来修正飞机的姿态并保持直飞。也许刚开始会让你耗费全部注意力。这就如同开车，让飞机保持直线平飞将很快会变成一种本能。对长期的飞行来说，你还可以偶尔使用自动驾驶仪来保持平飞，不过本教程将不会涉及。

为了帮助你找到控制的感觉，眼睛始终关注着窗外景物，而不要叮着驾驶盘或仪表板。检查地平线与机头之间的位置，地平线和你飞机的发动机盖子是你的主要仪表。只是每隔一段时间扫一眼仪表板而已。

在鼠标处于控制模式时，千万不要将鼠标移出 FlightGear 窗口边缘以外。一旦鼠标离开窗口，就会停止控制飞机，很可能会造成极糟糕的动作！如果你想在窗口外使用鼠标，首先按 **Tab** 键两次切换到正常模式。

你可以使用键盘上的**方向键**来控制驾驶盘，或者使用小键盘上的 **8**、**2**、**4** 和 **6** 键。也许最开始这种方式会比鼠标控制来的简单一些，飞行时的微调控制却不如鼠标，因此持续练习鼠标控制才是王道。

也许你在机场附近飞行时，会听到蜂鸣器的声音，这是降落辅助信号。现在不用担心这些声音。

你会想知道在掌握了这些以会如何爬升。下一步要学习如何维持高度，或者在你的控制下缓慢上升和下降。

让飞机保持一个高度需要观察高度表，并通过前后小幅移动鼠标来让飞机停止上升或下降。

高度表位于仪表板的中上部，长表针表示百英尺，短表针表示千英尺。下面这张图里的高度表示意的是 300 英尺，约 100 米[1]。

上升或下降时高度表会作出反映，表针逆时针转表示下降，顺时针表示高度增加。如果你看到高度表“狂转”，说明你在丢高度，缓慢向后移动鼠标拉起机头即可。

[1] 1 英尺 = 12 英寸 = 0.3048 米。口算时可以简单将英尺数除以 3，即得公制米数。——译者注

一段时间以后你会发现，当平飞的时候，机头与地平线的相对位置永远是一样的。这就是平飞时的飞机姿态。让机头处在这个位置，你就几乎可以让飞机进入平飞而不需要参考任何仪表。现在你可以微调高度了。

谨记：高度表不会自动显示距海平面的绝对高度。你需要调整其与当地的气压相匹配。高度表左下角的黑色旋钮让你可以调整高度表拨正值。启动 *FlightGear* 以后一直待在地面上，点击（鼠标在正常模式）这个黑色的旋钮，点击左侧可以让旋钮数值变小，点击右侧则会变大。使用这个小旋钮调整到当前高度。前提是你能获得当前真实的高度。例如你在 1100 英尺，那么就调整旋钮，让高度表显示 1100 英尺。使用鼠标中键点击可以让旋钮旋转更快，或者你可以用鼠标滚轮。**`Ctrl-c`** 可以将可点击的区域高亮化。

为了更简便的设置高度表，机场会将其修正海平面气压通过各种方式告知出来。也许会通过无线电服务（美国称之为 ATIS[1]）广播当前的修正海平面气压。单位是英尺汞柱（inHg）。高度表上有一套齿轮组，高度表会利用拨正值来计算高度，你可以通过旋钮设置高度表拨正值[2]。另外，如果你在地面上已经知道当前机场的标高（相对海平面的高度），你也可以直接调节高度表直到其显示正确的高度。

注意，知道“距地面高度”和“距海平面高度”这两个概念的差异是非常重要的。如果你以距平均海平面高度（Above Mean Sea Level，AMSL）24000 英尺（约 7315 米），飞行在珠穆朗玛峰附近，你距地面（Above Ground Level，AGL）的垂直距离会小很多。因此了解自己距离周边地面的垂直距离，显然很有帮助[3]。

---

[1] ATIS 是 Automatic Terminal Information Service 的缩写，意为”自动终端情报服务“，中国大陆称为“情报通播”。机场在一个单独的无线电频率上广播，包括主要的与飞行相关的信息，如天气、可用跑道、气压及高度表拨正值等信息。——译者注，摘自中文维基百科

[2] 除美洲外，修正海平面气压的单位在欧洲和亚洲一般使用的是百帕（hPa）为单位。标准大气压是 1013.25 百帕，也就是 29.92 英尺汞柱。1 inHg = 33.86 hPa。设置高度表拨正值时，一般忽略小数点。这是高度表拨正值换算速查表：http://www.pcwp.com/mb*conversion.html*。——译者注

[3] 为了分清“距离地面”还是“距离海平面”，常会用“高”（Height）和“高度”（Altitude）来区别。本手册的翻译也遵循此规律。——译者注

8.4 基本转弯

当你可以或多或少的学会直线平飞以后，让我们来开始学习转弯。方法非常简单：

- 当飞机向左横滚时，就会向左转弯
- 当飞机向右横滚时，就会向右转弯

如果想很好的转弯，不需要很大的坡度，20° 就已经足够并可以安全而稳定的转弯了。转弯侧滑仪表示从后面看你飞机滚转时的角度。上图展示了向右 20° 坡度转弯侧滑仪的状态。你也可以通过观察地平线来确认角度。

尝试这么做：让你的飞机保持 20° 坡度几分钟，看看飞机外面的景物不断在眼前重复。每隔 120 秒，这些景物就会重复一遍，这就意味着你做一次 360° 转弯需要 120 秒（做一次 180° 转弯则需要 60 秒）。这个时间在导航时非常有用。无论飞机以何种速度飞行，在 20° 坡度时，塞斯纳 172P 需要 60 秒完成一次 180° 转弯。

尝试向左和向右转弯，注意保持机头与地平线之间的距离，与之前平飞时一样。

转弯侧滑仪下部的紫色小球表示侧滑时的力度。在真实飞行中，你能在转弯时自己感受到，因为在模拟器里很难模拟出来，所以你需要经常注意此小球。如果你转弯时动作很平滑（稍稍用一下方向舵），小球会一直保持在中间。如果小球向右侧移动，表示飞行员也会在座舱内向右移动，就如同汽车左转时人向右运动。平滑转弯时，即便是很猛的转弯，乘客也不需要忍受侧滑力。他们只是被离心力稍稍带离座椅。

稍稍试验以后你就会发现坡度越陡，转弯角度越大，也越需要向后拉杆[1]。坡度超过 60° 的转弯出现在特技飞行和军机中，对塞斯纳这样的飞机来说是很危险的。

[1]这是因为升力的侧向分量参与转弯产生向心力了，此时在垂直方向上的升力自然就少了，因此高度会降低。此时可以通过稍稍带杆的方式维持恒定高度，但会造成速度稍稍降低。——译者注

8.5 地面滑行

默认情况下，启动 *FlightGear* 会让你正对跑道并准备出发，你也许会想飞机如何从机库，经过滑行道进入跑道的。这就是滑行。

下图是 转速表。表示发动机运转时的速度，单位是每分钟百转数（RPM）。

按 **Page Up** 键几次，直到转速表显示大概 1000 RPM 左右（如上图）。可以用 **Page Down** 降低发动机速度。

在大概 1000 RPM 时，飞机会在跑道上移动，但却不会加速到起飞速度。

按 “**.**” 键飞机将会快速向右转弯。如果你一直按着 “**.**” 键飞机会刹停。当你按 “**.**” 键时会触发飞机右轮刹车。

要触发左轮刹车，可以按 “**,**” 键。

“**,**” 和 “**.**” 键，模拟了真实飞机上的刹车踏板。使用油门和刹车踏板，你可以控制飞机地面的速度。

在跑道慢速滑行时左右轮的刹车是非常有用的。你也可以转动飞机前轮。在真实中，飞机前轮是通过用脚蹬踏板实现的。用脚蹬相应方向的脚踏板就会向那个方向转弯[1]。如果没有真实的脚踏板，可以用两种方式来虚拟:

- 使用小键盘上的 “**0**” 键和 **Enter** 键。如果按小键盘上的 **Enter** 键几次，你会看到飞机向右转并一直右转。按 “**0**” 键几次，来让飞机恢复直线。

- 使用鼠标。当鼠标是控制模式时（+ 形的鼠标指针），如果你按住鼠标左键并移动鼠标，此时会控制脚踏板而不是驾驶盘。方向舵脚踏板同时连接着方向舵和前轮。这样控制起来更加精确。

启动模拟器，按 **v** 或 **V** 键从外部来看飞机，按 **x** 几次来放大飞机。观察前轮并按小键盘上的 “**0**” 键。然后再按 **Enter** 键。你可以看到前轮的转动。按 **Tab** 键，让鼠标进入控制模式。保持鼠标左键按下，进入方向舵控制模式，移动鼠标向左向右。注意此时方向舵，也就是垂直尾翼上的移动面，会与前轮同时移动。

[1]飞机上的脚踏板又称为 “脚蹬”，它同时控制方向舵（包括前轮）和左右轮刹车，不过却用不同的方式。向前方蹬左侧或右侧的脚蹬会控制方向舵偏转，同时带动前轮向左或右转弯。如果踩下左侧或右侧（当然可以同时踩）的踏板，会触发相应的左轮或右轮刹车。这些操作在 FlightGear 里面都有模拟实现，读者可留心观察 “脚下” 的动作。——译者注

我倾向于在地面上用鼠标控制前轮，在空中用小键盘的 **0** 键和 **Enter** 键来控制方向舵。也就是说：在地面上我可以用鼠标更加精确的控制前轮来转弯，而当前轮离地以后，我就会放开鼠标左键。

8.5.1 空速

就像开车一样，知道你运动的速度肯定有帮助。航空上用空速表（ASI）来测量速度，以海里每小时为单位（节）。

1 节（knots）等于 1.852 千米/小时。因此粗略来说，如果你想换算成千米每小时（km/h）的话，只需将节数乘 2 即可。1 节等于 1.15115 英里/小时（mph），因此非常粗略的，1 节就等于 1 mph。注意很多飞机的空速表（比如 Piper J3 Cub）就是用 mph 而不是节做单位来显示空速[1]。

空速表表示飞机与周围空气的相对速度，而不是像汽车那样是相对地面的速度。如果飞机停在地面，而有 10 节的风正对飞机，那么空速表将会显示 10 节的空速，虽然此过程飞机相对地面纹丝未动[2]。

当飞机在跑道上运动超过 40 节时，你必须避免让前轮接触地面。前轮并没有设计运行在高速状态，在真实状况下前轮会摆动甚至折断。

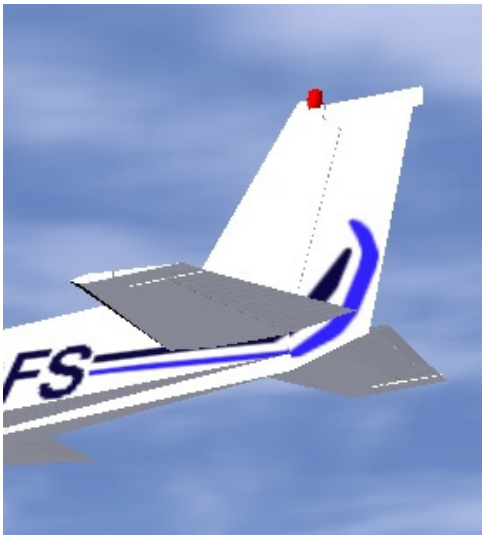
[1]稍微精确的心算方法：将节数（海里数）乘以 2 再减去 10%，例如 22 节，先乘以 2 得 44，再减去 10%，即 44 减 4 得 40 公里，已经很接近精确值 40.31 公里。——译者注，摘自《飞行快速心算》一书，中国民航出版社

[2]空速表指示的空速一般称为“指示空速”（Indicated Airspeed，IAS），而这一段例子里的“10 节”空速其实应为真空速（True Airspeed，TAS），在海平面高度且标准大气压下，指示空速与真空速相等。然而平时为了简便起见，我们还是使用指示空速来操纵飞行。同时飞机相对地面的速度称为“地速”（Ground Speed，GS）。——译者注

起飞时，超过 40 节你可以轻柔的拉杆，让前轮离开地面。不要在地面上高速转弯，可能会导致飞机翻覆。

下图展示了前轮轻柔离开地面。不要过度离开地面，保持飞机前盖低于地平线。你只需轻柔抬起前轮。

问题：如果前轮不再接触跑道，如何转弯？回答：依旧使用方向舵踏板。正如前文所述，方向舵踏板同时连接了前轮和尾部方向舵：



当空速超过 40 节时，方向舵上有足够的空气流过，这样就可以来控制飞机。

注意前轮和方向舵让飞机转弯时的曲率是不同的。所以当用方向舵来控制飞机时，你需要适应方向舵的角度控制。这意味着你需要更快的按小键盘上的 **0** 键和 **Enter** 键（或者按住鼠标左键用鼠标紧密的控制方向舵）。

当你更加熟悉前轮和方向舵的控制以后，你可以用这些来控制飞机在跑道上直线滑行起飞。

当飞机向右太多时，你按 **0** 键数次来纠正。不要等飞机完全对准中心线再转弯，当飞机快要到你想要的方向时，按几次 **Enter** 键。如果你用鼠标，修正操作就会变得更容易也更精确。

一言以蔽之，有两种方法控制飞机地面的运动：左右轮的刹车以及方向舵踏板。这种控制冗余在航空业非常普遍。如果一种方法失效，你还有另一种来备份。

你也许会想，为什么飞机在地面会向左倾斜，导致你必须在方向舵上施加向右的力？主要原因是因为流过螺旋桨的空气。这些空气流过机身并因为机身作用形成涡流，涡旋气流的上部会“推”着垂直尾翼向右侧偏，进而导致飞机前部向左侧偏航[1]。

你可以用小键盘上的 **5** 键将驾驶盘和方向舵置中。飞行前这是个很好的预防措施。有时在飞行中此操作还能“救命”。

8.6 高级转弯

就如同在地面一样，空中也有两种转弯方式。你可以如上文所述使用副翼（转动驾驶盘/鼠标）或者使用尾部的方向舵（蹬方向舵踏板/小键盘 **0** 键和 **Enter** 键）

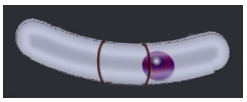
为什么有两种方式？既互为冗余，又互为补充。方向舵的主要作用是偏航（绕垂直轴线旋转），副翼的作用是滚转（绕飞机纵轴旋转）。

- 当飞行离地面很近的时候，最好不要通过滚转来转弯，更多的时候会使用方向舵。方向舵可以让你转弯而飞机不带任何坡度。
- 当飞机离跑道很近时，两侧机轮需要保持距离跑道同样高度，这表示机翼必须水平。飞机不允许有坡度。通过副翼来保持飞机机翼水平。注意不需要特别完美，有一点点小坡度是无害的。
- 在飞行中，特别是高速时，方向舵转弯是非常低效的：
  - 它可以使飞机增加侧翼的气流，进而增加阻力
  - 飞机转弯很慢
  - 高速时离心力会非常强甚至有危险

  因此使用副翼可以让你更有效率、快速、可靠且舒适的转弯。
- 方向舵在机翼失速时很重要。因为，在失速时机翼副翼变得低效或者无法使用（注意有些飞机会在低速时，因为过度使用方向舵而进入非常危险的失速）。

飞行中转弯使用副翼，你依旧需要用一点点方向舵。稍加方向舵可以用来抵消副翼转弯时产生的反向偏航。在真实飞行中，你可以感受到这股侧滑力。在模拟器里你需要关注转弯侧滑仪。下图展示在猛的使用副翼右转时，小球被甩向了右侧。这意味着飞行员也承受着相同向右的侧滑力。你可以通过稍稍向右蹬方向舵来抵消这个力（按 **Enter** 键数次）。在正常的飞行中，你需要用方向舵保持小球在中间。

[1]此现象有个专业名词叫“螺旋桨滑流”。——译者注



因此在正常飞行中，使用副翼来转弯。当接近地面低速时使用方向舵。一种方式不能完全取代另一种。你依旧需要在高速高空使用方向舵。相反的，你也可以在低空时用副翼来保持机翼水平。

地面滑行时，你甚至可以用副翼。强风很可能会把飞机吹翻，为了抵消这个，你需要向风的方向转动副翼。这样来风的一侧升起副翼，可以帮助机翼向下。

你需要避免快速而粗暴的移动方向舵。在地面，这会造成飞机快速旋转。在空中低速时会造成非常危险的一种失速，高速时会造成空气动力学部件和物理损坏。所以对待方向舵要温柔。

我建议你在飞行时练习使用方向舵。在低速，比如 70 节时，通过增加和减少发动机功率来保持高度。用方向舵来转弯或保持航向，然后再将飞机转向新的航向。观察飞机如何偏航，并学习预判方向舵的控制。不要尝试陡峭的转弯，使用副翼来保持机翼水平。

8.7 这些是什么?

这一章将会讲解飞机的仪表、开关和飞机的控制。虽然在模拟飞行里，你只需要通过油门和驾驶盘就可以控制飞机的基本操作，然而你需要了解所有控制方式以便更安全和高效。

8.7.1 发动机控制

飞机的发动机被设计成简单、可靠和高效的。与现代汽车使用高级电力点火和燃油喷射系统不同，飞机使用老式技术而不依赖电能。因此当飞机电力完全失效的时候，飞机依旧可以飞行。

磁电机

在仪表板左下角你可以找到磁电机的的开关和发动机点火开关：

要看清此开关，可以按 **p** 键查看 2D 概要仪表板，或者按 **Shift-x** 来缩小（用 **x** 键或者 **Ctrl-x** 恢复缩放）。

也许你知道汽车使用电动火花塞来引燃汽油。现代汽车引擎使用电力点火装置。飞机发动机则使用更加老式（但更可靠）的磁电机点火开关。为了冗余设计，包括左和右两套磁电机。当你将磁电机转换到 OFF 位时，两侧的磁电机都会关断，发动机也会停止。将磁电机开关转到 L 位则会使用左侧的磁电机，转到 R 位则使用右侧磁电机，在 BOTH 位则同时使用两侧的磁电机。在飞行中我们使用 BOTH 位。

既然在飞行中我们使用双侧磁电机，为何还要有切换开关? 原因是在飞行前我们需要检查所有磁电机都工作正常。要检查磁电机可以将发动机转速设置在 1500 RPM 并切换磁电机到 L 并观察转速表。你能观察到转速有小幅下降。如果发动机被切断，左侧磁电机故障。如果你没有看到转速下降说明切换开关有问题，因为双侧磁电机都在工作。你可以用同样的方式来检查右侧磁电机。当然在模拟飞行里，磁电机是不会故障的!

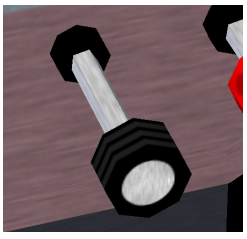
在飞行中如果一个磁电机故障，那么另一个会保持发动机运转。单侧磁电机失效是很罕见的，双侧都失效几乎没有听说过。

你可以按 **{** 键来关闭发动机。要起动发动机可以按 **}** 键三次以便让磁电机切换到 BOTH。然后按 **s** 键使用起动电机，按住几秒钟，直到发动机起动。

你可以通过正常模式的鼠标转动磁电机开关。按 **Ctrl-c** 来查看以黄色矩形框高亮出来的可操作区域。

如果你将磁电机开关置为 OFF 位，发动机噪音就会停止。此时如果快速切换到 L 位，发动机又会起动螺旋桨依旧旋转。如果你一直等到螺旋桨停止，再切换到 L、R 甚至是 BOTH 位，都不会起动发动机。当发动机停止时，永远要将磁电机开关置为 OFF 位。

油门



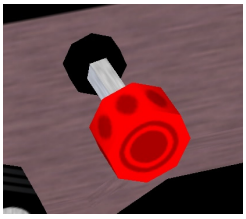
你已经知道通过推油门杆（按 **Page Up** 键）可以增大发动机动力，拉出油门杆（按 **Page Down** 键）可以减少发动机动力。在油门杆上按鼠标左键和右键（鼠标中键可以快速移动）也有相同的效果。

“增大动力”到底是什么意思? 是不是意味着将输入发动机的油量增加? 是的，但这并不能完全解释。你需要知道飞机发动机也需要大量空气。发动

机的汽缸里燃烧的是油气混合物质。只有燃油是不能燃烧的。必须是一定比例混合的燃油和气体，才能爆开和移动发动机的活塞。所以当你推油门杆的时候，同时增加了燃油和空气到发动机里。

油气混合

一定的空气遇上一定的燃油是非常危险的。必须时刻密切关注这两者的比例。这也正是油气混合杆的作用。下图展示了油气混合比杆，已经拉出很多。



当混合比杆完全推进的时候，进入发动机的是大量燃油和少量空气。这就是所谓的“富油”混合。当油气混合比杆被完全拉出的时候，将会有过量的空气进入发动机，也就是“贫油”混合。正确的混合比则是在这两种极端之间，多数情况会比较接近完全推入状态。

当你起动发动机和起飞的时候，你需要富油混合。这意味着混合比杆需要完全推入。富油混合可以让发动机更容易起动，也会让发动机多了一些可靠性。缺点是一些油并不会完全燃烧，只会造成浪费并被排出。这会让发动机变得很脏，同时降低了发动机的输出功率，并因为残积物堵住汽缸而慢慢劣化发动机。

在正常飞行中，你需要将油气混合杆稍向后拉，以获得最佳油气混合比。在模拟器里可以这么做。启动模拟器以后，设置停留刹车，按 B 键（**Shift-b**）。将油门推到最大，发动机转速此时会接近最大转速。慢慢拉油气混合比杆（使用正常模式下的鼠标）。你会看到转速有小幅升，你获得了更多的动力，而没有增加燃油量。你没有浪费燃油也减少了污染。如果你继续拉油气混合比杆，转速又会开始下降，因为现在空气太多了。过量的空气减慢汽缸里的膨胀速度，并降低了膨胀温度，因此热度下降。你必须优化混合比。因为热力学的原因，最佳混合比不会恰好产生最大动力——至少好过发动机只比最大动力轻微富油或者贫油。这样也能避免燃油的爆燃损坏发动机。你可以通过检查最高转速来获知最大动力。另一种方式是检查发动机的排气管温度。粗略来看，当温度最高时动力最大。

控制混合比还能让你在同样速度和距离上，燃烧更少的燃油，这样可以飞的更远并减少污染。然而若你不懂如何控制它，将会产生很严重的问题。假设你在高空将油气混合比杆拉出，高空氧气含量较少，因此高空正确的混合比需要更贫油，也就是更少的燃油使用。然后你准备降落开始下降。如果

下降时你忘记将油气混合比杆推多一些，导致油气混合变得越来越贫油，直到发动机停止。

降落时，你需要将油气混合比调到稍微富油，也就是将混合比杆推多一些。这会让发动机更可靠一些并可以更好的适应高度的下降。

我在上文讲磁电机的时候，将磁电机置 OFF 位并不是让发动机关车的好方法。正确的做法是将油气混合比杆拉出。首先将油门杆完全拉出，这样可以让发动机耗油最少。然后拉出油气混合比杆，直到发动机因为混合了太多空气而停止。这可以确保发动机不会因为废燃油堵住。最后将磁电机置 OFF 位确保发动机不会突然起动。

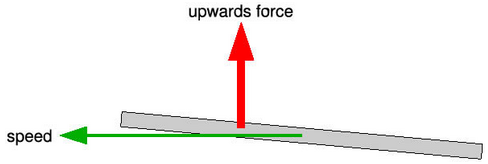
一个很重要的警告：你也许会以为转速表反映了发动机的动力。错。两个因素会让动力增加：发动机输出功率和飞机的速度。为了验证这个，飞到某个高度然后将发动机减到最低。尝试向地面俯冲，然后回到刚才的高度。你会看到转速表在你速度变化时有显著变化。它会在俯冲时变大在爬升时变小。

这带来的一个陷阱时当你打算降落时。假设你正在快速下降进入机场。你知道理想的降落转速大概是 1900 RPM。因此你将油门拉出，得到 1900 RPM 的转速。你认为使用了合适的转速，不再需要为此烦恼。然而当你拉平，飞机的速度开始降低，因为转速低。几分钟后飞机速度达到你预期的低速。你没再注意转速已经很慢了。此时你要么会掉高度，要么就会失速，也许两者都会发生。因此请注意油门和转速表。注意收油门的时候要更稳定，要随时准备将油门快速推回。

8.7.2 机翼和速度

假设你正在全动力飞行。轻微降低机头会让你降低高度而抬升机头则可以让你升高。你也许会想这很简单嘛。飞机飞行时的航向就是螺旋桨的指向。而这并不是最好的思考方式。此模型也许适合描述火箭，但不适合一架飞机。火箭是靠其引擎推进，而飞机则是靠机翼。这有巨大的差异。

找一个正方形的硬纸板，水平放在你的手上，手臂伸开并在水平方向快速移动。当纸板平着穿过空气是不会产生升力的。如果你将手臂旋转，让纸板有一个向上的角度，你会感觉到纸板将要飞到空中，一个向上的力作用在纸板上。这就是机翼让飞机留在空中的原理。给纸板越多角度，其升力就越大。直到角度越来越陡峭，你会感觉到下降。纸板进入“失速”状态（接着看下文）。



- 当你拉驾驶盘的时候，飞机机头向上。机翼以一个陡的角度穿过空气，因此机翼上的升力就变强了，飞机就升入天空。
- 当你推驾驶盘的时候，飞机机头向下，机翼与气流的角度变小，机翼上升力变弱，导致飞机下降。

重要的是机翼穿过空气时的角度，这就是迎角（Angle of Attack，AOA）[1]。

上面说如果机翼穿过空气的时候没有角度，将不会产生升力，这是错的。如果机翼像硬纸板那样全平的确实如此。但是机翼都有一个稍微弯曲的翼型。这就让机翼可以在没有迎角时，也能产生升力。甚至上在反向角度时，也能产生升力。高速飞行时飞机会稍微指向地面。

机翼穿过空气是一方面，另一方面则是速度。还是用硬纸板举例，现在不改变纸板的角度，用不同的速度移动手臂，你会发现移动速度越快，则其升力越大。

- 增加发动机动力，飞机速度加快，增加机翼升力，飞机会上升高度。
- 降低发动机动力，飞机速度减慢，减小机翼升力，飞机会降低高度。

稍微复杂一些：为了升的更高，飞机会倾向于丢失速度。而为了降低，则得到了速度。

这些都需要折中考虑。如果你想在恒定的高度以给定速度飞行，你需要同时调整发动机动力和驾驶盘/副翼（或者最好使用调整片，下文详述），直到获得你所需的高度和速度。如果你想以同样的速度下降，你需要轻轻推杆并收油门。也就是说你经常需要同时调节发动机动力和驾驶盘。不过，在正常飞行中，可以先调整发动机动力到合适的位置，然后再通过驾驶盘或调整片控制高度。

模拟器里你可以做这样的有趣实验，全速直飞，在水平飞行时尽可能获得最大速度。然后将发动机调到最低动力（慢车），然后控制飞机驾驶盘使飞机保持高度。飞机会慢慢下降，同时你需要更加努力向后拉杆以保持飞机水平。因为速度降低，机翼上的升力也降低，但是你通过增加迎角的方式补偿了速度的损失。这证明了飞机不是按照其机头指向航行。在此实验中，我们让机头抬起来保持高度，当飞机越来越慢，机头也会抬的越来越高，此时你会听到一声尖锐的警报，这就是失速警报（下文详述）。这表示飞机的迎角太大，以至于机翼翼型无法产生足够的升力，机翼不能产生升力，飞机会

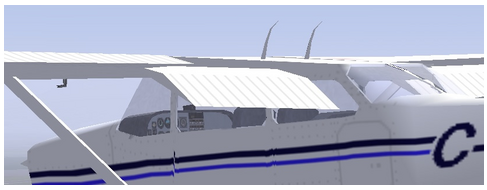
[1]注意迎角是飞机翼弦与相对气流之间的锐角，为了更好的理解这个概念，下文有精彩的讲解。“迎角”的另一个中文翻译是“攻角”，但不太常用。——译者注

迅速下降。唯一解决办法是推杆来减小迎角，让机头下降并推发动机动力到最大来获取速度。最后慢慢恢复正常驾驶。

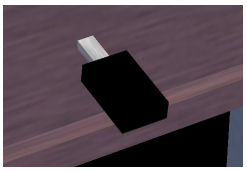
问题：控制飞机的高度和速度，用油门还是用驾驶盘，哪个更好？回答：这取决于你具体的情况。在正常飞行中，你会倾向于先设定合适的发动机动力，然后依靠驾驶盘和调整片。在起飞和降落阶段，使用驾驶盘和油门的程序比较复杂和严苛，与之前相反，先用驾驶盘和调整片控制速度，进而用油门控制高度并降低速度。下面的章节会深入讨论。

8.7.3 襟翼

襟翼是机翼后缘靠近机身两侧的活动部件。



你可以使用襟翼控制杆来放出和收起机翼：



你可以用鼠标点击，也可以使用 **[** 和 **]** 键来控制。**[** 键会收一档襟翼，**]** 会放出一档襟翼。按 **v** 键从外面观察并尝试用 **[** 和 **]** 键收放襟翼。

塞斯纳 172P 有四种襟翼设定：

- 0°：正常飞行和起飞时使用。
- 10°：用于短跑道起飞，慢速飞行时获取高度，或者进近着陆的第一阶段。
- 20°：用于快速降低高度和速度，比如在跑道上准备落地时
- 30°：用于更快的降低高度。

襟翼是很脆弱的，不要在速度 110 节以上时放出襟翼，也不要在 85 节以上速度时放出第二和第三阶段襟翼。

襟翼可以制造大量的阻力，这就是为什么在高于 85 节或 110 节时收起襟翼。

要检查襟翼的位置，可以使用鼠标的查看模式，回头看襟翼。也可以用**Shift-右箭头**快速切换到右侧查看，并快速使用**Shift-上箭头**回到原来的视图。

襟翼通过改变翼型以获取升力，放出第一阶段襟翼时在同等速度下机翼获得了更大的升力。这样你可以在起飞后更快飞入天空，同时也可以影响机头高度，这样可以在起飞和降落阶段有更好的视野。

襟翼也增加了飞机的阻力。第二和第三阶段襟翼产生的阻力大于升力，因此用襟翼可以让飞机减速。这在降落时非常有用，因为飞机滑翔性能非常好，如果你关闭发动机，飞机将会下降虽然很慢。你需要放出两档或三档襟翼来减速并向地面下降。

实际上襟翼在降落阶段需要更大的发动机动力。这会有些奇怪，为什么不将油门杆拉到最低并使用更少的襟翼呢？这是为了获得更好的减速效果和更多发动机动力，这样飞机可以更快的响应你给出的指令。当发动机失效时，收起襟翼并滑翔到跑道即可。

如果襟翼全部放出，而你需要更多的下降怎么办？慢慢蹬方向舵到某个方向，这将会让飞机侧翼承受气流并减速。然后通过适当转动驾驶盘（操作副翼）来补偿滚转倾向。这就是所谓的侧滑（Side-slip），这是一种有效降低高度的手段并可以在任何阶段改出[1]。

8.7.4 失速

飞机依赖机翼上流过的平滑气流来产生升力。然而若机翼的迎角太大，则会造成气流破碎，这样就不会产生升力。没有升力飞机就不能飞行，很快就会落回地面，这就是失速。

失速是一种危险的状况，任何速度下都可能会发生，在慢速飞行时更常见。飞行器都有一个失速速度，低于这个速度任何迎角都不能产生升力。你需要始终让飞机保持在失速速度以上。为此，飞机都安有警报器，在接近失速迎角时，会发出声音警告。

如果你进入了失速，补救措施是尽快降低机头，并增加发动机动力到最大，随后当速度恢复后将机头拉平。然而这么做会让飞机失去高度，因此在降落和起飞的时候，绝对不能进入失速状态！

当一侧机翼比另一侧先失速时，一般会在低速大坡度转弯时发生，此时飞机会进入螺旋[2]。因为一侧机翼还在提供升力，飞机会围绕失速一侧的机翼旋转，越转越快。为了从螺旋状态改出，你需要先用方向舵让飞机停止偏航运动，进入普通的失速，然后再如前文那样改出失速[3]。

---

[1]使用侧滑法降低高度最有名的例子是，1983 年加拿大航空 143 号客机（机型 767-233）因为燃油耗尽导致发动机空中停车。飞行员使用侧滑方式快速降低高度和速度，在最近的赛车道迫降成功。相关详情：https://en.wikipedia.org/wiki/Gimli_Glider。——译者注

[2]早期称为“tailspin”（尾旋）。——译者注

[3]通常来说，从螺旋中改出的动作第一步是油门慢车、副翼中立，在确定旋转方向后，使

像塞斯纳 172P 和 Piper Cub 这样的飞机，会很容易进入失速，但却很少能进入螺旋。而高速喷气式飞机，比如 F16 则很容易进入螺旋，却不太容易达到失速。

要在模拟器里练习失速，可以如此：

- 飞到一个恒定的高度和姿态。
- 降低发动机动力，并抬升机头避免下降。
- 继续减少发动机动力直到进入失速
- 尝试控制住飞机避免向地面下落
- 保持驾驶盘拉到最大位置，稳定飞机的姿态，机翼与地平线平行。尝试改变方向。
- 通过降低机头，推油门到最大来改出失速，当速度恢复以后修正姿态[1]。

你可以用不同的襟翼配置练习失速改出，或在高速飞行时突然改变姿态进入失速。

还可以尝试使用不同的机型，比较一下塞斯纳 172P 和塞斯纳“奖状”（Cessna Citation）喷气式飞机在失速表现上的异同，会发现失速更突然，而只有很少的警报。

8.7.5 升降舵调整片

升降舵调整片拨轮是一个黑色竖直的轮子，中间有灰色的点状突起。在仪表板下方可以找到：

用最大行程的反舵，制止旋转。对于某些飞机仅需松杆就可以从失速中改出，但有的飞机则需向前推杆以改出失速。当旋转减慢，逐渐回平方向舵，以避免进入反方向螺旋。当螺旋停止且方向舵回平时，逐步带杆退出下降。过多或突然的带杆或在恢复过程中使用副翼或未及时回平方向舵可能会导致二次失速和另外的螺旋。——译者注，摘自中国民用航空局飞行标准司-咨询通告，编号 AC-61-FS-2013-19

[1]安全起见，整个失速改出练习高度不应低于 1500ft AGL。——译者注

在 *FlightGear* 中，使用 **Home** 键和 **End** 键控制调整片。**Home** 控制拨轮向上转，**End** 控制拨轮向下转。你可以在拨轮上方和下方点击来控制。

近似来说，调整片与驾驶盘的作用是相似的，也可以完成升降舵的操作。向下转动拨轮，相当于向后拉驾驶盘。然而有个巨大的不同，那就是调整片调整以后会保持位置，而驾驶盘除非你一直施力影响，你放开驾驶盘会自动回中。

巡航阶段，为了维持飞机在恒定的高度，副翼可能不在中立位——依赖飞机周围的空气，当前的燃油量和配载。显然，飞行员一直控制着驾驶盘维持高度很快会疲劳。使用调整片可以“卸掉”升降舵上的力，驾驶盘也可以回中。

在起飞阶段，调整片必须回中。否则你会发现要么飞机抵抗起飞，很难拉起机头到正常位置，要么很快就起飞了。

在降落阶段，善用调整片让你的升降舵在中立位。这样只需小幅调整姿态就可以很容易控制了。对塞斯纳 172P 来说是调整片回中，而对 Cherokee Warrior II 来说就是调整片“拉出”。

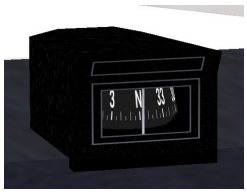
调整片拨轮的移动比驾驶盘慢的多，允许精细的调整片操作。操作时请耐心一点。

8.7.6 我的飞行方位是?

知道你的飞行方位显然是个好注意。有三种方式来获取你在空中的位置:

- 透过窗外景物。识别地面特征，比如道路、山丘、桥梁、城市、森林等。在模拟器中，你只能使用很狭窄的观察窗口查看虚拟世界。有如下方式让你可以在飞机里平移视角。
  - 使用 **Shift** 键搭配四个方向键来观察前后左右。
  - 使用 **Shift** 和小键盘的按键来观察前后左右，还有斜前方斜后方。
  - 在正常模式或控制模式按住鼠标右键，同时移动鼠标观察四周。
  - 在鼠标的查看模式（使用 **Tab** 键切换鼠标指针是 ↔ 时）。这可以让你查看任何位置，包括上和下。按鼠标左键，可以回复正中。
- 磁罗盘。位于仪表板正上方，罗盘非常简单，但却能很好的帮助飞机飞行。另外磁罗盘在地面会受到干扰而异常，还有磁罗盘指向的是磁北极，而不是地理北极，相应的差异与你所在的地区有关系[1]。

[1]有关地区磁偏角相关内容可参考：https://en.wikipedia.org/wiki/Magnetic_declination。——译者注



- 方位陀螺仪（如下图）或称“航向指示器”（Heading Indicator）。陀螺仪由一套真空系统驱动，与磁罗盘匹配，但不会受到磁力和飞机移动的影响。然而因为陀螺进动和摩擦的影响，一段时间以后陀螺仪会漂移，需要参考磁罗盘重置陀螺仪。要重置 HI，在巡航状态下，使用仪表左下角的黑色旋钮。在正常模式下，用鼠标点击旋钮左半边或右半边来调节，按鼠标中键可以更快调节。可以用 **Ctrl-c** 查看可操作的高亮区域。右下角的红色旋钮是用来告诉自动驾驶仪你打算飞的航向（**HDG** = “heading’’）。

8.7.7 关于仪表板

最后让我们来看看仪表板，上面介绍了一些，下面介绍剩下的：

六块主仪表

让我们从最重要的仪表开始，任何模拟飞行员都必须了解的，被称为“holy six”或者“six-pack”的主仪表。

仪表板正中靠上（见图 5）的是人工地平仪（姿态指示器），显示了飞机的俯仰和横滚角度。俯仰和横滚都都标示了 10、20、30、60 和 90 度这几个刻度。

人工地平仪的左侧，你可以看到空速表。除了指示空速以外，空速表还标出了一些你必须关注的速度指标。首先绿色弧线表示襟翼全收时空速可操作范围。白色弧线表示使用襟翼时的范围。黄色弧线表示只可在平缓空气中使用的警戒速度范围。其他的红色表示绝对不可超过的界线，除非你想让飞机在空中解体……

在空速表下方你可以找到转弯侧滑仪。中间的小飞机指示了你飞机的横滚。如果小飞机的左机翼或右机翼处在标线的位置，说明是一个标准转弯，也就是转 360 度需要精确的两分钟。

在小飞机下方，还是在转弯侧滑仪里，是倾角仪。它用来显示方向舵和副翼是否配合良好。在转弯时，你需要始终协调操作副翼和方向舵并保持管子里的小球处在正中的位置。否则飞机会侧滑。一个简单的规则是“跟着球走”，比如当球向左手边移动时，就向左侧蹬方向舵。

如果你没有方向舵踏板,或者对方向舵和副翼之间的操作比例,没有经验的话，你可以在启动 *FlightGear* 时使用 `--enable-auto-coordination` 选项。

在人工地平仪的右侧，你会看到高度表，显示了飞机距离海平面（不是地面!）的高度，用百英尺表示。在高度表下方是升降速率表，指示了飞机爬升和下降时的速率，单位是百英尺每分钟。你会发现在某些情况下，升降速率表比高度表更舒服，注意升降速率表会有一定时间的滞后。

在升降速率表下方的是螺旋桨转速表，或者 RPM（Rotations Per Minute）指示器。以每分钟百转为单位。绿色弧线表示巡航时的最佳范围。

主仪表板还包括陀螺仪罗盘，就在人工地平仪的下方。除此之外还有仪表板中间最上方的磁罗盘。

你会发现四块最重要的仪表构成了“T”字形：空速表，人工地平仪，高度表和罗盘。你需要在飞行时不断扫描这些仪表[1]。

辅助仪表

除了这六块主要仪表，还有很多辅助仪表。最左侧你能看到时钟，对于计算转弯至关重要。时钟下方有很多小标尺，表示发动机的各种技术参数。其中最重要的是油量表——飞行员必须知道。

点火开关在仪表板的左下角（见图 4）。有四个位置“OFF”、“L”、“R”和“BOTH”。第一个很显然，“L”和“R”不代表左右两台发动机（塞斯纳 172P 只有一台）而是代表两个磁电机，用来在故障时提供冗余。两个开关位用来在飞行前测试。在正常飞行中开关需要置为“BOTH”。磁电机开关的最右一档是用来起动发动机的，使用一个电力驱动的点火开关（使用 **s** 键）。

在驾驶盘下方的是停留刹车。竖直时表示设置了停留刹车（ON），否则表示停留刹车解除（OFF）。使用 **b** 键控制。

无线电

仪表板右手边的是无线电栈。你可以看到两台 VOR 接收机（NAV），一台 NDB 接收机（ADF）和两台无线电通话电台（COM1/2）以及自动驾驶仪。

无线电通话电台用来和空中交通设施通话；它只是使用特殊频率范围的普通无线电收发装置。其频率用 LED 显示出来。一般有两台 COM 收发机；

[1]扫描仪表的一个方法是，目光叮住人工地平仪，同时快速横向扫描空速和高度，恢复叮住地平仪，再向下扫描罗盘。多多练习在不同飞行动作时扫描仪表的方法，最终熟能生巧。——译者注

这样你可以收听下一个要联系的管制员的同时，依旧保持与前一个的联系。

COM 无线电可以用来收听机场当前的天气状况，也就是 ATIS。要这么做只需要调到相应机场的 ATIS 频率即可。你可以从 `ATC/AI->Frequencies` 菜单项找到邻近机场的 ICAO 四字代码。

每台 COM 无线电有两个频率配置，一个“活动”的频率也就是此时飞行员正在联系的，另一个“备用”的频率是用来改变的。这样你可以守听一个频率的同时，调节另一个频率。

可以用鼠标改变无线电频率，用鼠标点击相应旋钮的左右两侧，通信旋钮旁边的开关用来切换活动频率和备用频率。

后面的章节将会讲到使用自动驾驶仪和无线电导航设备。此刻你可以忽略无线电仪表，因为你现在遵循的是 Visual Flight Rules 目视飞行规则（VFR）。

8.8 让我们去飞吧

到现在为止，你已经会平直的飞行了，也学会了起飞时使用方向舵来维持角度，还学会了下降、爬升和温柔的转弯。这一节我们将会教授更加真实的起飞和降落，还有一些你需要知道的深入理念。

8.8.1 真实的起飞

下面的原则是正常起飞时需要关注的：

- 前轮需要在 40 节时离开跑道。
- 起飞后你需要立刻加速到 70 节，高于失速速度，以应对突然的阵风或者发动机失效。
- 不要超过 75 节，这样可以让你尽快获得想要的高度。
- 爬升到 500 英尺之前要一直沿着跑道方向，以防此时发动机失效，这样你可以直接落在跑道上。
- 飞越建筑物至少要 1000 英尺。
- 离地比较近时要温柔的转弯，并适当使用方向舵。

所以，你需要稳定在 75 节来起飞和爬升。然后在 40 节轻抬机头，飞机会在大概 55 节就起飞了。为了尽快加速到 75 节，你需要在起飞后慢慢降低机头，然后到达 75 节，你再通过驾驶盘来控制速度。

把以上所有之前学到的放到一起，使用鼠标来完成一次正常起飞如下：

1. 首先调整高度表到正确的高度，主要是机场标高。比如 KSFO 是在海平面高度——0 英尺。

2. 观察副翼和升降舵都已经回中，可以观察驾驶盘来确定[1]。
3. 按 **Tab** 键让鼠标进入控制模式。
4. 按住鼠标左键来控制方向舵。
5. 发动机动力推到最大（按住 **PgUp** 键，直到油门杆完全推入）。
6. 当飞机在跑道上加速时，用鼠标做一些细微调整以使其保持中间。
7. 在 40 节时，松开鼠标左键，轻轻抬起前轮。现在鼠标在控制驾驶盘。
8. 飞机将会在大概 55 节时飞离跑道。
9. 慢慢降低机头加速到 70 节。
10. 保持与跑道中线对齐
11. 用驾驶盘稳定速度在 70 节爬升。如果空速降低，则推驾驶盘，否则拉驾驶盘。
12. 到 500 英尺高度，向你需要的航向转弯，飞越建筑物至少需要 1000 英尺。

8.8.2 降落

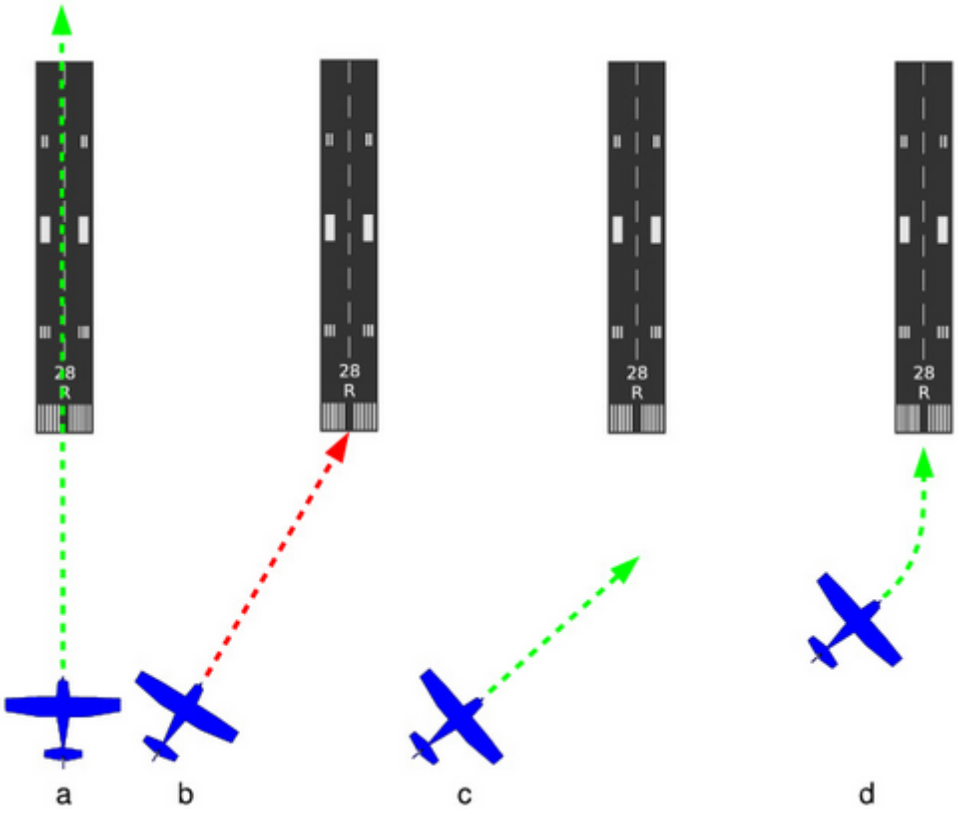
降落几乎和起飞一模一样，只不过顺序相反：

- 邻近地面，转弯要温柔并恰当配合使用方向舵。
- 最后进近落地之前，保持 500 英尺以上。
- 进入跑道时在 70 节左右。
- 55 节时两侧机轮接地
- 40 节时让前轮接地

如果你能在跑道上选出一个目标点，对着这个目标点落地，那么就会简单许多。观察这个目标点，你可以很容易的了解，降落的是否太快或者太慢。如果目标点向前移动了，说明你下降的太快。

显然你需要对准跑道。这就意味着你的飞行方向与跑道中线一致（如下图中的 a）。为了可以对准跑道，不要瞄准跑道的起始位置（如下图中的 b），而是瞄准跑道前方一个虚构的点（如下图中的 c），然后离这个虚构点还有一段距离时开始温柔的转弯对准跑道（如下图中的 d）。注意在降落转弯过程中，你的操作必须非常轻柔。你甚至没办法注意转弯侧滑仪，这就是为什么要依赖外部地平线，而不是飞机仪表。

[1]此时最好按小键盘上的 **5** 键确认回中。——译者注



使用鼠标降落的基本流程如下：

1. 距机场有 1500 英尺高度时，还在几英里开外，并已经对准跑道了。收油门到大概 1500 RPM。这可以让你减慢速度并开始渐进下降。
2. 在 115 节以下，放出一档襟翼（按] 键）。这会增加升力同时也让你慢下来。
3. 使用调整片让飞机继续下降
4. 在大概 1000 英尺时，再放出一档襟翼。这会显著增加阻力，但也同时会让机头向下，更利于观察。
5. 使用升降舵和调整片来控制速度，如果低于 70 节则拉杆，否则要推杆。如果你在使用游戏杆，随时通过调整片来释放杆上的压力
6. 使用油门来调整高度，如果你飞的太高则收油，否则就推油门杆。判断太高或者太低可以通过观察跑道上的数字来确定。如果数字向上移动说明下降太快——增加油门，如果数字向下移动说明高度太高需要收油门。
7. 小幅调整对准跑道

8. 在 500 英尺放出全部襟翼。这将会显著增加阻力，所以记得要增加发动机动力，保持连续下降

9. 在刚好跑道上空时，降低油门到慢车，并用驾驶盘让飞机温柔的指向水平。这就是“拉平”操作，会导致飞机与跑道表面有数英尺的距离。在正确的高度拉平是很难掌握的一门技术，简单来说，注意地平线而不是一直叮着目标点。

10. 继续保持机翼水平，我们需要让两侧机轮同时落地。

11. 继续向后拉驾驶盘，主轮会在 55 节左右落地，也就是“Flare”。

12. 落地以后，用方向舵让飞机保持直线（小键盘上的 **0** 键和 **Enter** 键）

13. 低于 40 节，降低机头让前轮接触地面。

14. 按住鼠标左键来控制前轮/方向舵

15. 低于 30 节，使用刹车（按 **b** 键）进一步减慢飞机速度。

当飞机停住或者以慢速前进时，放开 **b** 键并增加一点发动机动力，以便滑行到停机位或机库。

8.8.3 关闭发动机

- 设置停留刹车，按 **B** 键。
- 油门慢车（按住 **PgDn** 键）。
- 将油气混合比杆拉出，切断油路（正常模式下用鼠标点油气混合比杆左侧就可以将其拉出）。
- 将磁电机点火开关转到 OFF 位（按几次 **{** 键）。

8.8.4 中止降落

在降落时你需要时刻准备，因为状态不好或外部因素而中止降落，比如：

- 塔台的指令
- 错误的速度或降落角度，因为已没有时间来修正
- 强劲的阵风
- 跑道上出现飞鸟

要中止降落，将油门推到最大（按住 **PgUp** 键），抬升机头以便爬升，当爬升姿态建立以后，收起襟翼。

降落要比起飞困难很多，在一个大型机场的跑道降落比如 KSFO（默认的旧金山国际机场），要比小一点的跑道比如 KHAF（半月湾机场，在 KSFO 西南 10 英里处）要困难。

要练习降落，可以使用下面的命令行选项，启动模拟器并飞向跑道。飞机会被置于跑道外 5 英里处，高度 **1500** 英尺，速度 120 节。

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

使用 65 节而不是 70 节进近落地会让短跑道降落更容易。然而也需要更高超的控制技巧，因为更接近失速速度。因此与 70 节时很不同。

8.9 关于风的问题

设想一个热气球，它在一个巨大的空气立方体里。也许这个空气立方体相对地面以高速运动，然而热气球却在立方体里纹丝未动。无论风速如何，在热气球里的人根本感觉不到一丝风。

同样道理，飞机在周围气团里飞行与在一个巨大的空气立方体里飞行是一样的。这个空气立方体相对地面的运动对飞机并没有影响。

你，飞行员，则不同。你更关心周围空气相对地面的速度。因为这个可以让你向左或向右偏移，又可以影响你更早或更晚的抵达目的地。

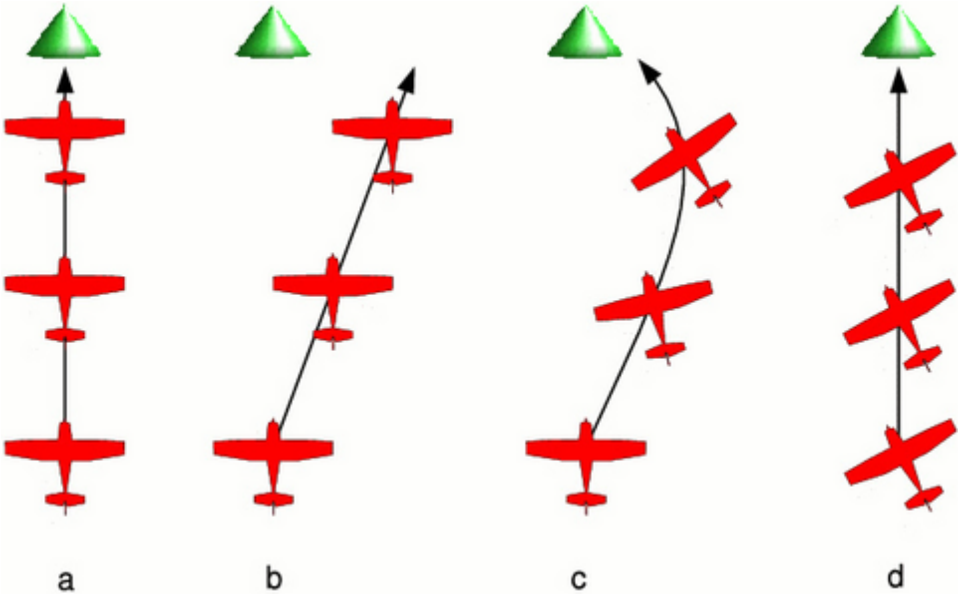
如果风与你飞行的方向相同，风就会把自身的速度叠加在飞机的空速上。因此相对地面你会更快，将更早抵达目的地。

反之当风从机头向机尾方向吹的时候，风会抵消飞机的空速，结果相对地面你变得更慢。会晚于预期抵达目的地，也让你有更多时间欣赏风景。

以上两种情况很好理解，更复杂的则是当风从侧面吹来的时候。来看下面的图表。

- 下图中的 a 表示无风。飞行员想要抵达北面绿色的山丘。他可以直接航向山丘朝向正北，之后就会抵达。若没有风，你只需要将机头指向你的目的地即可。

- 下图中的 b，飞行员还是将飞机航向正北。然而却有左侧的来风，也就是从西面。飞机会被吹到右侧偏离航线。
- 下图中的 c，飞行员保持机头指向山丘。他虽然会抵达，但飞机会飞曲线路径。这会浪费一些时间。同时这样的曲线路径会在精确导航时变得很可怕。
- 下图中的 d 表示了最优的方式来抵达山丘。飞机指向山丘的左侧，稍微偏西也就是来风方向。这会抵消掉风的影响并让飞机直飞山丘。



飞机向左或向右偏航多少度？严肃的飞行员会使用几何和三角函数来计算正确的角度。粗略飞直线时你不需要计算。简单方法是瞄准你要飞行的点，注意它怎么移动。你会意识到自己漂移到了左侧还是右侧。然后让本能驱使你慢慢将飞机指向左或右，来抵消这股漂移的力量。刚开始你可能需要思考一下，很快这就会变成本能，就如同之前你学习如何直飞一样。你就不会一直让飞机指向目标，而是让飞机飞向目标。

飞的越快，风偏修正就越小。

8.9.1 侧风起飞

侧风起飞是很严峻的。因为机场设计时都会避免这种情况，会让跑道与盛行风平行。很多机场都会有多条跑道，这样做的目的是尽可能让跑道与风平行。

顶风起飞会让飞机更容易起飞，因为机翼与相对空气的速度加快增加了升力。当无风的时候，飞机必须加速到 55 节来起飞。也就是如果有 10 节的

顶风，飞机就已经有了 10 节的空速，这时候只要加速到相对地面 45 节时就可以起飞，也就是缩短了起飞滑跑距离。

正如顶风会缩短滑跑距离，顺风则会增加距离。哪怕只是一两节的差距，就会产生滑跑距离上的巨大差异。几乎所有跑道都可以从两侧起飞，你可以很容易的从跑道另一侧起飞，并找到顶风。

要想获得风向和风速，最好是跑去询问塔台或者通过无线电询问之。另一个必要且备份的工具是跑道两旁的风向袋。可以指示风速和方向，如果风向袋变得越来越直越来越硬，说明风速越大。下图里的风向袋表示风速 5 节：

不幸的是，机场跑道并不总是正对风，你需要在侧风中起飞。
技术上说与正常起飞只有两点不同：

- 在起飞滑跑阶段，飞机会像风向标一样偏转。你需要用方向舵来修正，也许会需要方向舵有较大的角度来保持对准跑道。你需要在整个起飞过程里都保持方向舵。
- 起飞以后，飞机会因为方向舵的影响而转弯，你需要使用副翼来修正。当飞机升入空中以后，你可以适当减小方向舵和副翼，然后再对风进行修正。正如上文所述对准跑道。

8.9.2 侧风降落

侧风降落与起飞非常相似：

- 修正侧风对准跑道
- 在刚开始拉平时，用驾驶盘让飞机平直，使用方向舵抑制飞机的偏航倾向。
- 飞机可能会一侧机轮先着地，用方向舵让飞机另一侧机轮着地。

相应的技术细节可以参考：slip landing. 另一篇文章：crab landing[1]

[1]现时，这两篇维基百科链接都会指向同一个页面。——译者注

8.9.3 在风中滑行

10 节以下的风，对塞斯纳 172P 来说并不需要特别的处理。但突然增大的风会让飞机倾斜甚至将其掀翻。

若想模拟地面有风时的滑行，可以将模拟器配置比较大的风，比如 20 节。这样的风任何时候都足以将飞机倾斜甚至掀翻。一个微小的错误就可能让飞机失事。

总的原则是你必须向风的风向顶杆。这需要用物理的原理做一点解释：

- 当风从 12 点钟方向（正前方）吹来时，这比较合乎逻辑，向 12 点钟方向推驾驶盘，升降舵会让机尾稍稍抬起。这样可以防止飞机在风中被掀翻。

- 当风从 10 点钟方向（左前方）吹来时，将驾驶盘向 10 点钟方向推，这意味着升降舵几乎回中，同时左侧副翼向上而右侧副翼向下。这样会压住左侧机翼并抬升右侧机翼。同时这也是防止飞机被风吹翻最稳固的位置。

- 当风从 8 点钟方向（左后方）吹来时。你也许想将操作相反，以便让左机翼推向地面。因此你需要推杆到 4 点钟位置，错！保持驾驶盘在 8 点钟位置。右侧副翼向下的位置，它的作用与金属档板一样。同时还能增加右侧机翼的升力，这正是我们想要的。相应的左侧副翼向上也可以降低左侧机翼的升力。

- 当风从正后方吹来时，也就是 6 点钟方向。这时候拉驾驶盘（指向 6 点钟方向）。向上的升降舵会让尾翼向下推。而这也是最佳位置。因为强风会将尾翼推向地面。虽然这一点难以理解，但机尾设计时已经考虑到这一点。

如果想正对风移动，则需要更多发动机动力。而当风从后面吹来时，你也许根本不需要发动机动力。总是保持发动机在最低需求。

特别的，转弯时要特别慢。一次只做一点改动。慢慢移动并谨慎考量驾驶盘的角度，连续的对着风的方向推杆。连续的降低发动机推力。时刻注意使用刹车不要太猛，如果让飞机突然刹停，很可能恰好进入风将飞机吹翻的角度。

8.10 自动驾驶仪

自动驾驶仪并不是一个“智能”的飞行员，它只能解决一些简单重复劳动。你作为飞行员必须一直关注一切动向。做好在其发生错误时关闭自动驾驶的准备，无论是在真实飞行还是模拟器里。

自动驾驶仪安装在驾驶盘的右侧：

按上面的 **AP** 按钮来启动。自动驾驶就开始控制飞机的横滚，让机翼保持与地平线平齐。如下图所示，自动驾驶仪上会有橙色的 ROL 标识。要关闭自动驾驶再按 **AP** 按钮即可。

如果你按 **HDG** 按钮自动驾驶仪会让飞机飞向并保持某个航向，这个航向可以通过方位陀螺仪的红色游标来设定（详见 8.7.6 节）。HDG 表示 Heading（航向）。再按一次自动驾驶仪上的 **HDG** 按钮会回到横滚控制模式（也可以用 **AP** 按钮关闭自动驾驶仪）。

ALT、**UP** 和 **DN** 按钮用来告诉自动驾驶仪，要么控制垂直速率（VS），要么控制高度并保持（ALT）。要了解更多塞斯纳 172P 自动驾驶仪的高级模式，可以参考此文档：Bendix King 的网站

8.11 然后呢?

此教程讲解了塞斯纳 172 的基本飞行方法。从这里你可以探索 *FlightGear* 提供的更多特性。

当你掌握了此教程的内容以后，你可以看看本指南里的其他教程，比如飞向其他机场，天侯不佳时的仪表飞行以及直升机的飞行。

此教程略过了一些真实飞行员必须关心的问题：

- 如何使用真实的检查单

- 如何在短跑道做紧急降落，以及发动机失效
- 如何依据大气规律、航图、法规、无线电信号和天气状况来导航
- 如何创建飞行计划，并按其精确飞行
- 如何放置人员、燃油和行李以得到正确的重心。
- 如何处理塔台和其他飞行器
- 如何处理多油箱及其系统
- 如何处理飞机各个系统的失效

此教程也没有涉及更高级飞机的特性，包括：

- 起落架的收放
- 可变桨距的螺旋桨
- 多发动机
- 喷气式发动机

8.12 鸣谢

我想鸣谢：

- Benno Schulenberg 帮我纠正了此文中的英文错误
- Albert Frank 告诉我大量飞行员的数据并修正了技术性错误
- Vassilii Khachaturov 教会我有关 *FlightGear* 的新特性
- Dene Maxwell 有关在 Windows Me 下出现问题的处理
- Mark Akermann 和 Paul Surgeon 感谢他们的备注
- Michael “Sam van der Mac” Maciejewski 翻译了此教程的波兰语版本并将其用到了此指南里
- *FlightGear* 邮件列表里各位网友的热心帮助
- 4p8 此教程使用了我朋友 Frdric Cloth 的网页空间

8.13 其它机型

我已经交叉检查了所有关于塞斯纳 172P 的数据，一个飞行员朋友证实了我并没有写太多垃圾，而且我在模拟器里也做了很多测试飞行。本节的内容基于我在模拟器里飞其它机型的经验，数据并不那么可靠。对于介绍相应的机型操作很有用，但仅仅满足基本飞行。

8.13.1 如何降落 Cherokee Warrior II

Cherokee Warrior II 比塞斯纳 172P 更高级。因为其下单翼上反角设计在侧风时没有那么敏感。完全放出的襟翼提供了更多减速，并可让其在更短的距离内降落。

在 *FlightGear* 里起飞和塞斯纳 172P 是一样的。在真实中，两者的检查单并不完全一样。

注意 Cherokee Warrior II 降落时有少许不同：

- 平飞准备落地时，调整片必须稍微前推，以让驾驶盘处在中间位置。
- 最佳降落时的转速是稍稍低于转速表的绿色区域。大致保持表针竖直。
- 降落时只使用两档襟翼，不要收油门太多。
- 如果你保持两档襟翼降落，拉平和接地的过程与塞斯纳 172P 是一样的。然而三档襟翼会让空速显著下降，会快速接触跑道并很快停住。注意尽快降低前轮。（也可以在进近阶段使用三档襟翼下降，而不是收油门。在第二档向第三档襟翼过渡时开始对准跑道，不过保持两档襟翼并收油门似乎简单一些。一个有趣的特技玩法是平稳飞到跑道入口点，然后将发动机调到最低并放出三档襟翼。飞机几乎是直接落在跑道上。此方法很有趣但也能凑合[1]）

在真实中，塞斯纳 172P 比 Cherokee Warrior II 更有优势的是油箱靠近机翼中间并高于发动机。另外还有一个自动油路转换开关。这意味着你不需要在飞行中关注燃油到发动机之间的问题。而 Cherokee Warrior II 的油箱是分离的，在两侧机翼并低于发动机。这意味着你需要在飞行中要不断切换两个油箱，如果一个油箱比另一个油箱轻，则会造成飞机的不稳定。而低于发动机则需要你来控制燃油泵和备用燃油泵。

一些链接：

- http://en.wikipedia.org/wiki/Piper_Cherokee

[1]译者亲测，在 FlightGear 里若采用此法降落，需要较高的控制高度能力，否则很容易造成接地瞬间过高下降率（超过 300 英尺/分钟），有可能对机轮造成损坏。因此不建议采用此特技方式降落。——译者注

- http://freechecklists.net/Resources/Piper/PA-28-151+Warrior/ [1]

8.13.2 如何起降 Piper J3 Cub

Piper J3 Cub 与塞斯纳 172P 和 Cherokee Warrior II 有很大的不同。塞斯纳 172P 和 Cherokee Warrior II 的起落架是前三点式（也就是前机轮在主机轮的前面），而 Piper J3 Cub 是后三点式的飞机（主机轮在前）。后三点式飞机在起飞和降落时更困难。跑道上滑跑起飞时，你需要更加稳健的使用方向舵踏板。Piper J3 Cub 是介绍后三点式飞机最好的机型，合适的流程可以更简单的起飞和降落。失速速度略微低于 40 mph（此飞机空速以 mph 来表示）（大概 27 节）。起飞低于 50 mph。

我在飞 Piper J3 Cub 时的经验是向后完全拉杆，同时发动机油门推到最大。当前轮离地以后，再慢慢将驾驶盘回中。后面就和正常起飞一样了。然后让飞机加速到 50 mph。然后控制驾驶盘让飞机略高于 50 mph 来爬升。

降落流程与 172 很不一样，因为这架飞机没有襟翼，而且非常轻。

1. 500 英尺高度并保持 “恰好” 52 mph 速度对准跑道。让发动机盖子对准跑道入口，此时发动机盖子会完全挡住跑道。要想看跑道的位置，慢慢推杆然后再拉回正常。

2. 当跑道入口对准仪表板的位置时（如果你的视线可以穿过仪表板），收油门到几乎最低并开始向跑道入口俯冲。使用驾驶盘保持 52 mph。如果你发现自己即向错过跑道边缘则稍稍推油（注意一股小风都可以对 Piper J3 Cub 产生较大影响）。

3. 拉平并让发动机置于最低。不要拉的太猛，让机轮首先接触跑道。

4. 当机轮接触跑道以后，推杆以便让后轮悬空。你也许会想螺旋桨会撞到跑道，而飞机也会翻过来并损坏吧。然而一切都还 OK。机翼的负角度会让飞机慢下来（不要在其他机型上如此推杆，即便外形与 Piper J3 Cub 很相似。这样做会导致翻覆）

5. 驾驶盘已经推到最大，此时按住鼠标左键控制方向舵，让飞机在跑道中线滑行。这并不简单，一个窍门是不要在飞机刚刚向左转的时候就说飞机已经向左转了。

6. 当速度已经很低（方向舵控制也稳定了），你会看到后轮会落到地上。松开鼠标左键并回到驾驶盘控制模式，并将驾驶盘完全向后拉。机尾接触地面而机头抬高。现在你可以使用机轮刹车（按 **b** 键）（如果使用刹车太早，机头会撞到地面）。

上面所述的起飞流程与第一降落流程对应。还有一个第二起飞流程与第二降落流程对应，然而我并没有成功，所以也没写出来。

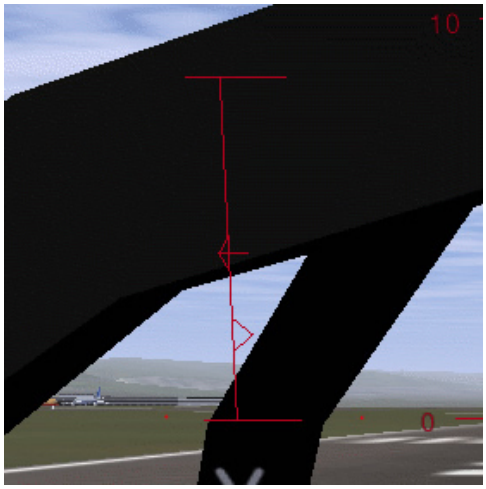
[1] 此链接也很有用 http://wiki.flightgear.org/Piper_Cherokee_Warrior_II ——译者注

8.13.3 如何起降喷气式飞机

起飞喷气式飞机很简单，但你必须有快速反应能力。在 FlightGear 下我最喜欢的喷气式飞机是 A-4 Skyhawk。在 Linux 下可以使用命令 `-aircraft=a4-uiuc`，前提是它已经安装了。

下面是一个“简朴的”起飞流程：

- 按 **h** 键两次以打开红色并全屏化的 HUD。油门显示在 HUD 的最左侧。
- 空速表在仪表板左上部以“KIAS”表示。你也可以使用 HUD 上的空速指示。
- 将油门推到 $\frac{1}{2}$ 位置。
- 保持驾驶盘在其后拉行程 $\frac{1}{2}$ 的位置（如下图所示，图中线右侧的红色箭头）

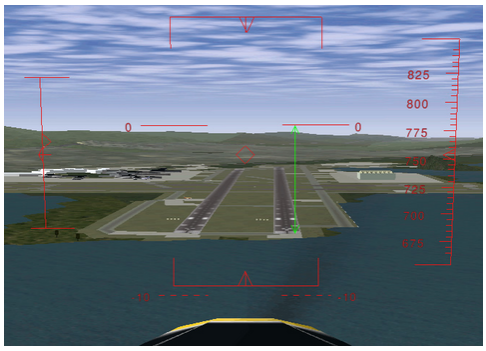


- 不需要一定用方向舵以保持跑道方向，因为飞机会在其偏离跑道前起飞。（为了“安全”起见，还是要保持在跑道中线的，但对初学者来说用方向舵会很忙碌）
- 超过 160 节，飞机会抬起前轮，此时立刻拉操纵杆到中立位或稍向后，这会让飞机稳定在空速 200 节（这会让飞机平滑的爬升），我也不知道为何是 200 节，对真实 A-4 来说是不是正确。而我更希望使用 AOA（见下文）。
- 按 **g** 键收起起落架。
- 保持 $\frac{1}{2}$ 的发动机推力，或者 200 节来穿过云层。或者收油到 $\frac{1}{4}$ 左右来正常飞行（当然你可以用全发动机推力飞行，更有乐趣）。

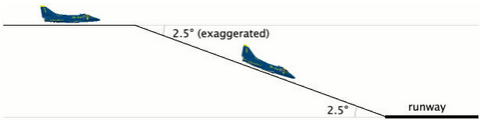
更“紧张”的起飞流程与此相似，不同的是，使用全部发动机推力。飞机爬升很快而且你需要更陡峭的爬升角度以维持 200 节。也需要更快的收起起落架。

降落喷气式飞机和小型螺旋桨飞机方法不一样。通过在网上找到的一些文章，这是我的降落方式：

- 距离跑道很远，保持 2000 英尺高度 200 节速度。然后放下起落架（按 **G** 键）并放出全部襟翼（总共三档，按三次 **]** 键）。
- 稳定保持 1000 英尺并准确减速到 150 节。使用升降舵控制高度，使用油门控制速度（与塞斯纳正好相反）
- 尝试对准跑道
- 如何知道何时必须开始向跑道俯冲呢？你需要使用 HUD；完整的 HUD 有很多特性。如下图。当你看到“0”线与跑道入口之间的“距离”，是“0”线与“−10”点线之间距离的 25% 时。说明可以开始俯冲。瞄准跑道入口处（下图的“距离”大概 64%，距离降落还很远）。



让我们来解释一下，两条标示为“0”的水平线段标示地平线。也就是水平线的位置，如果地球是平的。当你眼睛对准这两条“0”线时，你的目光是水平的。现在观察“−10”的虚线，这表示理想水平线下 10° 的位置。也就是说：当你看着目标“隐藏”在“0”线以下时，你需要把目光降低 10° 看“−10”的虚线处的“隐藏”物体。说个很重要的比喻，如果有个人在划船，位于“−10”虚线的位置。他如果要看你的飞机，需要抬起头 10°。也就是说他看到你的飞机处在地平线上 10° 的位置。如上图所示，跑道入口在红色“−10”虚线的 64% 的位置。这就意味着你需要降低视线 6.4° 以看到跑道入口点，也就是下降角度需要 6.4°（太陡了）。所以 HUD 可以帮助你测量下降的角度。喷气式飞机需要 2.5°（最大 3°）的下降角度，也就是距离“−10”虚线 25%（最大 30%）。



- 开始下降到跑道，使用驾驶盘保持对准，使用油门杆保持 150 节速度。
- 注意理想水平线与跑道入口之间的角度，必须保持在 2.5 度（25% 的 10° 线）
    - 若角度高于 2.5°，说明你高于下降路径，你需要更快的下降高度。同时降低发动机动力并降低机头。
    - 若角度低于 2.5°，说明你低于下降路径，我不会说让你增加高度，而是你不要降低高度太快。同时增加发动机推力并稍抬机头。
- 当非常接近跑道入口时，不要拉平，不要像塞斯纳 172P 那样拉杆。只需要让飞机高速接触地面，让飞机砸在跑道上。也就是说三个机轮几乎同时落地。然后将发动机收到慢车（如果你尝试通过拉杆让飞机悬停在跑道，会让机头猛然抬高，如果是 F-16 上会刮到机尾，有可能造成损坏）。
- 按住 **b** 键并使用方向舵让飞机对准跑道。只需要很小的方向舵移动，否则飞机有可能会侧翻。

在真实中，HUD 会提供一个指示飞机移动的游标。会如下图所示，当你在一个固定高度飞行，此游标会在理想水平线上。当你下降对准跑道入口，你需要将此游标对准跑道入口，这会很容易并精确对准跑道（FlightGear 里的 HUD 中间有一个菱形标志，有时也很有帮助，但目的不一样。它表示了飞机机头的指向。比如当你以较低的速度向地面下降，游标应该指向地面，然而 FlightGear 的菱形会指向空中。另外 B-52 上的 HUD 有速度游标，在降落时很有用）。

同样在真实中，HUD 还会有一条 $-25°$ 线来协助找到正确下降角度。只要将此线保持在跑道入口即可。

此外关于空速，快速的军机依赖于使用迎角。迎角（AoA）是机翼与相对气流之间的锐角。保持最优的迎角降落有个好处是不用依赖于飞机的配载，就如同最优的空速一样。为了保证每次落地的迎角都是正确，你需要保持正确的空速，无论配载如何。

迎角也在 HUD 上显示出来了，如下图的三个灯。当上部的 ∨ 亮起，表示你的迎角太大，需要向下俯。当下面的 ∧ 亮起表示迎角太小，需要向上抬头。中间的 ◯ 表示迎角正好。显然，俯仰会影响到下降率，你需要同时配合油门的调节。

塞斯纳 172P 和 A-4 Skyhawk 是两个极端，大多数飞机都在这两个极端之间。如果你能够掌握这两种飞机（和一两种后三点式飞机），你将会明白大多数飞机的起降。

160 节对 F-16 来说是很合适的降落速度，当然也需要飞机落在跑道的一瞬间，调整油门让发动机到最低转速，否则会错过跑道。不用考虑襟翼，貌似它们会在起落架放下时自动放出（参考 8.7.4 有关失速的讲解）。

对虚拟的波音 737 来说，140 ～ 150 节以及全部 8 档襟翼全部放出可以很好的降落。然而不要盲目相信我，因为我只有很少的经验，也没有搜索专业的数据。落地速度与载重相关，我猜 140 节是空载。波音 737 貌似需要柔和的拉平，拉平操作要更早[1]。

讲解塞斯纳 172 和 A-4 Skyhawk 的起飞流程时，我建议你将升降舵拉到其 $\frac{1}{2}$ 的位置。不过对于 Pilatus PC-7 并非如此，保持升降舵回中。让飞机加速到 100 节，然后柔和拉杆。降落开始对准跑道时，放出全部襟翼不要收油门。当越过跑道入口以后，再收油门。看起来 100 节是很好的降落速度。

塞斯纳 310 也是如此，跑道加速时最好保持升降舵回中。飞机会因为你放出的一档襟翼自己抬起机头。如果你一直保持驾驶盘后拉，机头会快速仰起而你也会面对可怕的俯仰问题。

很多虚拟的飞机，比如大型航线客机或快速飞机，需要快速的物理计算。然后需要在命令行加入 `--model-hz=480` 参数。如果发现飞机不太容易降落，可以尝试使用这个选项。

降落塞斯纳 172P 的时候，其实角度比喷气式的 2.5° 要陡。当然你可以在小角度落地，前提是跑道前方地形允许。或者作为乘客你的耳朵会对压力比较敏感……

8.13.4 如何起降 P-51D “野马”

你有机会驾驶一架 P-51 “野马” 吗？也许不会。其实此飞机的起降很危险。你需要很多训练，而当 P-51D “野马” 升入空中以后就没什么危险了。

[1]参考波音 737 手册，正常降落速度在 135 ～ 150 节之间，襟翼 15 度，而不是全部放出。在距地面 50 英尺左右开始拉平，并柔和收油门到慢车。——译者注

它很容易驾驶。

在中低高度，P-51 不会好过“喷火”和梅塞施密特战斗机。最大的不同在高空。当敌机窜入高空以后，P-51 可以保持效能和机动性。另一个不同，P-51 有很好的流线型外观，因此可以比“喷火”飞的更远。这两大不同，使得 P-51 可以为轰炸机远程护航，让轰炸变得更有效率也能有效打击纳粹。

要在 Linux 下启动 P-51D“野马”可以使用命令行参数 `-aircraft=p51d`

在 *FlightGear* 里起飞 P-51D，襟翼放出一档。将操纵杆完全向后拉，发动机推到最大，并按住鼠标左键控制方向舵，以保持在跑道上。一旦你达到 100 mph，突然将方向舵向右蹬全程的 1/3。快速松开鼠标左键并推杆以抬起尾轮（不要推太多，恰好将尾轮抬起即可）。现在保持鼠标左键放开，只在必要时对方向舵做调整。在大概 150 mph 的时候拉杆让飞机爬升。不要忘记收回起落架和襟翼。

转弯时不要太陡，你可能会失去控制并坠机。

要降落，放出全部襟翼并放下起落架。130 mph 看来很不错，最大 140 mph。从 1000 英尺开始进近并以低角度俯冲，就如同喷气机。当进入跑道以后，完全关闭发动机（按 **{** 键）。不要飞过跑道。尽快将机轮落在跑道上（就像喷气机）。使用方向舵让飞机沿跑道中线。当后轮落在跑道上时，柔和拉杆以便让机尾紧贴跑道。继续使用方向舵控制飞机，现在机尾已经在地面上了。按需使用刹车。

8.13.5 如何起降 B-52 “同温层堡垒”

FlightGear 里的 B-52F 非常成功，也是我最喜欢的飞机。B-52F 和塞斯纳 172P 的不同主要有：

- B-52 起动时需要设置停留刹车并放出襟翼。
- 襟翼只有两种状态：收起和放出。放出意味着机翼上更多升力，而不是减速。如果你想减速，需要使用绕流板，它们在机翼上方。按 **k** 键升起绕流板，按 **j** 键收起。绕流板有七块。
- 塞斯纳 172P 的主机轮有两个，飞机两侧一边一个。为了让机轮同时落地，你需要保持机翼与地面平行。B-52F 的主机轮有前后两套。而为了让所有这些主轮落地，你需要让机体与地面平行。

这是我的 B-52F 起飞流程：

- 将驾驶盘推到全部行程 $\frac{1}{3}$ 的位置。
- 发动机推到最大推力
- 松开停留刹车（按 **B** 键）
- 按住鼠标左键控制方向舵，以让飞机在跑道中线

- B-52F 需要全部跑道来起飞（KSFO）。
- 当 B-52F 离开地面，大概 190 节就可以很好的飞到一定高度了
- 收起襟翼和起落架。

要降落，B-52 的 HUD 已经提供了飞机形状的游标，正如我在上面喷气机那一节里说的。你只需要将那个图标对准飞机的中间。并保持低于理想水平线 2.5° 的位置。130 节到 140 节看起来很适合降落。保持迎角在 3°，我必须承认我倾向使用速度控制，而不是迎角控制。如果飞机在 130 ～ 140 节，只需让飞机“砸”在跑道上。否则如果速度太快，拉平以后飞机飘的距离会增加。刹车看起来很有效果（按 **b** 键）。它们可以让 B-52 像塞斯纳 172P 那样在跑道上快速停下。

飞行重放看起来是很明智的。可以用来检查机体降落时是不是与地面平行。还可以坐到乘客座椅上，来比较和航线飞行时你作为乘客的感受。按 **k** 键可以显示飞机在空中的虚拟航迹。

B-52 造成的事故可能因为：

- 转弯太陡，机翼几乎擦碰到地面.
- 让飞机恢复水平，飞机反应太慢。因此你会发现转弯会超过预期。
- 做了太极端的操作：蹬方向舵到极值，与当前转弯方向相反，这会让飞机瞬间从空中跌落。

第九章
短途转场飞行教程

9.1 简介

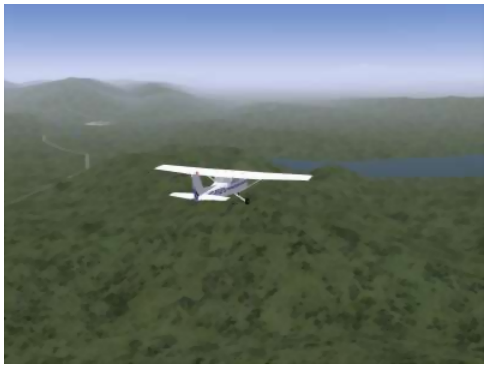


图 9.1: 从圣安东尼奥大坝飞到利弗莫尔

本教程将会模拟一次转场飞行,从里德—晓岚机场(Reid-Hillview,KRHV)到利弗莫尔机场 (Livermore, KLVK), 使用目视飞行规则 (Visual Flight Rules, VFR)。这两个机场都已经包括在了 FlightGear 的基础包里，因此不需要另外安装地景。

我假设你已经学会了 FlightGear 里的起飞、爬升、转弯、下降和降落。否则，建议你看一下之前的教程。本教程基于之前的教程基础，并增加了飞行系统和程序方面更复杂的信息。

9.1.1 免责和感谢

简短的说一句免责。在真实中我是飞 Microlights 的而不是塞斯纳 172。此文中的很多信息从大量非认证的源获取。如果你发现任何错误和误解，请联系我 stuart_d_buchanan -at- yahoo.co.uk。

我很想感谢如下这些朋友的帮助，使此教程更精确更深入浅出：Benno

Schulenberg, Sid Boyce, Vassilii Khachaturov, James Briggs。

9.2 飞行计划

飞行之前，我们需要先计划一下。否则我们起飞以后都不知道要向哪边转。

首先来看看这一地区的区域图。这是一张标示有机场、导航辅助设施及障碍物的地图。VFR 所用的区域图有两种比例尺，分别是 1:500,000 区域图和覆盖某些繁忙空域的 1:250,000 VFR 终端区域图。

都可以在飞行员商店买到，在线的也有很多，比如：

http://www.runwayfinder.com/[1]

只需要搜索 Reid-Hillview 就可以看到如图 9.2 这样的区域图了。

如果你需要包含整个地区并显示飞机具体地点的地图，你可以使用 Atlas。这是一个连接到 FlightGear 的可移动地图。详情可参考 6.3 一节。

那么我们如何从里德一晓岚机场到利弗莫尔机场呢?

我们要从 KRHV 的 31R 跑道起飞。KRHV 是 Reid-Hillview 里德一晓岚机场的 ICAO 四字代码，并可在 FlightGear 的向导里找到。

“31” 表示跑道的磁航向是 310 度，字母 “R” 表示使用右侧的跑道，从区域图上也可以看到，KRHV 有两条平行的跑道。多条跑道主要是用在交通繁忙的机场，这样每条跑道双向都可以起降。31 跑道从另一个方向看就是 13 跑道。因此可用的跑道就是 13R、13L、31R 和 31L。起飞降落都可以很容易处在顶风方向，因此当风从西北方吹来时，就可以使用跑道 31L 和 31R。跑道入口处会喷涂巨大的跑道号以供飞机在天上识别。

起飞以后，我们会转航向到 350 度直飞 Livermore 利弗莫尔机场（KLVK）。巡航高度在 3500 英尺，这可以让我们有 500 英尺余量来飞越途中的各种地形和障碍物，比如无线电天线。

我们也会飞过卡拉韦拉斯水库（Calaveras Reservoir）然后是圣安东尼奥水库（San Antonio Reservoir），两个都是巨大的水库，我们可以利用这两个大水库当作地标导航，确保在正确的路径上。

当我们距离利弗莫尔还有 10 英里的时候（飞越圣安东尼奥水库），我们会联系利弗莫尔的空中交通管制（ATC），获悉我们需要降落在哪里，然后加入机场起落航线并降落。

9.3 让我们开始吧

好了，现在我们已经准备停当了，该开始飞行了。

[1]此网站已关闭。可以使用 VFRmap.com 这个网站来查询 VFR / IFR 区域图和终端图，但只支持美国地区。——译者注

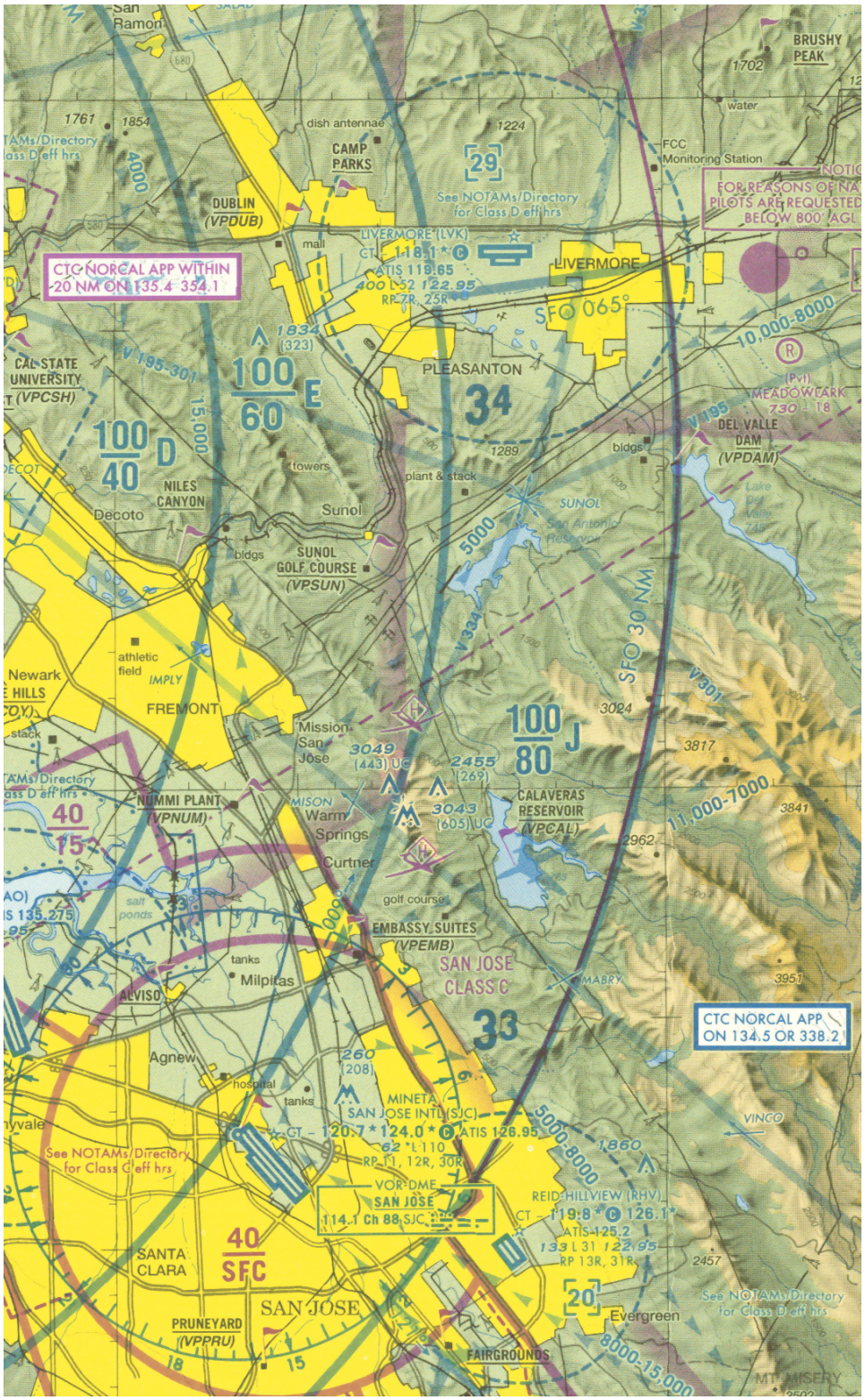


图 9.2: 一张包含里德—晓岚机场和利弗莫尔机场的区域图

使用向导启动 FlightGear（用命令行亦可）。我们用 C172P 从圣克拉拉县的里德—晓岚机场（KRHV）31R 跑道起飞。在佛罗里达飞行，选择黎明会很美。

如果你愿意，可以选择当前真实天气，只需要在向导里勾选“获取真实天气”选项即可。

图 9.3: 在 KRHV 的跑道上

9.3.1 飞行前的程序

起飞前，我们需要对飞机做起飞前的检查。在真实中，我们需要在飞机周围转一圈，以检查一切部件，并确保油量充足。

在这里，我们要检查天气、设置高度表拨正值并预设一些事情，飞行前来做会比较容易。

天气对飞行来说显然很重要。我们需要知道是否有侧风会影响起飞，以及云的高度（因为是 VFR 飞行，我们需要全程避开云层），任何风都会把我们推出航线。

我们还需要设置高度表拨正值。高度表通过测量当前气压来得知高度，然而天气会影响到气压值，所以高度表读数也会有误差，这在飞越山丘地带时会是致命的。

9.3.2 ATIS

方便的是，机场会通过 ATIS[1] 广播当前的修正海平面气压，以及有用的天气资讯和机场信息。ATIS 是一条预先录制好的无线电语音信息。为了收听 ATIS，我们需要首先调节无线电。

ATIS 的频率可以在区域图里找到（机场标识旁边的“ATIS”字样），同时也可以在 FlightGear 里找到。要查询机场的频率（包括塔台、地面和进

[1]中国大陆一般将 ATIS 称为“情报通播”，而标准翻译应为“终端区自动情报服务”。本手册中出现 ATIS 的部分，会根据语境，翻译成“情报通播”或保留 ATIS。——译者注

近），使用 ATC/AI 菜单并选择 “Frequencies”。然后输入 ICAO 代码（此例是 KRHV），然后机场相关的各种频率信息就会显示出来。如果有多个频率，使用其中之一即可。

好了，里德机场的 ATIS 频率是 125.2MHz。

9.3.3 无线电

现在我们该开始处理无线电了。所有无线电相关的都在主仪表板右侧的无线电栈里。实际上有两套独立的无线电系统，1 和 2。每一个无线电又被分成两部分，左侧的是负责无线电通讯的（COMM），右侧则是负责无线电导航的（NAV）。我们这里把 COMM1 的频率设为机场 ATIS 的频率。

图 9.4: C172 无线电通讯系统 COMM1 已高亮表示出来

无线电有两套频率。一个是活动频率，也就是正在使用的，另一个是备用频率，表示之后我们将会使用的。左侧的五个数字表示活动频率，而备用频率则在右侧。我们调节备用频率，然后将两者交换，这样备用频率就变成活动频率，相应的活动频率也就变成备用了。这么做可以防止在调节无线电频率时失去联系。

图 9.5: COMM1 调节旋钮

要改变频率，点击备用频率下面的灰色旋钮（也就是图 9.5 标出的部分），就在“STBY”旁边。左键点击可改变小数点后面的数字，鼠标中键点击可改变小数点前边的数字。点击左侧是增加，右侧是减少。FlightGear 里的大多数驾驶舱都是这样操作的。如果你操作有困难，也可以按 CTRL-C 来高亮可操作的区域。

图 9.6: COMM1 切换开关

当频率调到 125.2 之后，按“COMM”和“STBY”之间的那个按钮，来对调活动频率和备用频率（已在图 9.6 中标出）。很快你就可以听到 ATIS 的通播了。

9.3.4 高度表拨正值和罗盘

图 9.7: 高度表拨正值调节旋钮

仔细收听机场通播（ATIS）里有关“高度表拨正值（Altimeter）”相关的内容。如果你没有使用“真实天气”，此值一般是 2992，也就是标准大气压并且已经默认在飞机里设置好。如果你正在使用“真实天气”，高度表拨正值并不一样。因此我们需要将高度表拨正值设定到正确的数值，可以用图 9.7 里圈出的旋钮来调节，方法与之前调节无线电频率相近。高度表上的小窗会

显示当前的高度表拨正值，随着拨正值的变化（相当于基准气压改变）高度表的数值也会改变。

另一个方式是检查机场标高，也就是机场与海平面之间的高度。机场标高可在区域图里查到。KRHV 的机场标高是 133 英尺，因此我们可以通过这个数值来交叉检查我们的高度表拨正值。

图 9.8: 航向调节旋钮

趁此机会，我们也顺便把罗盘[1]上的航向游标设置在 350——也就是从 KRHV 到 KLVK 的航向。如图 9.8，调节罗盘下方的红色旋钮。方式和之前一样，用鼠标左键做小幅调整，用中键做大的改动。350 度只需要逆时针旋转到 N 之后第一个标线的位置即可（N 代表 0 度）

图 9.9: 从 KRHV 起飞

9.3.5 起飞

现在都搞定了，准备起飞吧。起飞以后，当高度超过 1000 英尺以后，温柔转向 350 度，正如之前我们航向游标设置的那样。现在我们正向着一块突

[1]此处应为方位陀螺仪，翻译时尊重原文并为改正，下同。——译者注

出的山谷飞去。

以 500-700 fpm（Feets per Minutes，每分钟升降速率）的上升率继续爬升到 3500 英尺。到达该高度以后，收油门并适当调整让飞机保持平飞。再次检查发动机动力，确保处于 RPM 转速表的绿色弧线区间。除起飞以外，不要让发动机处于最大转速[1]。

9.4 巡航

好了，我们已经起飞并前往利弗摩尔的路上了。现在我们可以用自动驾驶让自己轻松一些，顺便也可以节省一些燃油。同时我们需要检查自己是否已经在正确的航线上。

9.4.1 自动驾驶仪

图 9.10: C172 的自动驾驶仪

自动驾驶仪面板在无线电栈的底层（如图 9.10 所示）。其上有很多按钮可以很容易的找到。自动驾驶仪可以工作在多种模式，我们这次只关注其中之一——HDG（航向模式）。正如其名字一样，HDG 模式会让飞机飞向之前我们在罗盘上设定航向游标所指示的航向。

要接通自动驾驶，只需按 AP 按钮来打开自动驾驶，然后按 HDG 按钮激活航向模式。自动驾驶接通以后，会控制飞机调整片来调整飞机的航向。你可以修改航向游标，自动驾驶仪就会令飞机适时做出调整。然而自动驾驶仪并不会考虑风的影响，只会设定飞机的航向。如果是侧风飞行，飞机也许会指向某一点，但实际却偏差很大。

你需要使用调整片来保持水平飞行。也可以用自动驾驶仪来设置，不过会很复杂。

[1]根据真实中塞斯纳 172P 飞机的飞行手册，起飞和爬升阶段，油门都保持最大（FULL POWER）。爬升时可根据高度上升灵活调整油气混合比，以保持较高 RPM。到达巡航平飞高度时，收油门保持 21 ～ 26（x100 RPM）转速区间即可。——译者注

当飞机在自动驾驶仪的帮助下稳定下来以后，我们可以更多专注于外部世界，以及更高的技能。

9.4.2 导航

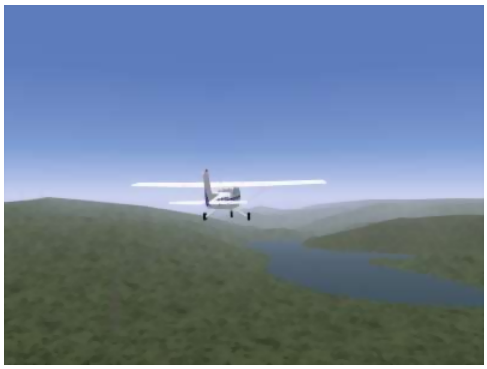


图 9.11: 飞临卡拉韦拉斯水库

正如之前所说，我们会途径几个水库。起飞以后，首先涌入眼帘的也许是右手边的卡拉韦拉斯水库。你可以通过地图和地标来检查。如果感觉偏离航线，可以直接改变航向游标来修正。

9.4.3 油气混合比

图 9.12: 卡拉韦拉斯水库

随着高度的增加，空气越来越稀薄，氧气越来越少。这也就意味着发动机需要燃烧的燃油就减少了。C172 的发动机比较简单，没有配备自动燃油调整，以适应和修正氧气减少的情况。结果就会造成大量燃油消耗而动力却并不增加，因为油气混合比太多了（富油）。我可以通过油气混合比控制杆控制油气的配比。油气混合控制杆是油门杆旁边红色的那个，将杆向外拉表

示“贫油”，即减小油气混合。我们不想让富油，也不想贫油，这种状态都不会有较高的动力输出。我们也不需要太完美，而只需要受控的燃烧即可，否则油气混合会产生爆炸，这很快就会把发动机报废。

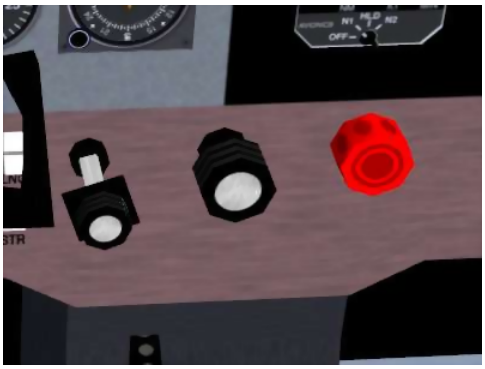


图 9.13: 混合比控制

用红色混合比杆来控制油气混合。你可能需要稍微移动一下视角才能看到它。

要移动视角，可以按住鼠标右键并移动鼠标。能看到油气混合比杆以后，就松开右键。

图 9.14: 燃油流量和排气温度（EGT）指示计

轻轻向后拉油气混比（可以用 CTRL-C 观察可操作的热点），减少混合比。这样做可以观察到发动机的很多指标有了变化（主仪表板左侧）。燃油流量会下降（因为烧了更少的燃油），EGT（排气温度，Exhaust Gas Temperature）会上升（因为更接近“完美混合比”）并且 RPM 转速也会增加（会增加更多动力）。向外拉混合比杆，直到 EGT 表针超过量程，此时稍稍推混合比杆。现在我们在最佳富油状态。3500 英尺高度并不需要太贫的油，更高的高度适度贫油对发动机性能至关重要。

9.5 下降阶段

当我们抵达第二个水库（圣安东尼奥水库）的时候，我们要开始计划如何下降并降落在利弗摩尔。降落比起飞更复杂一些，如果你想一次性完成，此时可以先读完，并将模拟器暂停（按“p”键）。

9.5.1 空中交通管制

在真实飞行中，我们需要一直保持与空中交通管制（ATC）的联通。正如湾区的地面交通一样，空中交通也很拥挤。ATC 会提供“飞行指引”服务，并会警告我们附近的飞机，防止在交汇时发生碰撞。FlightGear 的天空一般没有什么交通，因此我们也不需要飞行指引服务。如果你想修改空中交通的数量，可以从 AI 菜单找到。

利弗摩尔机场有塔台管制（塔台管制机场在区域图里以蓝色标出），因此我们需要联系塔台获取降落相关的信息。

在此之前，我们需要先收听机场情报通播（ATIS），并重新调整高度表拨正值，以防飞行过程中发生变化。短途飞行可能不会发生什么变化，但在数百英里的长途飞行中就有可能了。为了节省时间，可通过“设备”菜单的无线电设置对话框，输入相应的无线电频率。利弗摩尔机场的 ATIS 频率是 119.65 MHz。

ATIS 的信息以字母代码来标示，比如 Alpha（代表字母 A），Bravo（代表字母 B），……Zulu（代表字母 Z）。此字母代码会在 ATIS 录音信息更新的时候变化。当联系塔台的时候，你需要告知你收到的 ATIS 字母代码，这样塔台可以双向检查飞行员是否获悉最新的机场信息。

除了高度表拨正值和天气状况，ATIS 也会告知正在使用的跑道，可以帮助我们计划降落。正常情况下，因为强劲的西风，利弗摩尔通常使用 25L 和 25R 跑道。

当你获悉 ATIS 以后，就可以将频率调到利弗摩尔机场的塔台了，频率是 118.1 MHz。取决于你设置的 AI 交通情况，你可能会听到塔台与其他正在起降的飞机之间的通话。这些信息并不会通过扬声器放出来，只会看到屏幕上的字幕。

当频道里安静以后，按“’”键打开 ATC 菜单。在左侧无线电按钮上选择你打算对塔台说什么（目前只有一个选项），然后点确认。

你的通话将会在屏幕上方显示出来。通话信息会包括你是谁（机型以及尾号），你在哪（比如，机场以南 6 海里），正打算落地，以及你收到的 ATIS 编号。

数秒之后，利弗摩尔机场塔台就会回应，会以名字称呼你并告诉你用哪条跑道，以及加入哪条起落航线，并告知何时联系他们。例如：

“Golf Foxtrot Sierra, Livermore Tower, Report left downwind runway

two five left."[1]

为了理解这句话，需要讲解一下机场起落航线[2]。

9.5.2 起落航线

机场周边会有很多飞机，必须有起飞和降落的标准流程，否则一架飞机可能会落在正在起飞的飞机头上。

起落航线就是一个机场附近飞机必须遵守的标准航路，无论起飞还是降落。起落航线有四个阶段（或称四个“边”），如图 9.15 所示。上文所说的“Downwind”（下风边，第三边），就表示的是图里标号为 3 的那一边。

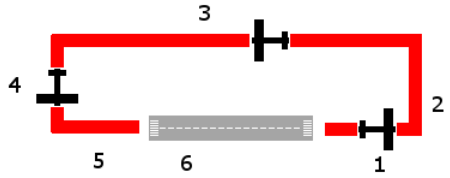


图 9.15: 起落航线

1. 飞机从跑道起飞和爬升时。如果正在离开机场，可以继续直线爬升直到脱离起落航线范围，之后做什么都行。如果要返回跑道（比如练习降落），就需要爬升到“起落航线高度”以下几百英尺的高度。此高度与地区差异有关，不过一般都是地面以上（AGL）500 到 1000 英尺。这就是所谓的离场边（*upwind*，第一边）。

2. 随后飞行员左转 90 度进入侧风边（*Crosswind*，第二边）。他们会继续爬升到“起落航线高度”。

3. 在侧风边飞了 45 秒到一分钟以后，飞行员再向左转 90 度进入下风边（*Downwind*，第三边）。此时很多从其他机场来此机场降落的飞机会加入，他们会以与跑道成 45 度角切入第三边。

4. 当飞过跑道尽头或一两英里（最好是跑道入口与你成 45 度角[3]），随后飞行员再向左转 90 度进入基线边（*Base*，第四边）并开始下降准备落地，适当放襟翼，下降率在 500 fpm 是比较合适的。

[1]这句话的意思是：GFS（表示飞机的呼号），利弗摩尔塔台，进入 25L 跑道左三边时报告。——译者注

[2]一般将 Traffic Pattern 翻译成“机场起落航线”（简称“起落航线“），或“五边航线”，此手册中出现类似情况都会按此翻译。——译者注

[3]简单方法是跑道

5. 45 秒之后飞行员转入最后进近（*Final*，第五边）。这个转弯可能会较难以很精确对准跑道，最后降落时还需调整。我经常需要做小幅转弯以对准跑道。

6. 飞机落地，若飞行员正在练习起飞和降落，可以推油门并收襟翼来起飞，飞机就再次起飞了。这称为“Touch and Go”。

大多数起落航线都是左手侧[1]，如上所说，所有转弯都是向左转。也有向右转的[2]。在区域航图上会以“RP”标注。ATC 也会告诉你用何种起落航线。

9.5.3 进近

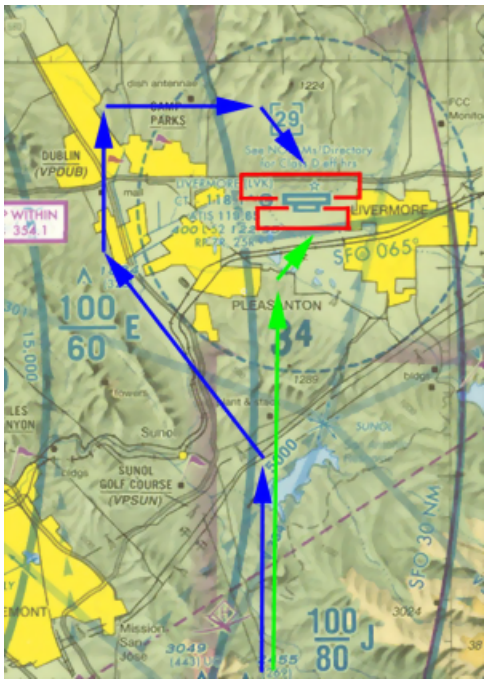


图 9.16: 从区域航图里摘抄出来的利弗摩尔进近航路

我们正在从南方接近利弗摩尔，因为盛行西风的关系，跑道是东西向的。我们需要使用 25L 或 25R 跑道落地，25R 用来做为右手侧起落航线的，而 25L 则是左手侧起落航线。两种起落航线都在图 9.16 里标出。取决于分配的跑道，我们需要用这两种起落航线之一。如果是用 25R 跑道落地，我们需要按上图中的蓝色线飞行；25L 跑道，则需要按绿色线飞行。

[1]因为飞行员坐在左侧，起落航线向左转有利于飞行员观察地形和其他飞机。——译者注

[2]若左侧起落航线会飞过人口稠密区，或其他不适宜飞机经过的地区，就会将起落航线设置为右手侧。——译者注

同时，我们需要降低高度，目标是到达起落航线时，大概距离地面（AGL）1000 英尺。利弗摩尔机场距离海平面（ASL）400 英尺，因此我们需要下降到 1400 英尺 ASL。

我们需要让下降操作恰好到达机场。否则可能会太高或者太快，或者甚至从错误的方向进入。不利于最佳降落:)。

所以，让我们立刻下降。

1. 首先，关掉自动驾驶仪。按其上的 AP 按钮。
2. 将混合比恢复最大（完全推入）。如果我们降落在一个高海拔机场，需要慢慢增大油气混合并在进入起落航线以后重新调整。
3. 开启汽化器加温（Carb-Heat），这可以防止油气混合进入汽缸前结冰，在潮湿空气中下降时会发生。汽化器加温操纵杆在油门杆和油气混合比杆的左侧，将其拉出就可以打开汽化器加温。
4. 慢慢降低飞机动力，否则我们会因为超速损坏机体。
5. 稍低机头，开始下降。
6. 使用调整片稳定飞机。

利用你和机场、普莱森顿（Pleasanton）和利弗摩尔（Livermore）两镇之间的相对位置来导航，进入如上所述的起落航线。

一旦切入第三边以后，你需要再次联系 ATC。方法与上文相同。他们会告诉你降落的顺序，“Number 1”（第一个）表示前方没有其他飞机降落，“Number 9” 表示你也许会想飞向另一个不太忙的机场！他们同时也会告诉你谁在你前方，及其位置。例如，“Number 2 for landing, follow the Cessna on short final” 表示你前方只有一架飞机，它正在最后进近。当他们落地并离开跑道以后，他们会报告 ATC，然后 ATC 会告诉你 “Number 1 for landing”。

9.5.4 VASI

当最后进近时，可以看到跑道左侧两排灯（如图 9.17）。这是 VASI[1] 提供了视觉辅助检查你进近时是否太高或太低。每一排灯光都可以是白色或红色之一。白色表示太高，红色表示太低，如果既有红色也有白色表示恰好在下滑道上。塞斯纳进近时为 60 节，下降率约 500 fpm 时比较好。如果太高，则稍微收油将下降率调整到 700 fpm。如果太低，增加发动机动力，减少下降率到 200 fpm。

[1]VASI 的全称是 Visual Approach Slope Indicator，目视进近下滑道指示器。另一种更加精确的则是 PAPI（Precision Approach Path Indicator，精确进近下滑道指示器）一般用于可起降喷气式飞机的大型机场，使用单排四灯的方式，两白两红表示恰好在下滑道上，三红一白表示略低，三白一红表示略高，四白表示太高，四红则表示太低很危险。——译者注

图 9.17: 利弗摩尔最后进近，可以看到左侧的 VASI

9.5.5 复飞

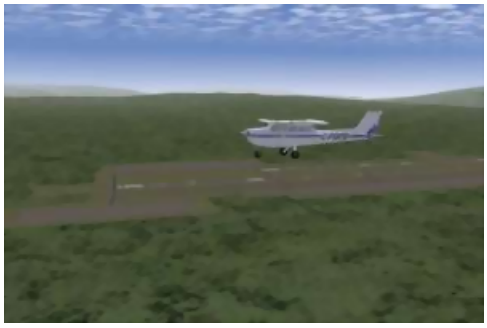


图 9.18: 在利弗摩尔机场复飞

因为各种原因，你可能无法完成降落时，你可以中断降落并重新尝试。这就是“复飞”。复飞的流程：

1. 将发动机动力推到最大
2. 等你飞机建立正上升率——比如通过高度表看到高度在增加了
3. 收襟翼到 10 度（第一档）
4. 告诉 ATC，你正在复飞
5. 爬升到起落航线高度
6. 如果你是在最后进近阶段复飞，继续飞过跑道，直到重新加入第二边（侧风边）。如果你在第四边时复飞，则继续转过第五边，然后沿着与跑道平行但与第三边相反的方向飞行，以重新加入第二边。
7. 之后完整飞过起落航线，当进入第三边时告诉 ATC，然后再尝试降落。

9.5.6 离开跑道

降落地面以后，你需要尽快滑离跑道，然后告诉 ATC 你已经离开跑道。在高海拔机场，你需要让发动机进入贫油状态，防止过多的燃油堵塞火花塞。找到一个好的位置将飞机停下，将油气混合杆完全拉出到贫油状态关闭发动机，然后将油门完全拉出并将磁电机转到 OFF 位。关闭航电系统主电门，将飞机系牢，然后去享受汉堡吧！

我希望此教程有所帮助。如果你还有一些问题，可以发邮件到 stuart_d_buchanan {at} yahoo.co.uk。

第十章
IFR 转场飞行教程

10.1 简介

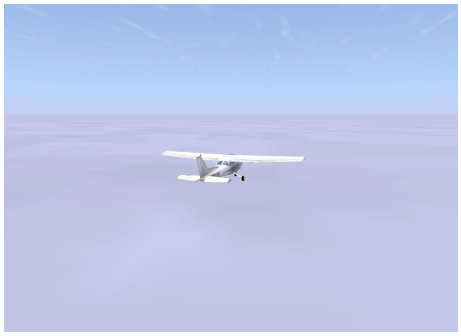


图 10.1: 正在从圣安东尼奥大坝飞往利弗莫尔，我猜如此

之前的转场飞行教程，你已经学会使用 VFR（目视飞行规则）飞行，而在此课程中你将更多学习用仪表飞行 C172P。现在我们将会做一次仪表飞行规则（IFR）下的飞行。此飞行中你将会使用剩下的仪表，学习一点 IFR 飞行，以及很多很多的三字母缩写。

我们将会做同样的飞行，从里德 - 晓岚机场（KRHV）的 31R 跑道飞往利弗摩尔机场（KLVK）的 25R 跑道，只不过这次我们使用 IFR 天气条件：云底高 200 英尺，能见度 800 米。此教程假设你已完成上面的转场飞行教程。

10.1.1 免责声明

此教程不会具体教授 IFR 飞行详情。而仅仅只是帮你建立 IFR 的直观印象，并将此转场飞行教程里没有用到的仪表细节删除。

我并不是一个飞行员，与之前的教程一样，这些信息大多通过非授权渠道获得。如果你发现错误或误解，请给我发邮件告知：bschack-flightgear -at-

usa -dot- net。

此教程已在 FlightGear 3.0 下测试过，其他版本可能与此稍有不同。

10.2 起飞之前

我们需要告诉 FlightGear 我们需要的飞行状况。有很多方式来设置我们需要的天气，这里我们使用全局天气。启动 FlightGear 以后，按 **Environment** ⇒ **Weather** 启动天气对话框，在 **Weather Conditions** 里选择 **CAT I minimum**。

这会降低云层和能见度。不幸的是，这也会造成较大的风。如果你不想面对这些，可以做一些修改：

- 重新进入 **Weather Conditions** 并选择 **Manual input**
- 将 METAR 字符串中的 “15015KT’’（150 ° 刮来的 15 节风）修改为 “15000KT’’（风速修改为 0 节）

最后按确认键关闭对话框并让 FlightGear 准备天气。

我发现，关闭 Atmospheric Light Scattering（大气光散射）可以让渲染更好。打开 **View** ⇒ **Rendering Options**，不要勾选 **Atmospheric Light Scattering**[1]。

10.2.1 飞行计划

如果你从窗户望出去，可能会看到如图 10.2 一样的景致。云层很厚，甚至无法看清跑道尽头。

当无法目视的时候，如何从 A 点到 B 点呢？多年来先驱已经总结了多种方法，各有优缺点。我们的教程将会使用塞斯纳 C172P 上所有的导航仪表，尝试各种可能性。

我们的整条航线，以及用到的导航设施都在图 10.3 里标示出来了。绿色是我们的航路，导航设施用蓝色和红色标出。航路有一点疯狂——也许你会想我们如果多使用这些设备而不是靠大脑方向感，就不会迷航了——方法很疯狂。如其在这里罗嗦一堆文字解释，不如下面让我慢慢道来。

[1] 早期版本的 Atmospheric Light Scattering（ALS）不够稳定，启用时会大幅降低帧速率。从 3.4 开始 ALS 有了大幅改善，几乎不会对帧速率产生什么影响，同时会给飞行带来更加真实的渲染体验。读者可自行决定是否开启。——译者注

图 10.2: 在 KRHV 的 31R 跑道

10.2.2 VHF 全向信标

首先我们使用 VOR[1]（VHF[2] Omnidirectional Range，甚高频全向信标）导航，它会将我们带到利弗摩尔南方 5 海里左右的位置。

区域航图里用翠绿色有罗盘标记的大圆圈住 VOR 信标台。我已经在图中用蓝色标记出来了。从图 10.3 里可以看到里德 - 晓岚机场与一个 VOR 信标台（San Jose[3]）很近，在圆圈的中心有一个翠绿色的方框，可以看到信标台的信息。根据这些信息，可以看到这是个 VOR-DME 信标台（稍后解释何为 DME），名字是 San Jose，频率是 114.1 MHz（或者 88 频道，同一个事情的另一种说法），其识别码或“ident”是 SJC（莫尔斯码是 ··· ·––– –·–·）

要调到 VOR 信标台，我们需要使用两台 NAV 接收机中的一台。也就是与 COMM 捆绑在一起的接收机（可参考图 10.4）。同时我们用对应的 VOR 仪表指针。在此例子中我们会选择 NAV1 接收机（对应的是 VOR1 仪表指针），当然也可以用 NAV2。调节频率之前，先来检查 VOR1 仪表。其应该如图 10.5 中左侧的 VOR1。最重要的是红色的“NAV”指示器。这表示没有收到 VOR 信号，因此我们不能相信此仪表。

---

[1]关于 VOR 的更多详情可参见维基百科 http://en.wikipedia.org/wiki/VHF_omnidirectional_range

[2]Very High Frequency，甚高频无线电

[3]San Jose 是美国地名圣何西，下文中的 Oakland 也是地名奥克兰。此手册中所有的 VOR、NDB、ILS 和定位点的名称，均保持原文不做翻译，以方便读者在航图中查询和理解，下同。——译者注

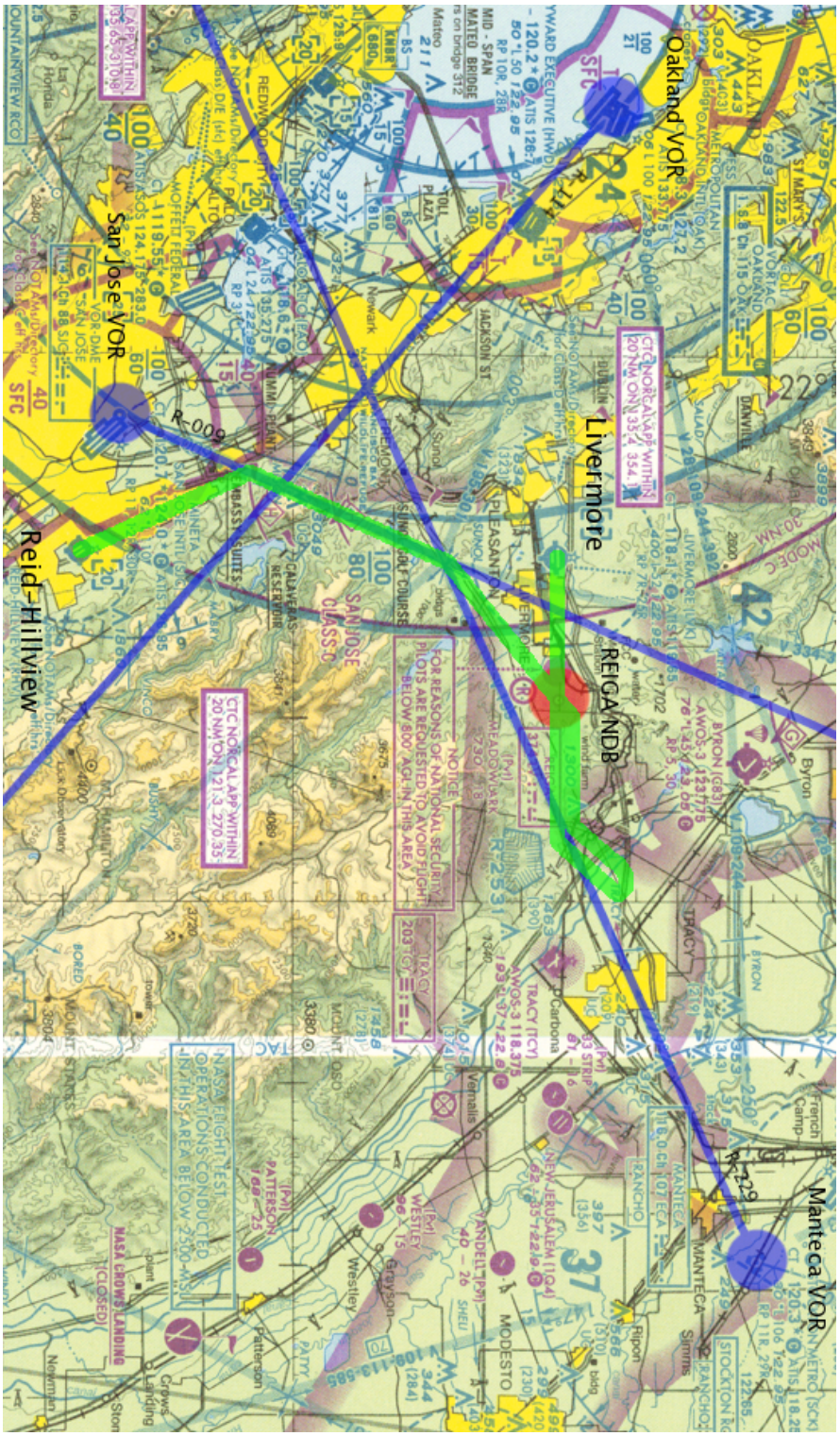


图 10.3: 绿色：我们的航路，蓝色：VOR 及其径向线，红色：NDB

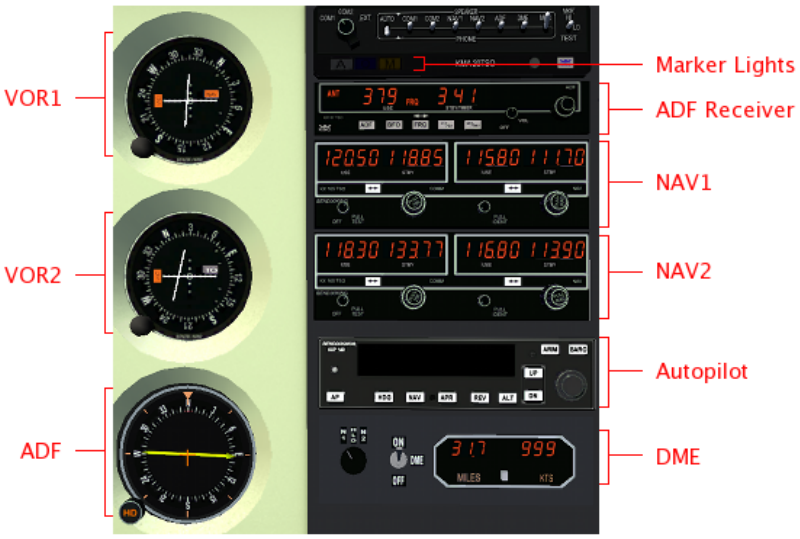


图 10.4: IFR 导航仪表

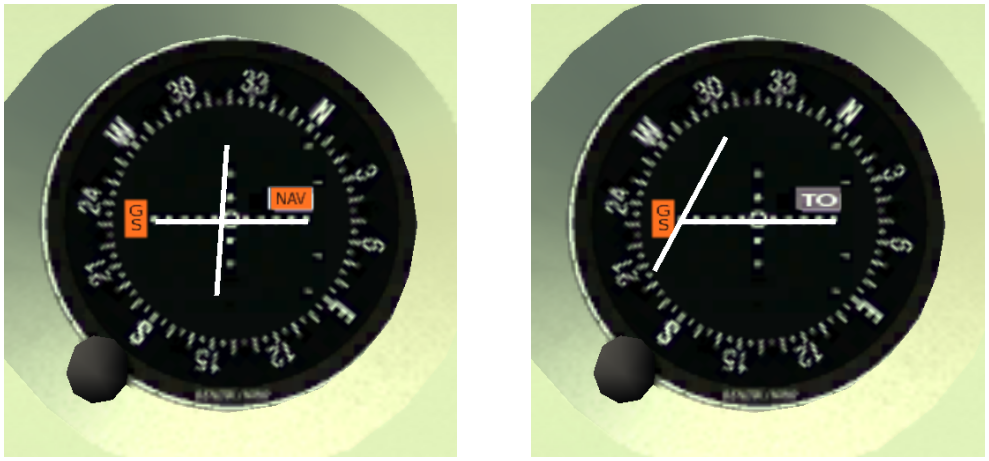


图 10.5: VOR1，调节频率之前和之后

NAV1 ⇒ 114.1[2]

NAV 接收机有一个活动频率和一个备用频率，还有一个调频旋钮，与 COMM 接收机类似[1]调节其频率到 114.1，并按 SWAP 交换按钮。如果你观察 VOR1，会注意到红色的 “NAV” 消失了，取而代之的是 “TO” 标识[3]，正如图 10.5 左侧所示。这表示我们收到了信号，然而这是不是正确呢？如果我们恰好设置了错误的频率呢？

为了确认我们调到了正确的 VOR 信标台，我们可以听其识别码。如果你听不到，可能是不符合航图，此时不要相信指针。目前为止，也许你还没有听到任何声音。为什么呢？检查音频面板（参考图 10.4）。你会注意到有很多开关控制这些设备，NAV1 也在其中。将开关向上拨（或向下——无所谓），你能听到莫尔斯码 ··· ·––– –·–·[4]。若已经厌倦了听莫尔斯码，可以将开关拨回中间。

回到 VOR1。左下方还有一个旋钮，叫做 OBS（OmniBearing Selector，航道选择旋钮）。正如名字所说，此旋钮用来选择航道。如果你旋转这个旋钮，你能看到垂直的指针在动，这个指针称之为 CDI（Course Deviation Indicator，航道偏离指示器）[5]。尝试让指针维持在中间，此时小三角应该指在 277 左右。这个数字配合 TO 指示表示：“正在 *277*° 航向上向台飞行”。

好了，根据我们的航路，我们不会这么向台飞行，我们会截获图上蓝色的那条 009° 径向线，背台（FROM）飞行。如何做呢？非常简单，设置 OBS

[1]操作 COMM 接收机的教程，已经在上一章的教程里介绍过。

[2]所有重要操作都会放在页边，避免与正文冗长的文字放在一起，更利于飞行参考

[3]VOR1 和 VOR2 的 TO/FROM 指示器（向/背台指示器），文中出现 “TO” 时会翻译为 “向台”，而出现 “FROM” 时会翻译为” 背台 “。——译者注

[4]若还不能听到，检查 NAV1 接收机的音量。若还不行，打开 **File ⇒ Sound Configuration** 并设置声音。检查你电脑的音量，外接音箱的音量。如果依旧不行，检查听觉

[5]水平的指针用于 ILS 降落，稍后会解释

到 009 。当穿过径向线时，指针会回中，同时指示器会变成 FROM。这意味着“正在沿 009° 航向背台飞行”，这正是我们想要的，因此在截获径向线时我们转向 009°。

VOR1 OBS ⇒ 009

起飞前最后一件事，设置方位陀螺仪的航向游标到当前航向，大概 310°。

.

航向游标 ⇒ 310

10.2.3 我们要飞多高?

设置好天气以后，气压值就不再是标准的“29.92”了。我们需要直到正确的高度表拨正值，否则高度表会设置错误高度。在起飞时也许不严重，但在云中下降时，却有巨大的不同（你也许会说这是“可控飞行撞地”[1]）。

正如上一章的短途转场飞行教程所说，我们可以通过 ATIS 来获取当前气压值。进入 **AI ⇒ ATC Services in Range** 并选择我们的起飞机场，搜索 ATIS 频率（应该是 125.20 MHz）。将 COMM1 或 COMM2 设置在这个频率（记得在音频面板上打开相应的声音），收听机场通播，并注意调节高度表拨正值。

我们会使用自动驾驶仪（如图 10.4 和 10.6）来稳定高度（稍后介绍）。因此自动驾驶也需要知道大气压。按自动驾驶仪上的 BARO 按钮，你会看到“29.92”显示出来了，这就是自动驾驶仪内部计算用的大气压值。在 29.92 消失之前（大概 3 秒），转动小旋钮改变其值。

10.3 起飞

好了，起飞以后暂时保持航向 310°，建立正上升率。我们计划爬升到 4000 英尺，现在只有一个问题比较棘手——我们面前这些丑陋的云在挡路。

起飞；沿跑道航向爬升

10.4 在空中

如果这是你第一次尝试 IFR 飞行，你也许会发现几乎不可能在云中飞行，因为缺少目视参考。也许你会说“没关系，只要保持稳定就可以”。有时候，你会发现指针疯狂旋转，飞机也会进入螺旋，上上下下或失速，也可能各种运动复合在一起。

我们需要练习在没有目视参考的情况下飞行，这是在学习 IFR 时必须掌握的技能。而现在，我们使用自动驾驶来让飞行简单一些。

10.4.1 自动驾驶 I

当我们建立正上升率并开始爬升时，按自动驾驶仪上的 AP 按钮启动自动驾驶。你会看到“ROL”显示在左侧，表示“横滚模式”——会保持机翼水

[1] CFIT，Controlled Flight Into Terrain

图 10.6: 启动之后的自动驾驶

图 10.7: 典型 IFR 场景

平，中间会显示“VS”，表示“垂直速率模式”——会保持恒定的升降速率。右侧会看到在 AP 启动瞬时显示的升降率。初始，此值就是自动驾驶启动时的升降速率。比如图 10.6，自动驾驶将垂直速率设为 300 fpm。

当你启动自动驾驶以后，请仔细确认此升降率。有时自动驾驶会进入很怪异的状态，比如 1800 fps。我们的小塞斯纳可无法承受，很可能会进入失速状态。这可能是一个 bug，因此为了保险起见，防止设备出问题。你需要不断交叉检查设备，并随时应对可能的问题。

我们需要垂直遂率在 500 到 700 fpm 之间。按 UP 和 DN 按钮，来调节垂直速率的值。同时还要考虑到空速，所以我们需要合适的爬升率。

启动自动驾驶；设置垂直速率；启用航向模式

最后，当爬升率建立以后，按航向按钮（HDG）。此时“ROL”会变成“HDG”，因为你刚刚已经将航向游标设置在了跑道航向，所以此时飞机不会有大的转弯。

10.4.2 MISON 交汇点

此时大概距离我们要截获的 009 径向线 8 海里的位置，因此我们还有一些时间。因为没有可观察的地面景物，如图 10.7，我们需要准备下一阶段飞行。

看一下我们的航路，截获 009 径向线并向北，我们会经过一个称为 MISON 的交汇点（如图 10.8 是更细致的图示，MISON 在右下角能找到）。在 MISON 上面可以看到交叉的十字，表示 MISON 是航路交汇点。实际上我们

图 10.8: Oakland VOR 及 114 径向线穿过 MISON 交汇点

会飞过 MISON 的东侧，但径向线是西北到东南的穿过 MISON，我们对此没兴趣。我们只是用其来监视飞行过程。

我们只需要沿着 009 径向线背向 San Jose 台飞行，直到我们需要转弯的时候。但这个交汇点很重要：第一，可以检查我们到哪了。第二确认我们想象的飞到哪里。如果我们一直飞阿飞，没有穿越径向线，就需要警告自己了。

研读区域航图，我们可以看到径向线是背向 Oakland VORTAC（VOR TACAN，TACAN 表示 Tactical Air Navigation，战术空中导航）114 度。Oakland 的频率是 116.8，其识别码是 OAK（--- •- -•-）。我们需要将 NAV2 设置在 Oakland，现在就设置吧。同时打开音频面板上的 NAV2 确认收到了正确的识别码。 NAV2 ⇒ 116.8

我们需要调节 OBS，告诉 VOR2 我们关注哪条径向线。设置 OBS 到 114 VOR2 OBS ⇒ 114

图 10.9: 高度保持模式进入预位状态时的自动驾驶仪

[1]。你可以猜测一下，当我们穿过 114 径向线的时候，VOR 上会指示 TO 还是 FROM。也可以猜测指针会从左向右运动，还是从右向左运动。

最后要说，依照我们的目的，无所谓是什么径向线，可以是 113，115，或者 100 或者 90，之所以选 114 是因为恰好图上标注出来了，这样就不用自己画了。

10.4.3 自动驾驶 II

在我们继续飞往截获 009 径向线的路上，让我们再近距离了解一下自动驾驶。首先，如果你没有使用调整片的习惯，你会看到自动驾驶仪上经常会有闪烁的“PT”标志。这是自动驾驶仪在提醒你用调整片调整一下飞机。我倾向于忽略这个，因为用鼠标飞行时，不调整反而更好。而那些使用游戏杆或驾驶盘的朋友，也许多多使用调整片会避免出现这个提醒。

同时，右侧还有一个大旋钮，也就是高度选择旋钮，可以用这个旋钮来调节目标高度。我们需要用到它，调节旋钮直到达到目标巡航高度，4000 英尺，显示在右侧。当你调节旋钮的时候，“ALT ARM”会显示在自动驾驶仪上（如图 10.9 所示），这表示你已经修改了目标高度。自动驾驶仪会保持当前上升率直到这个目标高度，之后会改平飞并从垂直速度模式（VS）变成高度保持模式（ALT）。在高度保持模式，它会保持在一个固定高度（比如此例中的 4000 英尺）[2]。当你距离目标高度还有 1000 英尺，也就是到达 3000 英尺时，会听到 5 声蜂鸣报警音表示高度保持模式进入预位状态。

设置自动驾驶仪目标高度 4000

注意，自动驾驶不会调节油门，因此当飞机改平飞时，速度会增加。你需要适时修正油门符合巡航需求。

10.4.4 保持航道

在某一时刻，你可能会截获到 009 径向线（也就是 VOR1 的 CDI 指针在中间位置）。此时转向 009 。若使用自动驾驶仪的话，你可以通过修改方向陀螺仪上的航向游标。

当 VOR1 截获后转向 009°

[1]如果你对手动调节频率感觉厌烦，可以用 **Equipment** ⇒ **Radio Settings** 对话框

[2]当然，你也不一定要如此，你可以叮着高度表，当到达 4000 英尺时，将上升率设置为 0，然后按 ALT 按钮进入高度保持模式。之所以多介绍一个旋钮，也是为了揭秘更多自动驾驶仪相关的内容

除非你做的足够好且幸运，否则指针也许不会回中。我们需要调整一下航道。CDI 指针（VOR 上的竖直指针）会告诉我们飞向哪里。如果它在左侧，表示径向线在我们的左侧，因此我们要向左飞。反之亦然[1]。

理论上很简单，然而在实践中你会发现很难保持指针一直在中间，会被风吹离航向。关键是：指针的*位置*告诉我们*在哪里*，而指针*运动的方向*告诉我们要*如何做*。

我来解释一下，如果指针在我们的左侧，那么无疑径向线也在左侧[2]。如果指针正在向我们*移动*，表示你正在或早或晚会穿过径向线，所以我们需要更快的进入径向线或者等待指针回中。另一方面，如果指针正在*远离*我们，我们需要阻止这种趋势，逆转它的运动。

注意，刚开始我们很难猜测需要转多少。可先尝试转 10°，如果指针运动太快，我们就改成 5°（或反向 5°）。如果另一种情况，指针移动太慢，我们就加大到 20°（另外增加 10°），然后看看会发生什么。

10.4.5 更多的交叉检查

经常交叉检查飞机的位置是个好习惯。与 Oakland 114 径向线的交汇点就要到了。前方就是 SUNOL 交汇点。如果你仔细看，5 条径向线交汇在此，因此我们要选择使用哪条径向线。因为之后会非常有用，我们将会使用右上的那条，它是 Mantech VORTAC 的 229 径向线，频率 116.0 MHz，识别码 ECA（· −·−· ·−）。

穿过 OAK 114 径向线

你应该已经知道该如何做了：调节 NVA2 到 116.0，设置 OBS 到 229，并检查确认信标台识别码。

NAV2 ⇒ 116.0

同时，让我们介绍仪表板上的另一个功能，用来交叉检查 SUNOL 交汇点。一些 VOR 信标台有测距的功能，也就是 DME[3]（Distance Measuring Equipment，测距仪）。比如 San Jose 就有（还记得它是 VOR-DME 信标台），而 Oakland 和 Manteca 也有（VORTAC 具备 DME 的能力）。

VOR2 OBS ⇒ 229

使用 DME，你可以知道我们飞了多远，距 VOR 信标台的直线距离。在我们的场景中，DME 不重要，但我们依旧会使用，顺便学习它怎么工作，也确认我们的位置。

DME 仪表就在自动驾驶的下方[4]（可参考图 10.4）。确保已经打开它，左边的开关旋钮一般是在 N1 位，而“N1”则表示“收听 NAV1”。因为现在 NAV1 是 San Jose 频率，这样就告诉我们距 San Jose VOR-DME 信标台的距离。我们把旋钮转到 N2，这样会告诉我们距 Manteca VOR 的距离。

DME ⇒ N2

DME 会显示三样东西：距离信标台的海里数，飞向或远离信标台运动的速度，以及按此速度飞向信标台所需时间。注意此处的距离是飞机到信标台的直线距离（也就是所谓“斜距”），而不是地面距离。同样速度也是相对

[1]原文这里缺少一个前提条件，当飞行方向与 VOR 的向背台指示器一致时符合这个规律，如向台飞行且 VOR 也指示为 TO 的时候。而当向背台指示与实际飞行方向相反时，指针和飞机的运动是相反的，即指针向左偏转时，我们需要向右飞行以截获径向线。读者可在 Flight-

信标台的速度，因此除非你是向台或背台直飞，否则这个速度会略低于你真实的地速。比如，从 San Jose 背台飞行，它就在我们后方，会稍快于我们向 Menteca 信标台的速度，因为它在我们右侧。

如果我们要搜索有关 SUNOL 交汇点的信息[1]，会告诉我们它在 ECA 信标台 229.00 径向线 33.35 海里的位置（也就是 "ECAr229.00/33.35'' 表示的含义）。

现在我们有两种方法确认 SUNOL 交汇点：VOR2 指针会回中，DME 的读数是 33.4 左右。注意，DME 不会提供精确修正，因为 Manteca 信标台的测距有个角度[2]。

你也许会问 DME 上 HLD 的含义（也就是 N1 和 N2 之间）。这表示 "Hold"（保持），意思是 "不管 NAV1 和 NAV2 如何调节，DME 都保持当前频率"。比如如果我们从 N2 调到 HLD，DME 依旧会显示我们到 Manteca 的距离，即使我们重新调节 NAV2，DME 始终会保持在 Manteca。这非常有用，相当于 DME 变成了第三台独立的接收机，IFR 飞行时两台接收机看上去不够用。

10.5 准备下降

我们正在靠近 SUNOL，沿着 San Jose 的 009 径向线，关注着 DME 上的位置。在 SUNOL 点，我们距离利弗摩尔就不到 5 海里了，也就是云层下方的某处。也许我们降到 700 英尺左右（利弗摩尔标高 400 英尺，而云底高 750 英尺）并稍稍向北就能到了吧？我的朋友也许一场灾难即将降临，你会懂的。

10.5.1 仪表进近程序

正如上一章的教程所述，当我们飞 VFR 的时候，你不能让飞机直接向最近的跑道落地。你需要飞到起落航线中，这会帮你对准跑道，并避免与其他飞机相撞，这是一件好事。

IFR 与此类似，也有流程可循。事实上，确实有程序[3]可用。因为 IFR 降落的复杂性，没有一个适用于所有机场的固定程序。你需要根据特定的机场

Gear 里亲自实验并领会。——译者注

[2]除非你向相反方向飞行，那就是另一种情况了。

[3]详情可参考http://en.wikipedia.org/wiki/Distance_Measuring_Equipment。

[4]有些是在自动驾驶仪的上方。——译者注

[1]比如使用 http://www.airnav.com/airspace/fix/SUNOL.

[2]因为这个角度很小，所以常常将 DME 测得的距离等同于地面距离。吐个槽，原文这里最好有个图。——译者注

[3]此处的 "程序" 是为 "Procedure" 一词的中文翻译，在民航领域中，一般带有流程性的内容称为 "程序"，不要与计算机相关的 "程序" 混淆。——译者注

检查相应的程序。事实上，你需要检查特定机场、跑道和导航设备。

我们的机场是利弗摩尔（Livermore，KLVK）。让我们搜索它的信息。前往 http://www.airnav.com/airport/KLVK[1]。在靠近底部的位置，我们可以看到 IAPs（Instrument Approach Procedures，仪表进近程序）。表里有两个 25R 跑道的仪表进近程序，一个是 ILS（Instrument Landing System，仪表着陆系统）进近，另一个是 GPS（Global Positioning System，全球定位系统）进近。因为我们没有安装 GPS 设备，但我们可以 ILS 进近（稍后介绍），因此我们就选这个了。

虽然利弗摩尔只有两个仪表进近程序，而大型机场则有很多。如果看一下附近的旧金山机场，你可以看到大量的程序，有 ILS 程序、GPS 程序、LDA 程序、VOR 程序等等，学习 IFR 飞行，你会学到所有这些程序。

回到利弗摩尔，如果你已经下载了进近航图，你会看到如图 10.10 这样的内容（原图是黑白，这里涂色帮助理解）。最开始会感觉一头雾水——因为在一张纸上写了太多的信息。我们会忽略很多，只集中在用颜色标出的这三个部分。这些部分称为“需要知道的”——也就是我们只在需要知道的时候，才去看。

从哪里开始？首先，一个 IAP 会有一个或多个起始进近定位点（Initial Approach Fixes，IAF）。这是你进入进近程序的入口点，可以在航图的“Plan View”（平面图）里找到，也就是在图 10.10 中用紫色标出的部分。我们的 IAP 有两个，一个是在中间，另一个是右侧（如图 10.11 的放大）。

一个 IAF 就是一个定位点（*Fix*），定位点是空间中的一个虚拟的点。实际上我们已经接触了另一种类型的定位点，比如 VOR 的交汇点。定位点通常也有名字（比如 MISON，SUNOL）。右侧的 IAF 称为 TRACY，由一条径向线，一个距离标识和一个高度标识合并组成。具体来说，它位于沿着 ECA（也就是 Manteca）VOR 信标台 229 度径向线，DME 15 海里（用 DME 接收机测量的 15 海里）的位置。

10.5.2 无方向性信标台

当然，我们并不使用 TRACY 作为此次飞行的 IAF，我们会使用中间的 IAF，也就是一个指点（图中的 LOM 表示“Locator Outer Marker”，外指点标），稍后再关注外指点标，目前我们集中在此定位点上。而这个外指点标，也是一个 NDB（Non-directional Beacon）信标台[2]。它与 VOR 相似，用它可以辅助航向，并从一个地方导航到另一个地方。和 VOR 一样，它也有名字（比如此例中的 REIGA），一个频率（375 kHz），一个识别码（LV，或莫尔斯码·–·· ···–）。NDB 也会在区域航图上标出，一个模糊的红色圆形，中间有很多小点点，旁边红色方框里是它的相关信息（如图 10.12。不要与旁边红色实心圆混淆，或者那个中间有字母“R”的圆形）。

[1]此网站只提供美国境内的机场和导航设备信息。——译者注

[2]详情参考 http://en.wikipedia.org/wiki/Non-directional_beacon。

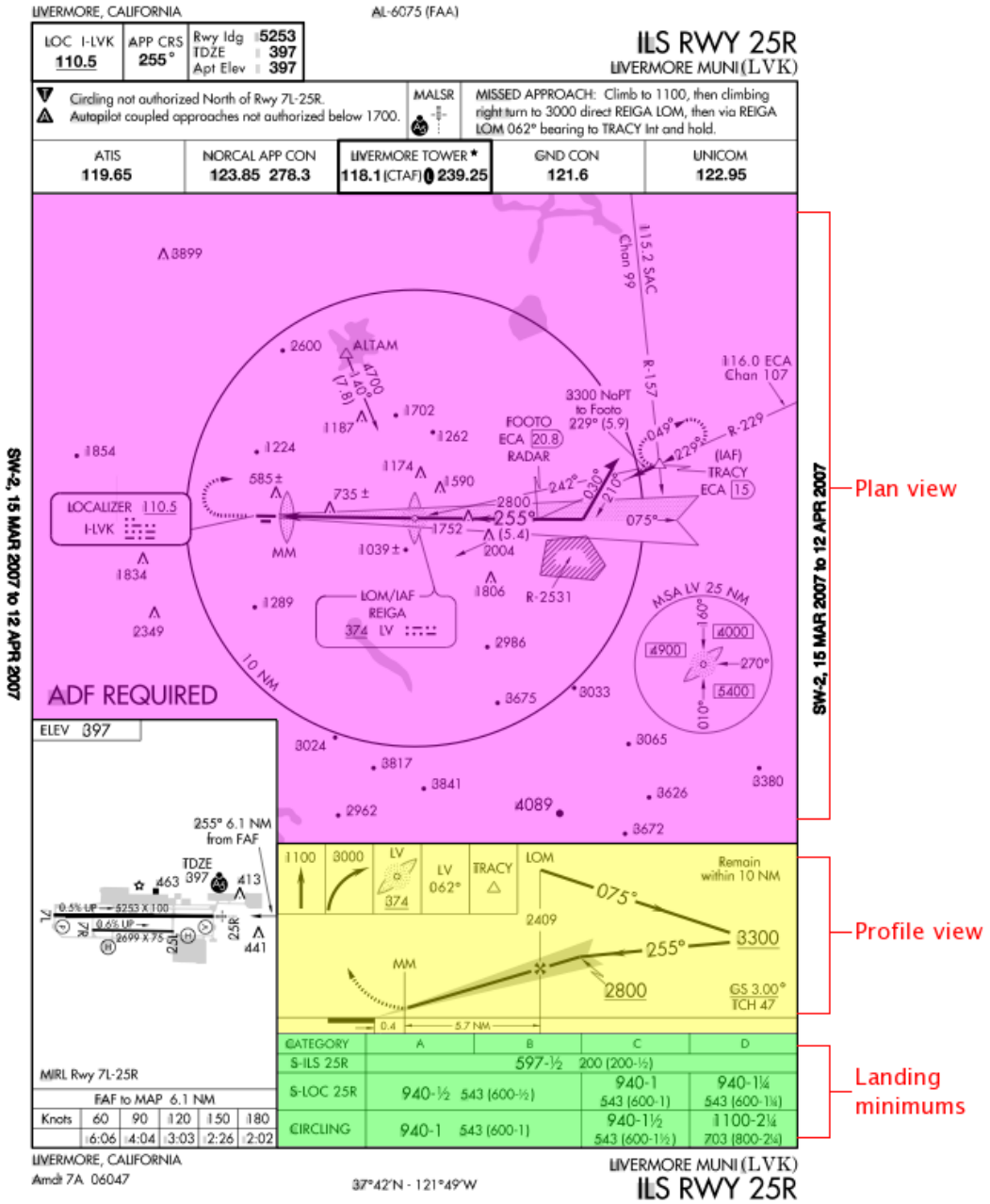


图 10.10: 利弗摩尔 25R 跑道 ILS 进近示意图

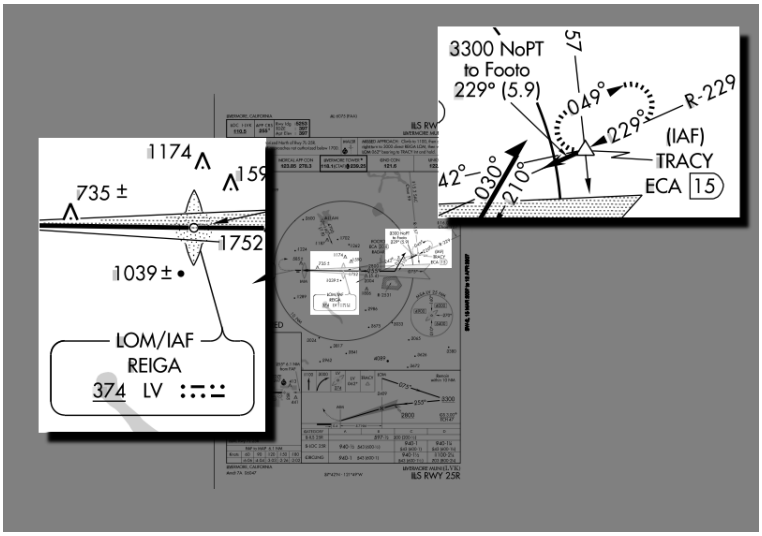


图 10.11: 起始进近点 IAF

图 10.12: REIGA 无方向性信标台

NDB 信标台会广播一个信号像在说“我在这里”，而飞机里的接收机会接收到这个信号并告诉飞行员，“信标台就在那里”。你只需要调节接收机频率并监视住正确的仪表。接收机，也就是 ADF（Automatic Direction Finder，自动定向设备）接收机，而相应的仪表也是 ADF，如图 10.4 所示。

ADF ⇒ 374

要调节到 REIGA，转动备用频率（STDBY）旋钮到 374。和之前一样，鼠标中键做大幅修改（这里每次修改 100 kHz），用鼠标左键做小幅修改（1 kHz）。然后交换活动频率和备用频率（按“FRQ“按钮）。这样 374 就是选定的（SEL）频率。ADF 指针也会摆动，可能会指向右侧，也可能直指 REIGA。也许不会，为什么？因为接收机可能会处于天线模式（左上角会显示“ANT”）[1]。如果在天线模式，按 ADF 按钮，会看到显示为“ADF”。这样指针会摆向 REIGA 信标台。与 VOR 一样，为了确保我们收听到了正确的频率，需要识别信标台，因此我们将 ADF 的音频开关打开。

注意，ADF 没有 OBS，因为 ADF 的指针是指向信标台的，这太好了。这就推导出了第一个 ADF 原则：

ADF 原则 1： 指针指向信标台的方向

非常简单，事实上你也许认为这不能算是一个“原则”，但是这却是 ADF 和 VOR 的巨大差别。一台 VOR，还记得吧，追踪单一的无线电径向线，也就是你通过旋转 OBS 旋钮选择的。而一台 ADF 也有一个旋钮，也有一个一模一样的罗盘卡，也许会认为表示相同的含义。并不是，旋转这个 ADF 航向旋钮（有“HD”标签的）看看会发生什么吧。罗盘卡在移动，而指针没动。因为它永远指向信标台的方向。

我们现在的状况下，无论从何处飞向 REIGA，只要我们用 ADF，指针指向“就在那里”，我们就飞到“那里”，并且会飞过 REIGA 上空。然而，因为缺乏实践，又因很快就要用，我们来解释罗盘卡的用途：

ADF 原则 2： 如果罗盘卡指向当前航向，那么指针就表示我们飞向信标台的航向。

也就是说，罗盘卡给出飞到“那里”的具体数值。

好了，这样我们就准备航向 REIGA 了。旋转 ADF 航向旋钮直到其顶部航向与当前航向一致（一般的，ADF 航向需要与方位陀螺仪航向一致）。当我们经过 SUNOL 交汇点，注意看 ADF 指针，设置方位陀螺仪的游标到那个航向（我假设你在使用自动驾驶仪，如果没有，那么只要航向那里即可）。转弯结束以后，ADF 的指针应该指向正前方。如果没有，那么就调整航向。[2]

经过 SUNOL；转向 REIGA

顺便说，距离 REIGA 越近，指针就会变得越敏感，不要疯狂的让指针保持中间。只要稳定在一个航向就行，然后准备……

[1] 天线模式，主要用来识别 NDB 信标台，因为它有较好的音频接收效果。在天线模式，ADF 指针不会指向信标台，而是永远指向右侧。

[2] 在侧风时这样做并不好，但此处忽略风的影响。

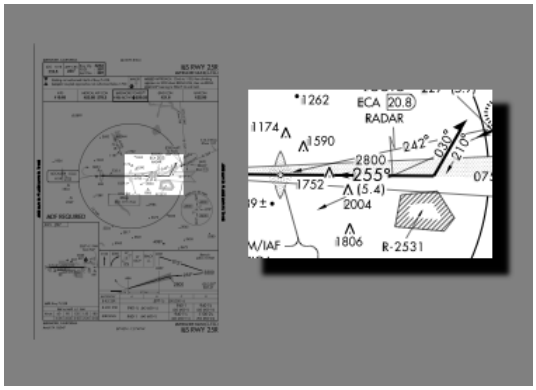


图 10.13: 利弗摩尔 ILS 程序转弯

10.5.3 程序转弯

好了，我们飞过 REIGA，是不是直接向左转并对准跑道了？哈，如果真是这么简单就好了。我们需要向右转，飞离机场，并做一个程序转弯。如图 10.13 中的平面图，加粗的箭头标示了一个程序转弯。如果你按箭头所示，需要飞离机场航向 075°，然后左转 45° 到 030 °。我们需要做了一个 U 字形的转弯（向右转飞离机场。——这也是程序转弯的一条原则）回到 210°，之后再右转 45° 到 255°，对准跑道。这么做可以让我们有时间修正航向道，到达正确的高度，为降落 25R 做准备。

我所说的 “正确的高度”，我们如何知道呢？在图 10.10 中黄色标注的剖面图（Profile View）部分，你可以注意到上面是 LOM，也就是我们的 IAF。跟着箭头，IAF 之后我们会航向 075°。在程序转弯中我们下降到 3300 英尺，然而不能低于这个高度（这就是 3300 数字下划线的含义）。程序转弯之后航向 255°，我们可以下降到 2800 英尺，但不能低于这个高度，直到我们截获下滑道。

仪表进近程序并没有告诉你程序转弯的长度。唯一要求是不能飞离 NDB 超过 10 海里。你会注意到平面图里有一个 10 海里的大圆圈。这表示 “保持 10 海里以内”。这不是开玩笑。因此假设我们速度 110 节，那么每一边飞两分钟就差不多了——也就是在 075° 飞两分钟，然后转向 030° 以后再飞两分钟。再往后，我们就不需要关心时间了，我们只需截获 255°[1]。

所以，飞过 REIGA 之后，向右转到 075°。我们的 ADF 有内置定时器，这样我们可以用这个来计时两分钟。按 “FLT/ET”（flight time/elapsed time）按钮。中间的 “FRQ” 会消失，“FLT” 会显示在右边，备用频率会被定时器替代。现在表示全部飞行时间，不能修改，除非重新换电池。再次按 “FLT/ET”

[1]关于利弗摩尔机场，REIGA NDB 10 海里限制，在 2011 年 6 月新修订版航图里已经取消。——译者注

图 10.14: 定时器运行时的 ADF

按钮，你会看到“ET”显示出来了，还有一个时间，可能会与飞行时间相同，要重置此预计时间，按旁边的“SET/RST”按钮。这样定时器会归零，然后开始计时（见图 10.14）[1]。在计时模式，当你按“SET/RST”按钮，计时器都会归零。若此时想看备用频率，按一次“FRQ”按钮，计时器持续计时。

10.5.4 追着指针

飞越 REIGA；从 REIGA 的 075° 航道飞离 REIGA 两分钟

当我们到达 REIGA，之前我们并没有特别关注过我们的航道——我们只要对准 REIGA 就行了。而现在航道就对我们非常重要了。我们需要从 REIGA 的 075° 航道飞离。

在理想情况下，当我们转向 075°，ADF 指针会直接指向后方（假设我们在航道上）。实际并不是如此，我们需要调整我们的航道，调整的方法遵循 ADF 的第二个原则。如果我们正确设置了罗盘卡，指针会显示当前 NDB 所在相位角，如果我们转弯然后直飞直到截获 255 相位角，然后转到 075°，这样我们就在航道上了。

图 10.15 表达了我的意思。图中飞机沿绿线飞行，起初是在航道之外[2]。航向是对的 075°，然而信标台却在 225° 的位置。而不是 255°。要修正我们需要右转（记得随时调节 ADF 的罗盘卡）。当我们飞到这个新的航向时，离正确位置越来越近，穿过 235 和 245 相位角（如图红色）。最终当我们的 ADF 指针指向 255° 时，我们转向 075°。并重新调整 ADF 的罗盘卡[3]。这样我们就在正确的航道上了。

当然，当我们回到正轨，这一切并没有结束。你的飞机还在移动，你的思维在游离，你的罗盘也在改变，风会让这一切发生变化。后面你需要不断做出修正。现在还不错，当我们越来越近接近程序转弯的起始点（差不多飞 2 分钟），我们向左转 45° 到 030° 开始程序转弯，之后就可以忽略 NDB 了。

10.5.5 FOOTO 时间

在你向机场外飞行时，有机会来看一下 VOR2，此时已经调节到了 Manteca 信标台，还有 DME。假设 OBS 还是 229，而 DME 还是 N2 的话，在

[1] 定时器也可以设置成倒数计时模式，这里并没有实现。

[2] 实际是偏离航道的，图里的角度略夸张，更利于解释

[3] 你也许会想有没有可能 ADF 的罗盘卡自动旋转？确实有这样的 ADF，这就是 RMI（Radio Magnetic Indicator，无线电磁航向指示器）

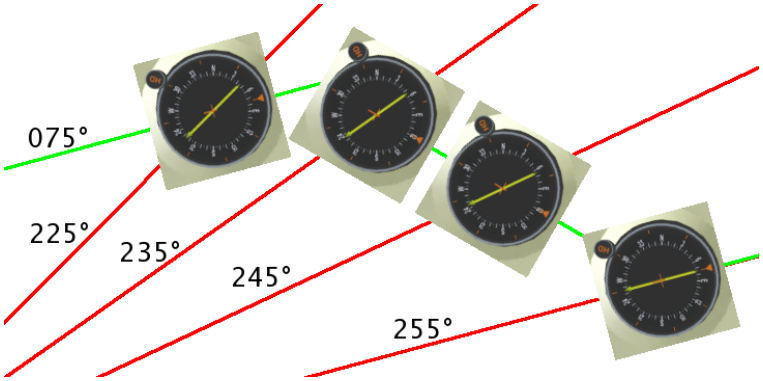


图 10.15: 回到航道

某点指针会回中，表示你正在穿越 229 径向线，如果你依旧在航道上，同时 DME 的读数应该是 20.8。我是怎么知道的？看一下进近航图（见图 10.10），你会注意到一个交汇点，标为 FOOTO。FOOTO 在进近路上，而它相对 ECA 是 20.8 DME。虽然这个交汇点对我们没什么意义，就当免费送的，这可以帮我们离场，还有一会进场时，提供很好的确认素材。

取决于你飞行的速度，你经过 FOOTO 时应该已经很接近 075° 的两分钟了。当计时结束时，我们向左转 45° 到 030°。重置计时器，保持 030 再飞行两分钟。

10.5.6 自动驾驶 III

左转到 030°；飞行两分钟同时下降到 3300

这段航程比较波澜不惊，所以利用这个优势，我们下降到 3300 英尺。下降前，检查 KLVK 的 ATIS （频率是 119.65 MHz）并确保你的高度表拨正值是正确的。

假设你在使用自动驾驶，下降时需要做这么几件事：

1. 如果使用高度保持模式（ALT），需要返回垂直速率模式（VS）。按 ALT 按钮，中间显示的 “ALT” 会被 “VS” 替代，而你当前的下降率（可能是 0）应该显示在右侧。

2. 按 DN 按钮，设置垂直速率 -500 fpm

3. 如果你还想设置目标高度，和之前一样，旋转右侧大旋钮让右侧显示到 “3300”。“ALT ARM” 也会在下方显示出来。

注意使用自动驾驶仪下降时，它只会将机头向下，就像一个不好的飞行员，因此飞机速度会增加。而我们想下降而不是加速，因此我们需要将发动机转速降低以保持 110 节的速度。之后，当你在 3300 英尺平飞时，还需要增加发动机动力。

如果你在手动飞行，需要调整发动机动力到所需的下降率。如果之前已经调整到了 110 节，飞机应该会稳定在 110 节。

10.5.7 ILS 着陆

下降时，我们也需要思考如何截获 255° 径向线并对准跑道。你也许会想，我们像刚才那样使用 NDB，然而 NDB 已经不能满足需求。我们需要“精确”落地，因此就需要用 ILS 着陆，NDB 不够精确，它可以引导我们靠近跑道，但无法靠的更近。

所以我们现在要切换到 ILS 系统。ILS 提供了更精确的水平方向定位。同时也提供垂直引导，而这也是像 NDB 这种所缺乏的。另外也利用这几分钟时间，学习一些新东西。

与 NDB 和 VOR 导航一样，ILS 系统[1]也有一个地面信号发射机（包括一个航向信标和一个下滑道信标），飞机上也有相应的接收机和仪表。接收机说白了就是我们两个 NAV 接收机中的其中之一。仪表与 VOR 指示器相似只是包括了下滑道指示器，也就是一个水平的指针。与 VOR 相似垂直的指针表示你是不是偏左或偏右。而水平的指针表示你是否太高或太低，而我们的 ILS 仪表就是我们的老朋友 VOR1[2]。

也许你也已经猜到了，航向道信标也有一个频率和识别码（没有必要单独为下滑道信标设立频率，当调节到航向道信标台时也就同时具备下滑道了）。在进近航图里有两个地方标明了频率：航图最上面的左上角，另一个是平面图的跑道头位置（见图 10.16）。我们可以看到，频率是 110.5 MHz，识别码是 I-LVK[3]（·· ·–·· ···– –·–）。

NAV1 ⇒ 110.5

现在来看一下 VOR1，会发现红色的“GS”指示（如图 10.5）。这表示没有收到下滑道信号。现在将 NAV1 的频率转到 110.5。红色的“GS”指示会消失。现在检查识别码，听到了可爱的声音。当你调节到航向道信标台以后，你同时会注意到 ILS 指针在移动。那么 OBS 呢？它就没什么用处了。可以尝试转动旋钮，无论你怎么转，指针都不会做出反应。这就是这么设计的。可以将航向道信标台看作是只有一条径向线（也就是进近航向）的 VOR 信标台。此时我们也不考虑其他航向了，因此也就不需要 OBS。不过为了提醒，还是将 OBS 转到 255，我们需要的航向。

VOR1 OBS ⇒ 255

10.5.8 截获航向道

右转 180° 到 210° 截获航向道

现在准备截获 ILS 航向道了。之前当我们飞过 030° 两分钟以后，向右做 U 字形转弯到 210°。当你完成转弯以后，ILS 的垂直指针（航向道）会开始移动。而且会移动很快，会比 ADF 和 VOR 指针移动更快。航向道信标比

[1]详情可参考 http://en.wikipedia.org/wiki/Instrument_Landing_System

[2]塞斯纳 172P 飞机的 VOR1 和 VOR2 都具备 ILS 能力，——译者注

[3]所有 ILS 信标台的识别码都是字母“I”开头，其莫尔斯码是非常容易辨识的 ·· ，这样可以帮助飞行员非常容易的获知是否收到了 ILS 无线电信号。——译者注

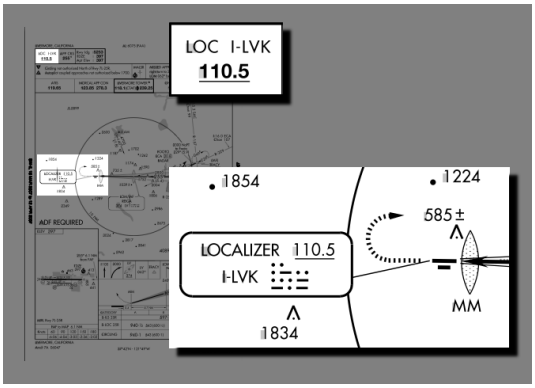


图 10.16: 利弗摩尔 25R 航向道信标

VOR 敏感 4 倍，飞机的小幅改变就会让指针发生大幅偏转。因此你可能会过度，然而不用着急，我们还有 5 到 10 分钟时间来对正。

只要记得：不要追着指针。这句口头禅比任何时候都重要。因为指针很敏感，如果你追着指针向左你也向左，指针向右你也向右，你就会飞得像个喝醉的水手。如果足够幸运，在你左摇右晃的时候，跑道可能就在你下方。我们不能依靠运气，所以要根据指针的运动模式，决定自己的飞行。

现在你已经回到了 255°，减速到 75 节，放出一级襟翼（10 度），并下降到 2800 英尺并保持（不要低于此高度）。通过检查 FOOTO 确认我们进场的位置。现在抬头挺胸迎接落地。

减速到 75 节；放出一档襟翼；下降到 2800

10.5.9 截获下滑道

当我们向跑道飞去的时候，不要忘记看一眼水平的指针，也就是下滑道指针。当我们截获了航向道之后，下滑道应该比我们高，因为我们实际是低于下滑道的。当我们在 2800 英尺改平的时候，下滑道开始“向下”运动。当指针完全水平，表示我们在下滑道上了[1]。很快我们就截获了下滑道，我们会经过外指点标。很多事情同时出现，都可以帮你确认自己的位置:

1. 你会听到了一系列连续的嘀嘀声 。
2. COMM1 上边的标有“O”的蓝色灯会亮起。
3. ADF 指针会偏向一侧[2]。

[1]还有可能我们进入了错误的下滑道，而 FlightGear 也实现了这个功能。因此需要我们确认进入了正确的下滑道，这就是为什么程序转弯要进入正确的位置，正确的高度，这样才可以截获正确的下滑道

[2]这是因为 REIGA NDB 就在外指点标上（航图上 LOM 标记），而飞机飞过 NDB 上空时，ADF 指针会指向左侧或右侧。——译者注

截获下滑道；穿过外指；放出第二级襟翼

截获下滑道以后，我们要开始下降。如何才是最好的下降率？这取决于你的地速。在我们的例子里，我们空速 75 节（因为几乎没风，可以认为空速就等于地速），也就是大约按每分钟 400 英尺速率下降。如果开着自动驾驶，就简单了。只需要调到 -400 就行（不过记得收油门以维持 75 节的速度，否则你会因为速度太快而撞到跑道上，同时准备随时高于或低于下滑道时转入手动）。

如果没开自动驾驶，也很简单。只需要收油。收多少？在我们这个例子里，我们的飞机大约减到 1700 RPM。注意，这取决于很多因素，飞机、升降舵、风、重量……因此你必须注意下滑道移动时做出调整。与航向道指针一样，*不要追着指针飞*。观察它如何运动，进而决定你的调整。

因为已经在最后进近阶段，你也许会想放出第二级襟翼。这会影响到你的调整片，同时还需要对发动机动力做出修正。

10.5.10 几乎要落地

激动人心的程序转弯之后，我们也许飞过外指标之后的很长一段航程。也许除了盯着这些指针以外没什么可以做的时请了。事实上，你还需要注意一下之前很少关注的仪表。看看空速，我们是不是太快或者太慢，以致快要失速。发动机转速是否合适？如果手动驾驶，你还需要同时扫描高度和方位陀螺仪。毕竟这是模拟器，我们不需要关注润滑油压力和发动机温度，不过也许你会看一眼，就当作一种习惯。同时我希望你已经将油气混合比推到富油（难道你在巡航时没有适当贫油吗?）。如果你希望还可以将襟翼全部放出，这样可以让你在跑道上更快停下。

10.5.11 一些说明

实际上我让你做了很多额外的工作。我们已经很好的利用了自动驾驶仪，而它还可以做更多的事情。你也许注意到自动驾驶仪上还有很多按钮我没有介绍——NAV、APR 和 REV。使用这些按钮，自动驾驶仪可以：

NAV： 切入一个 VOR 径向线

APR： 直接做一个 ILS 进近，同时追踪航向道和下滑道。

REV： 在程序转弯之前截获 ILS （比如*反向*截获航向道）。

实际上，你所做的很多工作都可以用自动驾驶仪来替代。起飞以后，我们就可以用 NAV 模式直接切入 SJC 的 009 径向线一路飞到 SUNOL；在 SUNOL 你也许会利用 REV 模式走 I-LVK 的“反向进近”航道；使用 HDG 模式完成程序转弯；最后，使用 APR 模式追踪航向道和下滑道。

我之所以没有把这些信息告诉你基于两点原因：首先，手动驾驶可以让你了解发生了什么。其二，很多官方手册里写出来的功能并没有实现，最好依赖于我们熟悉的功能。

10.5.12 不落地

虽然 ILS 进近可以引导我们比 VFR、NDB 或 VOR 进近更靠近跑道，我们还是需要一些能见度来降落[1]，因此我们需要一种方法决定是否降落。进近航图上写有着陆最低标准（见图 10.10 中绿色的部分）。“S-ILS 25R”（就是我们正在用的）一项下面可以看到“597-1/2 200(200-1/2)”，意思是我们可以跟着下滑道一直下降到高度 597 英尺，也就是距跑道高 200 英尺。在 597 英尺我们需要做决定，如果看不到跑道，我们需要执行复飞。597 英尺就是我们的**决断高**（*Decision Height*，*DH*）[2]。

关于高度表，再做一些补充。就这次进近而言，还有一个能告诉我们已经接近跑道——中指点标（MM）。此指点标会出发声音，也就是一连串嘀搭声，同时 COMM1 上方标有“M”的黄灯亮起闪烁。通过中指点标，也意味着快要到达决断高了[3]。

那么如果你在决断高（DH），看不到跑道怎么办？也许你会想，不管三七二十一直接复飞，而你不能如此莽撞。这里有一个程序，也就是复飞程序（Missed Approach Procedure）。可以在进近航图的很多地方看到此程序（见图 10.17）：顶部可以看到写有“MISSED APPROACH”的说明文字；在平面图里，也可以看到跑道尽头的虚线箭头，以及右侧的椭圆航径；剖面图里同样也能看到一些图标告诉你如何做。在我们的例子里，这些都告诉我们：

1. 直飞爬升到 1100 英尺
2. 右转并爬升到 3000 英尺
3. 飞向 REIGA
4. 沿 REIGA 062° 离场
5. 在 TRACY 交汇点飞一个等待航线

等待航线，也许你已经猜到了，就是将飞机“停泊”在这里等准备妥当，再行进近。等待航线也有其一系列的程序，我们这里不会讲到，因为……

10.5.13 落地

在我们理想化的模拟世界里，你也许不会执行复飞。假设你依旧还在下

目视跑道；断开自动驾驶；穿越中指点标

[1]除非使用三类 C（CAT IIIC）盲降。

[2]此处应为“决断高度”（Decision Altitude，DA），因为 597 英尺是相对平均海平面的，所以是“高度”（Altitude）。而 200 英尺是相对地面的，所以称为“高”（Height），因此 200 英尺才是“决断高”（Decision Height，DH）。为与原文保持一致，翻译时并没有修正。——译者注

[3]你也许会好奇，剩下的那个标有“A”的白灯是什么意思，这表示内指点标（IM）。我们的进近并没有这个设备，不过旧金山机场的 28R 跑道有。当飞过 IM 表示离跑道更近，一连串高音嗒声也会响起。为什么标字母“A”而不是“I”呢？因为在早期，此指点标还用来表示飞过“航线”（Airway）所在的无线电范围。

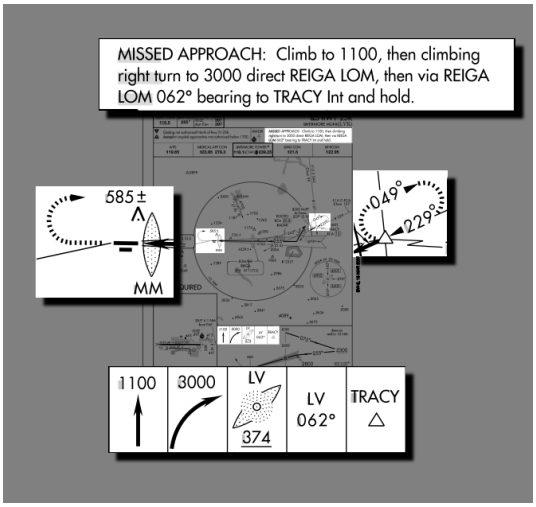


图 10.17: 复飞程序

落地；吃汉堡

滑道里，在 800 米能见度下，决断高附近跑道应该会突然出现。可以目视跑道以后，你需要疯狂的对准跑道[1]（因为很难完美的对准跑道）并“正常的”降落（而我在跑道上蹦跳滑行了好久）。停泊好飞机，然后爬出驾驶舱，去享用一顿汉堡吧！

10.6 结语

短时间给出了太多信息，特别是有关 ILS 飞行的介绍太简略。希望，这些信息没有让你离开，而是激发了你的求知欲，因为确实还有大量的知识要学。很多主要的都被我们忽略了：

风 这是最大的一环，在侧风中飞 IFR，会对你的所有操作产生影响，你需要时刻留意导航，否则就会迷航。

没有自动驾驶仪的飞行 自动驾驶仪已经很好了，但并不那么值得信任。你需要随时准备接手飞行。

方位陀螺仪的进动 C172P 的方位陀螺仪并不完美。随着时间推移，其值越来越不可靠——也就是进动了。它需要随时与磁罗盘校准（使用 DG 旁边的 OBS 旋钮）。

IFR 航图 我们已经用过区域航图了，对 VFR 非常有用。而还有一整套有关 IFR 的航图没有介绍。

[1]记住，一定要断开自动驾驶

图 10.18: 对准跑道，目视跑道。我们又活了！

ATC 空中管制人员需要知道你在做什么，同样的他们也会指引你要做什么，包括忽略你如此研究进近航图里的一些内容[1]。

SID/DP、航路以及 STAR 此教程介绍了 IAP（仪表进近程序），也就是进近的一种标准程序。在 IFR 飞行中，有许多标准方法离开机场（也就是“标准仪表离场程序”Standard Instrument Departures，*SID*。或者“离场程序”Departure Procedures，*DP*）；在机场与机场之间飞行需要穿越很多航路（Airway）；从航路进入仪表进近程序的“标准终端进场航路”（Standard Terminal Arrival Routes，*STAR*）。

等待航线 大多数复飞程序都会有等待航线，你最好学习一下如何飞行等待航线。

GPS 我们的塞斯纳飞机没有配备 GPS，但现在很多小飞机都有了，并且 GPS 正在快速取代无线电导航设备。

如果你还想学习更多，尝试下面这些资源：

- *Flight Simulator Navigation* Charles Wood 写的，里面讲到了从基本导航到 ILS 进近的方方面面，还有大量的实例来帮助你提高技能。所有实例都以一个包机服务飞行员的故事作为主线。

  两个警告，其一，这是面向微软模拟飞行，因此你需要自己适当调整成 FlightGear 可用。其二，教程有一些过时，这么多年很多信息都已经

[1]原文这里的意思是，ATC 可以协助进行雷达引导进近。——译者注

变化了。有些 NDB 信标台已经退役，新的方法取而代之。甚至有些机场也已经消失。不过作为一个学习机会，你也许会找到更多最新的信息，并学习到不盲目相信你的航图，正如你已经学习到不盲目相信仪表一样。

- 如果你真的想要获得更加官方的文件，可以访问 *FAA Instrument Flying Handbook*。里面讲的非常具体，也没有什么故事性。更多相关文档可以从 *Aviation Handbooks & Manuals* 里找到。

- 如果你只想学习仪表相关内容，而不想通过飞行实践（甚至虚拟飞行），可以尝试 *luizmonteiro.com* 。里面使用 Flash 演示了各种仪表，当然也包括 VOR 和 ADF。

- 另一个虚拟的仪表网站是 *Tim's Air Navigation Simulator*。通过 Java Applet 模拟飞机在两个导航台之间的轨迹。模拟器允许你使用不同的仪表和导航设备，因此你可以看到他们的行为差别，比较优缺点[1]。

- 如果你想获取相关导航信息，可以通过 *AirNav.Com* 也就是此手册里已经用过的。里面有机场、导航台和定位点的详细信息，并且还有相关 IAP 的链接。很不幸这个网站只覆盖美国境内。

- 另一个网站是 *World Aero Data* 虽然不如 AirNav 那么细节，但至少是覆盖全球的。

- *FlightSim.Com* 网站也有一系列文章 "How To . . . Use Approach Plates"。里面讲解了如何阅读进近航图，并以阿拉斯加科迪亚克机场的进近为例。这可以算作是 Charles Wood 的 *Flight Simulator Navigation*（见上文）的补充材料。

  最有意思的是里面还介绍了 "危险进近"，也就是降落在世界上最危险的六个机场。从彭蒂克顿机场到尼泊尔的加得满都机场，都有介绍。你可以挑战自己的极限！

  警告：这些都是以微软模拟飞行为中心介绍的，很多也已经过时（外链已经损坏，或很多进近方法也已经修改）。

- 还是从 *FlightSim.Com* 的文章 "Golden Argosy"。美国航空 767 的机长 Tony Vallillo 写的，详细讲述了一次从纽约到罗马的飞行。里面讲到了很多其他网站没有提到的导航信息，比如北大西洋航线（North Atlantic Tracks）。然而这篇文章很好的回答了 "真实飞行员是怎么飞行的?" 而且字里行间也透露出作者对飞行的热爱。

[1] 此网站已经失效。——译者注

- 如果对 ATC 有兴趣，希望获取相关信息的，可以看 Michael Oxner 的文章 “Aviation Topic of the Week”[1]。一系列文章讲到了在各种空域和各种情况下的飞行。Michael Oxner 是一名专业的管制员和私人飞行员。闲暇时光，他也是一位 VATSIM 的在线虚拟管制员，却没有足够多的飞机。他特别对一系列讲述完整 IFR 飞行和 VFR 飞行的文章感兴趣。

[1]此网址已经失效。——译者注

第十一章
直升机教程

11.1 前言

首先：任何适用于真实直升机的原理都适用于 *FlightGear*。相关的基本操作也可以在这里找到：
http://www.cybercom.net/~copters/pilot/maneuvers.html 很多细节在 *FlightGear* 里面都有所简化。特别是处理发动机相关还有超过应力的部分并没有模拟出来，或者不会产生什么后果。在 *FlightGear* 里，（至少目前）很难损坏一架直升机。

FlightGear 里的直升机飞行模型是非常真实的。唯一的例外是“涡环状态”。这会在下降太快太垂直（没有向前的速度）时发生。直升机升力会大幅减少，而进入自身旋翼产生的下洗气流中。要避免这种情况除非是比较高的高度。你也许可以在互联网上找到一个视频“Seaking helicopter”，一次飞行表演中直升机艰难的落地却进入这个状态，最后飞机完全损坏。

FlightGear 里的所有直升机参数全都没有优化，因此模型和原始之间会有轻微差别。硬件上，我建议适用一款“好”的游戏杆。没有弹簧的游戏杆最好，因为这样它不会靠自己的力量回中。你既可以自己将弹簧移除，也可以使用有力回馈的游戏杆，配有额外的电力供应。另外，游戏杆要有“油门

控制器”（油门）。为了控制尾桨，你需要有脚踏板或至少一个可扭转的游戏杆，因为使用键盘非常难控制。*FlightGear* 支持同时连接多个游戏杆。

11.2 开始飞行

FlightGear 里面的直升机数量很有限。我认为 Bo105 是最容易飞行的，因为它的反应比其他直升机更直接。从飞行行为上说，也推荐 S76C。S76C 的反应比 Bo105 要迟缓。

当你载入 *FlightGear* 以后，花一些时间将各控制面移动移动并回中。特别的，在启动时很多都是在最大值。

直升机由四个操纵面来控制。游戏杆控制其中两个，旋翼的倾角（同时也是直升机的倾角）前后和左右。而这些称为“旋翼锥角控制”[1]。还有一个是“总桨距控制”，由油门杆来控制。可以改变旋翼产生的拉力。因为主旋翼的动力，会对机体产生反向力矩，因此需要使用尾桨来修正。因为力矩与飞机总桨距相关，同时也与飞行时的风相关，因此飞行员可以用脚踏板来控制尾桨。如果你向右蹬脚踏板，直升机会向右转。脚踏板不是方向盘。可以用脚踏板让飞机绕垂直轴线旋转，而旋翼的公转速度保持恒定（如果可能）。

[1]旋翼锥体，表示旋翼旋转时的桨叶与桨毂运转平面的角度。——译者注

11.3 起飞

首先将总桨距设置到最小。为了增加总桨距你需要向上“拉”总桨距杆，为了让总桨距最小你需要向下推（而这却是油门控制最大位置）。同样的，“最大动力”也就是油门杆慢车的位置。使用 } 起动发动机。几秒之后，旋翼开始旋转并慢慢加速。保持操纵杆和脚踏板中立。等待旋翼加速完成。对 Bo105 而言左上方有一个发动机转速仪表。

当旋翼加速完成以后，慢慢拉起总桨距杆。眼睛盯住水平线。如果直升机轻微倾斜或偏转，不要继续拉总桨距杆，先用操纵杆和脚踏板修正位置和移动。如果成功，就继续拉总桨距杆（一定要慢！）。

11.4 在空中

为了防止你的失望情绪，先不学平飞，你也许会尝试向前飞。起飞以后，继续拉总桨距杆一段时间，然后用操纵杆降低机头。直升机会向前加速。因为向前飞行，所以尾桨不需要怎么控制，因为相对风是从前吹来。直升机的向前飞行就像一个没调整好的普通飞机。所谓“中立”位取决于空速和总桨距。

从前飞转到悬停是很容易的，只要你慢慢降低速度并升起机头。同时，降低总桨距防止直升机爬升。当直升机慢下来以后，“升力的分量”降低，所以还需要增加总桨距来修正。速度接近 0 时，稍稍降低机头维持悬停，否则直升机会向后倒退。

11.5 回到地面 I

直升机的降落，首先要如上文所述要转成悬停，通过降低总桨距来降低高度。简而言之，在碰到地面之前慢慢降低下降率。一个完美的落地是，高度为 0 时，速度和下降率也变为 0（温柔的操作）。然而降落是非常困难的，大多数飞行员可以在地面附近进入悬停状态并慢慢下降。落地时带着向前的速度会比较容易，但要确保不要落到任何物体的边缘（侧面），防止翻覆。

11.6 回到地面 II

还是值的讲一讲自转的。这是一个无动力飞行状态，也就是流过旋翼的空气会将旋翼转动。在适当的高度选择一个降落点（刚开始最好是一块较大面积的空域）然后按 **{** 键将发动机关闭。将总桨距降到最小，把尾桨的桨叶角设置在 0°（Bo105 是将脚踏板推到右侧一半的位置，而 AS350 是左侧）。以 80 节速度进近。不要让旋翼的速度超过 100% 太多，否则旋翼就会损坏（虽然并没有模拟出来）。到达地面附近，通过抬升机头降低速度。同时下降率也会降低，因此你不需要拉总桨距杆。这样会导致旋翼转速过快超过可允许的范围。可通过提升总桨距来抵消这个效应。快到地面时，提升总桨距来减少下降率。目的是以较低的下降率接地并且没有向前的速度分量。如果有向前的速度会更简单，但若起落架与飞行方向不平行的话会有翻覆的危险。在进近的过程中不需要调整尾桨，因为没有动力也就几乎没有力矩。如果你感觉（经过一些练习）自转太简单，可以增加一些真实的载重，可以通过 `payload` 菜单实现。

第四部分

附录

附录 A
复飞：如果有什么拒绝工作

下面的章节，我们尝试按操作系统列出一些问题，若你碰到一个问题，也许最好先看看“你的”操作系统。万一，如果你正在遇到问题，我们强烈建议你首先看看由 Cameron Moore 维护的一份 FAQ

http://wiki.flightgear.org/Frequently_asked_questions.

另外，源码目录里的 `docs-mini` 目录也包含大量特定问题的解决方案。这里也可以找到更多参考。

A.1 FlightGear 问题报告

寻找帮助最好的地方是邮件列表，特别是 **[Flightgear-User]** 邮件列表。如果你恰好正在运行 Git 版本的 *FlightGear*，你可以将问题报告给 **[Flightgear-Devel]** 相应的介绍可以看

http://www.flightgear.org/mail.html.

很多时候，也许别人已经解决了你遇到的问题，所以可以先在邮件列表存档里搜索一下

http://sourceforge.net/mailarchive/forum.php?forum_name=flightgear-users
http://sourceforge.net/mailarchive/forum.php?forum_name=flightgear-devel.

你也可以去 FlightGear 论坛搜索获得一些信息，有关论坛的介绍和存档：

http://www.flightgear.org/forums.

大量的开发者和用户会阅读这些邮件列表和论坛，因此泛泛而谈的问题，比如类似 *FlightGear 在我的系统上不能编译，请问怎么做？* 这种没有提供更多信息的问题，很难获得回答。那么该如何做呢？当你报告问题的时候，可以加上这样一些信息：

- **操作系统：** （Linux Fedora 17. . . /Windows 7 64 位. . . ）

- **计算机：**（奔腾酷睿双核 2.3GHz...）
- **显卡/芯片：**（ATI Radeon HD 770 XT/NVidia GeForce GTX 590...）
- **编译器/版本：**（GCC 版本 4.6.3...）
- **相关库的版本：**（PLIB 1.8.5, OpenSceneGraph 3.0.1...）
- **问题类型：**（链接器输出这些错误信息...）
- **复现问题的步骤：**（启动到 KSFO，关闭刹车...）

要想获得 *FlightGear* 的追踪信息，以下命令也许会帮到你（可能需要针对操作系统做一些修改，也许其他系统不能工作）：

```
%FG_ROOT/BIN/fgfs >log.txt 2>&1
```

最后要说：请不要向邮件列表发送二进制文件！邮件列表订阅者都是分散的，有些用户的网络连接可能很慢。过大的信件可能会被邮件列表管理员退掉。谢谢合作。

A.2 通用问题

- *FlightGear* 运行太太太慢了。
  如果 *FlightGear* 运行时的帧速率只有 1 fps 或者低于普遍的情况，像是硬件不支持 OpenGL。也许有很多种原因。首先，也许老旧显卡不支持 OpenGL。如果是这样，建议换一块新显卡。

  第二，检查驱动是不是安装正确。很多显卡除了 Windows “自带”的驱动还需要额外的 OpenGL 驱动支持。

- `configure` 或 `make` 命令卡在了找不到 *PLIB* 头文件或库文件
  确保你已经编译并安装了最新版的 *PLIB*（> 1.8.4 版）。其头文件 `pu.h` 必须放在 `/usr/include/plib` 目录及其库里，比如 `libplibpu.a` 需要在 `/lib` 目录下。反复检查 *PLIB* 的头文件和库文件是不是放在别的地方。

  仔细检查 `configure` 命令输出的错误信息，很多情况下它会告诉你缺少了什么。

A.3 Linux 的潜在问题

因为我们不能在所有 Linux 发行版下面测试问题，下面这些是可能出现的问题（此节感谢 Kai Troester 的贡献）。

- 错误的库版本
  这是手动安装 *FlightGear* 时比较常碰上的问题。确认 Mesa 库已经包含了对 3DFX 的支持，同时 GLIDE 库已经安装并可以找到。如果 `ldd` `which fgfs` 命令告诉你有缺失的库，那就是遇到麻烦了。

  你需要一直确保 *PLIB* 已经安装了最新版。很多人编译 *FlightGear* 失败大多是因为使用了过时的 plib。

- 缺少权限
  如果你使用 4.0 发布版之前的 XFree86，*FlightGear* 的二进制文件可能需要 setuid 到 root 以获得访问某些基于 3DFX 图形加速卡的权限（或者如前文所述的特定内核模块）。你可以用以下两种方法之一：

  ```
  chown root.root /usr/local/bin/fgfs ;
  chmod 4755 /usr/local/bin/fgfs
  ```

  来给 *FlightGear* 二进制文件赋予合适的权限或安装 3DFX 模块。后一种方法比较“干净”且强烈推荐！

- 非默认安装选项
  FlightGear 启动时会做大量的诊断。如果出现难看的界面或缺少文件，请检查是否按照预定的方法安装（最新的版本和适当的位置）。*FlightGear* 需要其数据文件都在 `/usr/local/lib` 目录下。请确保已经获取所有最新版的组件！

- 编译错误
  确保你安装了最新（官方）版的 gcc。旧版 gcc 会有大量的问题！另一方面，某些版本的 RedHat 7.0 编译 *FlightGear* 时出现问题，因为使用了原始版本的 gcc。

A.4 Windows 的潜在问题

- 可执行文件拒绝运行
  也许你需要直接在 Windows 资源管理器里双击 `fgfs.exe` 文件，或者通过 MS-DOS 命令行运行。如果在资源管理器里双击也不管用（除非你在 `autoexec.bat` 或其他地方修改了 `FG_ROOT` 环境变量），可以双击 `fgrun`。更多相关信息可以参考第 四 章。

  另一种导致这种问题的可能性是你没有下载新版的 *FlightGear* 基础包，或者你根本没有下载。请仔细留意，因为地景/贴图格式还在开发中会经常变化。相关细节请参考第 三 章。

  下一步，如果在运行时出现问题，不要使用 Windows 实用工具解压 `.tar.gz` 文件。如果需要解压，可以在 Cygnus shell 里执行 `tar xvfz` 命令。

- *FlightGear* 忽略命令行参数
  可能是因为传递参数时包含了“=”字符，这样会创建批处理进程而不是响应命令行参数。

- 我不能在 MSVC/MS DevStudio 下构建 *FlightGear*
  默认情况下，*FlightGear* 使用 GNU GCC 来构建。在 Win32 下移植的 GNU GCC 是 Cygwin。有关 *Makefiles* 对 MSVC 或 MSC DevStudio 的需求可以看：

  https://www.gitorious.org/fg/flightgear/trees/next.

  原则上，可以使用源代码里的项目文件编译 *FlightGear*。

- 编译 *FlightGear* 失败
  有很多原因导致，可能是 bug。然而在报告问题之前，确保已经使用了最新版的 *Cygwin* 编译器。确保无虞，最好运行 `setup.exe` 并下载最新版的文件，因为这些文件可能经常变动。

附录 B
落地：离开飞机之前的一些想法

B.1 *FlightGear* 的简略历史

讲述历史往往很无聊。然而，一直以来有很多人询问 *FlightGear* 的历史。那么我们不妨就来讲述一下吧。

FlightGear 项目肇始于 1996 年的一次网民讨论，最终 David Murr 根据这次讨论写了提案。很不幸，他后来放弃了此项目（也包括网络）。原始的提案依然可以在 *FlightGear* 网站找到：

http://www.flightgear.org/proposal-3.0.1.

虽然开发者和各种细节随着时间早已地覆翻天，然而这份提案的精神却保留至今。

实际上在 1996 年夏天开发就已经开始了，到年底时图形程序也已完成。那时，主要编码工作由伯克利大学的 Eric Korpela 负责。早期代码可以在 Linux、DOS、OS/2、Windows 95/NT和 Sun-OS 上运行。所以这是一个雄心勃勃的项目，而且编写系统独立的图形界面完全是白手起家。

开发工作很缓慢并最终在 1997 年初停滞，因为 Eric 完成了他的论文。就在此时，项目似乎要死掉，邮件列表也没有什么讨论。

1997 年年中的时候，在明尼苏达州大学就读的 Curt Olson 为这个项目带来了希望。他的理念简单却强大：为什么要重新发明轮子呢？在 UNIX 的各种工作站上有很多自由的飞行模拟器可用。其中之一就是 LaRCsim（由 NASA 工程师 Bruce Jackson 开发），看起来非常适合。Curt 重拾这个项目，并重写了很多界面代码以便可以运行在不同的平台。这么做的关键就是为了使用系统平台无关的图形平台——OpenGL。

另一项明智的决定是在第一版就选择基础地景数据。*FlightGear* 的地景是基于美国地质勘探局发布的卫星数据。这些地形数据可以从这里下载：

http://edc.usgs.gov/geodata/

美国和

http://edcdaac.usgs.gov/gtopo30/gtopo30.html,

和其他国家。这些可访问的地景数据，连同地景构建工具都包含在了 *FlightGear* 里面，这项重要的特性可以让任何人都创建自己的地景。

新的 *FlightGear* 代码（依旧大量基于原始 LaRCsim 项目）首先在 1997 年 7 月发布。从这时起，这个项目重获新生。下面介绍最近开发历史中的一些重要里程碑。

B.1.1 地景

- Curt Olson 在 1998 年春天加入了纹理贴图支持。这对现实性有非常大的提高。很多高质量的纹理贴图由 Eric Mitchell 提交到 *FlightGear* 项目。Hofman 则提交了另外一系列高质量贴图。

- 1998 年春天，在提升了地景和纹理贴图支持之后，*FlightGear* 的帧速率掉到了几乎无法飞行。这个问题因为 OpenGL 对硬件支持而解决，同时 Curt Olson 实现了视域剔除(一种渲染技术，可以将视野里不可见的地景剃除)。

  关于帧速率，有些人始终没有放下，是时，没有什么方法去优化，一直留待某人去完成。

- 1998 年 9 月，Curt Olson 成功完成了美国的地形模型。现在地景已经覆盖全球，可以使用这个可点击的地图:

  http://www.flightgear.org/download/scenery/.

- 后来地景更加入了地理特征，包括了湖泊、河流和海岸线。Dave Cornish在 2001 年春为跑道添加了纹理贴图。光线贴图也为晚间增加了视觉色彩。为了应对不断增加的地景数据，2001 年春一种二进制地景格式引入到项目中。同时 Curt Olson 也引入了跑道光线。最终到 2002 年夏天，William Riley 基于 David Megginson 的筹备文档，创建了一套覆盖全球的完整地景文件。基于的数据称为 VMap0 而现在使用的是 GSHHS 数据。这套覆盖全球的地景包括主要的街道、河流等等，然而海岸线还不够精确。*FlightGear* 的基础地景从 2002 年夏开始就基于这套地景文件。

- 2001 年还增加了对静态物体的支持，这样就可以在地景上放置建筑物、静止的飞机、树木等等。

- 2002 年夏覆盖全球的随机地面物体也引入了，以合适的类型和精细度覆盖本地地面。这项现实性增强要感谢 D. Megginson 的贡献。

- 今天，我们始终在努力，随着 Terrasyc 的引入，可以让用户一边飞行一边下载地景，强大的 mapserver 和地景建模基础架构作后盾，网站也让我们可以快速增加和更新地景。地景生成工具也大幅更新，以便可以将精细度提升到 8.50 apt.dat 格式。同时，当协议适当的时候，也会引入外部数据源比如 OpenStreetMap。

B.1.2 飞行器

- 1997 年秋，基于 Michele America 和 Charlie Hotchkiss 贡献的代码，HUD（Head Up Display，平视显示器）引入到了项目中，随后 Norman Vine 又做了大幅提高。虽然对默认的塞斯纳 172 并没有什么帮助，不过因为可以显示飞行姿态，对后面引入军用喷气机有很大帮助。

- 1998 年 4 月，低级的自动驾驶仪航向保持模式由 Jeff Goeke-Smith 贡献到项目中。随后当年 10 月，Norman Vine 又增加了高度保持和地形追踪开关。

- 1998 年 6 月，Friedemann Reinhard 开发了早期的仪表板代码。不幸的是，后来开发慢了下来。最终 David Megginson 在 2000 年 1 月决定开始重构仪表板代码。这大幅增加了仪表和特性，这样到 2001 年春，几乎所有主要仪表都已经包含进去了。一个小型的迷你仪表板在 2001 年夏天开发出来。

- 最终，LaRCsim 里的 Navion 飞机被现在的默认塞斯纳 172所替代，塞斯纳 172 在 2000 年 2 月变得足够稳定，这招致大多数用户的欢迎。这样在运行时就可以选择多种飞行模型和飞机选项了。Jon Berndt 投入大量时间实现了一个飞行器配置方式，提高了真实感和灵活性。*JSBSim*，后来替代了 LaRCsim 成为默认的飞行动态模型（FDM），它也打算包含一些特性比如燃油搅拌特效、乱流、完整的飞行控制系统和其他飞行模拟器里不太常见的特性。作为替代，Andy Ross 于 2001 年底开发了另一个飞行动态模型 *YASim*（Yet Another Flight Dynamics Simulator），使用简单特性并基于流体力学原理。此飞行动态模型带来了 747、A4 和 DC-3。另外，Michael Selig 领导的团队在 2000 年开发了 UIUC 飞行动态模型和一系列相应的飞行器。

- 一套完全可用的 无线电栈 和无线电系统由 Curt Olson 于 2000 年春开发完成。大量导航台数据库，这样就可以使用 ILS 也由 Robin Peel 贡献进来。基本的 ATC支持也在 2001 年底由 David Luff 开发进来。这还没有完全实现，但显示 ATIS 消息成为可能。磁电机开关和相应的程序在 2001 年底由 John Check 和 David Megginson 开发完成。同时在 2001 年到 2002 年间，大量仪表也由 John 和其他人持续完善。*FlightGear* 现在就可以允许飞 ILS 进近并可以使用 Bendix 应答机了。

- 2002 年多发动机支持引入到了 *FlightGear*。*JSBSim* 成为 *FlightGear* 默认的 FDM。
- 2002 春，John Check 和其他人的努力使 “真实的” 3D 仪表板变得足够稳定。同时可移动的操作面 比如螺旋桨等等，由 David Megginson. 开发完成。

B.1.3 环境

- 长期以来，太阳、月亮和星星的显示一直是 PC 飞行模拟器的弱点。*FlightGear* 的一大成就是早期就包含了精确的太阳、月亮和星星的建模。相应的天文学代码在 1997 年秋由 Durk Talsma 实现。
- Christian Mayer 与 Durk Talsma, 一起于 1999 年冬贡献了天气相关代码。包括了云、风，甚至还有暴风雨。

B.1.4 用户界面

- 1998 年 6 月菜单系统基于另一个库，可移动库 *PLIB* 开发而成。经历一段时间停滞以后，1999 年春，第一版可工作的菜单项问世。

  后来 *PLIB* 经过快速开发。Steve Baker 在 1999 年春将其分离单独打包，脑中也有了更广泛的应用前景。从 1999 年秋开始提供给 *FlightGear* 一个基本的图形渲染引擎。
- 1998 年有了基本的音频支持，一个音频库和一些基本的发动机背景声音。后来这些被集合到了上文提到的 *PLIB* 中。这个库在 1999 年 10 月扩展了对游戏杆、驾驶盘和脚踏板的支持，为提升真实感迈出了巨大的一步。为了适应不同的游戏杆，2000 年秋引入了游戏杆配置选项。2002 年夏天 David Megginson 基于 xml 游戏杆文件开发了游戏杆自动检测特性。
- 1999 年秋 Oliver Delise 和 Curt Olson 投入时间开始开发网络/多人游戏。目标是让 *FlightGear* 可以通过网络在多台机器上同时运行，无论是内部网还是互联网，可以和运行在另一个机器上的飞行计划程序媾和在一起。2001 年出现了多种通过网络远程控制 *FlightGear* 的方法。值得注意的是还有对 “Atlas” 的支持，也就是一个可移动的地图。另外，一个嵌入式的 HTTP 服务器由 Curt Olson 于 2001 年晚些时候开发出来，还能为外部程序提供属性管理器。
- 飞行模拟器里手动改变视角在某种意义上说，总是 “不真实” 的，尽管如此当前状况也需要它。一个可能的解决方案在 1999 年冬天由 Norman Vine 实现用鼠标改变视角。另外，现在你也可以用苦力帽达成这个目的。

- 一个属性管理器由 David Megginson 在 2000 年秋天开发出来。它可以使用一个配置文件，比如 UNIX/Linux 系统下的 `.fgfsrc` 文件，Windows 系统下的 `system.fgfsrc` 文件。这个纯 ASCII 文件可以避免提供大量的输入选项，还有游戏杆配置。2001 年春天时，游戏杆、键盘和仪表板的选项不再是硬编码，而是使用 *.xml 文件，这要感谢 David Megginson 和 John Check 的贡献。

在开发过程中，大量代码被重新组织和整理。很多代码子系统分立打包，结果代码组织就是现在的样子：

主图形引擎是 **OpenGL**，一个系统平台无关的库。基于 OpenGL 的是可移动库 *PLIB* 提供基本的渲染、音频、游戏杆等等程序。基于 *PLIB* 的是 *SimGear* 包括了 *FlightGear* 和构建地景所需的所有基本程序。在 *SimGear* 之上的是 *FlightGear*（模拟器本身），以及 *TerraGear* 包括了地景构建工具。

这不是一份详尽的历史，极有可能某些人做出重大贡献，而没有列在其中。除了上面所列的贡献者还有大量工作涉及内部结构：Jon S. Berndt、Oliver Delise、Christian Mayer、Curt Olson、Tony Peden、Gary R. Van Sickle、Norman Vine 等人。一份完整的贡献者列表可以在附录 B 里面以及源码目录的 `Thanks` 文件里找到。同样 *FlightGear* 网站也包含了一份详细的历史值的一读，包括了所有值的注意的开发里程碑：

http://www.flightgear.org/version.html

B.2 那些贡献其中的人

你是否享受飞行的乐趣呢？如果是这样的话，不要忘了那些人在这上面付出了数百小时。所有这些工作都是在业余时间志愿完成，如果不符合你的期待，不要责怪这些程序员。请坐下来并给他们写一封邮件，讲述你的想法和改善的希望。或者，你可以订阅 *FlightGear* 的邮件列表并向里面贡献你的想法。相关介绍可以在这里找到：

http://www.flightgear.org/mail.html.

注意有两个列表，一个是由开发者维护另一个是由用户维护。其他一些主要用来发通知。

以下这些是贡献者的名字及其贡献（信息从源码目录的 `Thanks` 文件提取出来的）

A1 Free Sounds
授权 *FlightGear* 项目使用其网站上的声音效果。主页在此：

http://www.a1freesoundeffects.com/

Syd Adams
增加 2D 仪表，ATC 音量控制并创建了大量飞行器。

Raul Alonzo
Alonzo 先生是 Ssystem 的作者并授权使用月亮贴图。他的一些成为添加贴图的模板。Ssystem 的主页：

http://www1.las.es/~amil/ssystem/.

Michele America
贡献 HUD 相关代码。

Michael Basler
安装和使用入门的作者，飞行模拟主页：

http://www.geocities.com/pmb.geo/flusi.htm

Jon S. Berndt
用 C++ 重写/重构核心 FDM。最开始他用 X15 来测试代码，开发完成后用其他飞机也可以。Jon 维护了一个网页处理飞行动态模型：

http://jsbsim.sourceforge.net/

特别留意此网站的很多页面 X15 是付费的。另外，Jon 对此指南贡献了很多建议和修正。

Paul Bleisch
重构了调试系统以便更加灵活,这样在产品化的系统中可以很容易的禁用,也可以选择性的启用子系统的调试信息。同时他也贡献了第一个配置文件/命令行参数系统。

Jim Brennan
为 *FlightGear* ! 的美国地景提供了大量的在线存储空间。

Bernie Bright
很多 C++ 风格、使用和增强实现，STL 等很多很多方面。增加了多线程支持和线程平铺分页器。

Stuart Buchanan
更新此指南的多个部分，写了教程子系统的第一版，开发了随机植物和建筑物。

Bernhard H. Buckel
贡献了 README.Linux。贡献了早期版本的安装和入门指南的多个部分。

Gene Buckle
大量工作让 *FlightGear* 可以使用 MSVC++ 编译器。也为细节提升做了很多提示。

Ralph Carmichael
支持此项目。公共领域航空软件网页：

http://www.pdas.com/

PDAS 销售的 CD-ROM 包含了很多航空工程师所需的程序。

Didier Chauveau
提供了解析 30 arcsec DEM 文件的基础代码

http://edcwww.cr.usgs.gov/landdaac/gtopo30/gtopo30.html.

John Check
John 维护了基础包的 CVS 仓库。他贡献了云贴图，写了非常优秀的游戏杆文档和仪表板文档。同时，她也贡献了新仪表配置文件。他的 *FlightGear* 主页：

http://www.rockfish.net/fg/.

Dave Cornish
Dave 创建了很酷的跑道贴图和云贴图。

Oliver Delise
开启了 FAQ、文档和公共关系。增加了一些网络/多人游戏代码。是 FlightGear MuliPilot 的创始人。

Jean-Francois Doue
矢量 2D，3D，4D 以及矩阵 3D 和 4D 的内联 C ++ 类。（基于 Graphics Gems IV, Ed. Paul S. Heckbert）

http://www.animats.com/simpleppp/ftp/public_html/topics/developers.html.

Dave Eberly
为 Christian Mayer 的天气数据库系统贡献了一些球面插值代码。

Francine Evans
写了 GPL 协议的三正条纹。

http://www.cs.sunysb.edu/~stripe/

Oscar Everitt
从 FS98 游戏的 F4U 包里，创建了单发活塞发动机的发动机声音。它们非常棒而且 Oscar 也非常高兴可以贡献到我们的小项目中。

Bruce Finney
贡献了一些补丁以支持 MSVC5。

Olaf Flebbe
增强了 Windows 下的构建系统并提供了预编译的依赖。

Melchior Franz
贡献了游戏杆苦力帽支持，LED 字体，增强的 telnet 和 HTTP 接口。值的注意的是还帮助发现了 *FlightGear*、*SimGear* 和 *JSBSim* 的内存泄漏。

Jean-loup Gailly 和 **Mark Adler**
zilib 库的作者。用于飞行时的压缩和解压缩程序。

http://www.gzip.org/zlib/.

Mohit Garg
贡献了此指南。

Thomas Gellekum
修改并更新了 FreeBSD 下的编译。

Neetha Girish
贡献了可用 xml 配置的 HUD。

Jeff Goeke-Smith
贡献了我们第一个自动驾驶仪（航向保持）。为外部时区/关照变量提供更好的 autoconf 检查。

Michael I. Gold
耐心地回答有关 OpenGL 的问题。

Habibe
修改了 SimGear 的 RadHat 包构建。

Mike Hill
响应我们的诉求使用其非常棒的飞机，可以从这里获得:

http://www.flightsimnetwork.com/mikehill/home.htm,

Erik Hofman
大幅修改和声音模块相关的参数，以便在运行时使用飞行器特由的声音配置模块。贡献了 SGI IRIX 支持（包括二进制文件）和一些很棒的贴图。

Charlie Hotchkiss
提升并增强了 HUD相关代码。大量代码风格和代码优化。

Bruce Jackson（NASA）
受雇于 NASA 开发了 LaRCsim 代码，提供给我们的飞行模型。Bruce 也回答了大量大量的问题。

Maik Justus
增加了直升机支持，为 YASim FDM 提供起落架/地面交互以及滑翔机/铰链之间的交互。

Ove Kaaven
贡献 Debian 二进制文件。

Richard Kaszeta
贡献了屏幕缓冲区到 ppm 截屏程序。也帮助了开发早期“高度保持和自动驾驶模块”，教会 Curt Olson 基本控制理论并帮助他编码和调试早期版本。Curt 的“老板”Bob Hain 也贡献其中。更多详情：

http://www.menet.umn.edu/~curt/fgfs/Docs/Autopilot/AltitudeHold/AltitudeHold.html.

Rich 的主页

http://www.kaszeta.org/rich/.

Tom Knienieder
将音频库首先移植到 OpenBSD 和 IRIX，然后又移植到 Win32。

Reto Koradi
帮助设置雾效果

Bob Kuehne
重构了 Makefile 系统使其更简单而可维护。

Kyler B Laird
贡献并修正了此指南

David Luff
大量贡献到 IO360 活塞发动机建模。

Sam van der Mac
贡献此指南帮助将 HTML 转成 Latex。

Christian Mayer
为 fgfs 的技术演示贡献多国语言转换工具。贡献了一些代码以便读取微软模拟飞行的地景贴图。Christian 正忙于全新的天气子系统。为此项目贡献了热气球建模。

David Megginson
贡献代码让鼠标输入可以控制视角。经济上贡献了硬盘空间为 FlightGear 项目使用。更新了 README.running 文件。贡献于无贴图的 fgfs 和 ssg。也增加了 2D 仪表和保存/载入支持。另外，它还开开发了全新的仪表板代码，对 OpenGL 更友好，还有新的贴图。开发了属性管理器并贡献了游戏杆支持。贡献了生成随机地面物体。

Cameron Moore
FAQ 的维护者。现任的邮件列表管理员。提供 man 手册。

Eric Mitchell
贡献了顶尖的地景贴图，所有原始的都是他创建的。

Anders Morken
欧洲网站的前任维护者。

Alan Murta
创建了多边形裁剪库。

http://www.cs.man.ac.uk/aig/staff/alan/software/

Phil Nelson
GNU dbm 的作者，一套可以用来扩展散列的数据库程序，与标准 UNIX 下的 dbm 很相似。

Alexei Novikov
创建了欧洲地景。贡献了一个脚本可以把 fgfs 地景转换成 2D 地图。写了第一版地景创建手册。

Curt Olson
此项目的主要组织者。
首先实现并修改了 LaRCsim。
除此之外，将分散各处的地景子系统和其他图形资源组合在一起。其主页：

http://www.menet.umn.edu/~curt/

Brian Paul
我们使用了他的 TR 库，当然还有 Mesa：

http://www.mesa3d.org/brianp/TR.html, http://www.mesa3d.org

Tony Peden
贡献了飞行模型的开发，包括了基于 LaRCsim 的塞斯纳 172，贡献了 *JSBSim* 的初始状态代码，更加完整的标准大气模型，以及其他 bug 修复和补充。

Robin Peel
维护了 *FlightGear* 和 X-Plane 的全球机场和跑道数据库。

Alex Perry
贡献了更加精确的代码以便为 VSI、DG 和高度表建模。在邮件列表里建议和提高了模拟器的外观并帮助撰写文档。

Friedemann Reinhard
开发了早期贴图的仪表板。

Petter Reinholdtsen
并入了 GNU 的 automake/autoconf 系统（libtool）。为所有类 UNIX 平台简化和标准化了构建过程。对 IDE 的影响比较小，因为它不使用 UNIX make 系统

William Riley
贡献代码增加了“制动”。同时也写了第一版对超过两个轴的游戏杆支持。基于 VMap0 数据创建地景。

Andy Ross
贡献了新的可配置的 FDM *YASim*（Yet Another Flight Dynamics Simulator），基于几何信息，而不是空气动力系数。

Paul Schlyter
提供了 Durk Talsma 及所有我们写天文学代码相关的信息。Schlyter 先生也很高兴回答天文学相关的问题。

http://www.welcome.to/pausch/

Chris Schoeneman
贡献了音频支持。

Phil Schubert
贡献了各种贴图和发动机建模。

http://www.zedley.com/Philip/.

Jonathan R. Shewchuk
Triangle 程序的作者。Triangle 用来计算德劳内三角化和我们的不规则地形。

Gordan Sikic
为 LaRCsim 贡献了切诺基飞行模型。现在已经无法工作并需要调试。使用配置选项 `--with-flight-model=cherokee` 来编译切诺基而不是塞斯纳。

Michael Smith
贡献了驾驶舱图形、3D 建模、商标和其他图片。Bonanza 项目。

Martin Spott
此指南的联合作者。

Jon Stockill
维护了一个物体和其位置的数据库以便推广到全球地景。

Durk Talsma
更加精确的太阳、月亮和行星。太阳可根据天空的位置修改颜色。月亮有正确的圆缺以及天空的位置。行星的位置做了调整。还帮助了时间函数、GUI 和其他一些。贡献了 2D 的云层。网站：

http://people.a2000.nl/dtals/.

UIUC ——航空宇航工程系
贡献了修改版的 LaRCsim 以便可以从文件载入飞行器参数。这些修改是结冰研究的一部分。他们负责编码并让一切可以工作：

Jeff Scott

Bipin Sehgal
Michael Selig 同时，他们帮助了此项目：

Jay Thomas
Eunice Lee
Elizabeth Rendon
Sudhi Uppuluri

美国地质勘探局
提供了此项目使用的地质数据。

http://edc.usgs.gov/geodata/

Mark Vallevand
贡献了一些解析 METAR 的代码，以及一些 win32 屏幕打印程序。

Gary R. Van Sickle
贡献了一些早期的 GameGLUT 支持和其他修正。他还帮助对二进制文件做解析，相关：

http://www.woodsoup.org/projs/ORKiD/fgfs.htm.

他的 Cygwin Tips 也许对你也有用

http://www.woodsoup.org/projs/ORKiD/cygwin.htm.

Norman Vine
为“FlightGear 社区”提供了大量网址。许多性能优化整个代码。对地景生成也贡献了大量代码。很多 Windows 相关的贡献。贡献了 wgs84 测距和航道程序。贡献了基于 wgs84 的圆形航路的自动驾驶仪模式。很多其他 GUI、HUD 和自动驾驶仪相关的贡献。贡献代码允许鼠标输入来控制视角的方向。拼接屏幕转储。贡献了初始的“Goto Airport”和“Reset”代码以及初始的 HTTP 图片服务器代码。

Roland Voegtli
贡献了图片级的贴图。他是 X-Plane 欧洲地景项目的创始人：

http://www.g-point.com/xpcity/esp/

Carmelo Volpe
移植 *FlightGear* 到 Metro Works 开发环境（PC/Mac）。

Darrell Walisser
贡献了大量代码移植 *FlightGear* 到 Metro Works 开发环境（PC/Mac）。最后导致第一次苹果电脑移植，也贡献了入门指南的苹果电脑部分。

Ed Williams
贡献磁场相关代码（使用 Nima WMM 2000）。我们也从 Ed 那里多次借用了其航空定式。他的网站：http://williams.best.vwh.net/.

Jim Wilson
主要修正了查看器代码，使其更加灵活和容易建模。贡献了很多小的修正和 Bug 报告。贡献到了 PUI 属性浏览器和自动驾驶仪。

Jean-Claude Wippler
MetaKit 的作者——一个具有适当数据格式的可移动、嵌入式的数据库，*FlightGear* 之前使用过。下面的网址可活动更多信息：

http://www.equi4.com/metakit/

Woodsoup Project
虽然 *FlightGear* 不再使用 Woodsoup 的服务，不过我们依旧感谢曾经提供的托管服务。因为提供的计算资源和服务，*FlightGear* 才有了真的家。

http://www.woodsoup.org/

Robert Allan Zeh
提供了异常多的有关 Cygnus Win32 编译的帮助，并指出如何链接 dll 文件。没有他的帮助，第一个可运行的 Win32 版本的 *FlightGear* 就会出现。

其他
下面这些独立开发者对地景数据库做出贡献：Jon Stockill, Martin Spott, Dave Martin, Thomas Foerster, Chris Metzler, Frederic Bouvier, Melchior Franz, Roberto Inzerillo, Erik Hofman, Mike Round, Innis Cunningham, David Megginson, Stuart Buchanan, Josh Babcock, Esa Hyytia, Mircea Lutic, Jens Thoms Toerring, Mark Akermann, Torsten Dreyer, Martin C. Doege, Alexis Bory, Sebastian Bechtold, Julien Pierru, Bertrand Augras, Gerard Robin, Jakub Skibinski, Morten Oesterlund Joergensen, Carsten Vogel, Dominique Lemesre, Daniel Leygnat, Bertrand Gilot, Morten Skyt Eriksen, Alex Bamesreiter, Oliver Predelli, Georg Vollnhals 和 Paul Richter。

B.3 尚未完成的事业

如果你一直读这本指南到此处，你也许会同意：*FlightGear*，即便是现在的状态，也不是最完美的。它已经是一个飞行模拟器的大杂烩了，包括很多可选择的飞行模型，各种带有仪表的飞机甚至一款 HUD，地形地景，贴图，所有基本控制和天气。

尽管如此，*FlightGear* 需要继续开发。不仅包括内部优化，还有很多 *FlightGear* 的领域需要基础的提高和开发。首先的方向是增加机场、建筑物和更多增加真实性的机场和城市地景。另一项任务是继续完善菜单系统，也许不会太难。很多通过命令行或甚至要通过编译选项来提供的参数，最终都会进入菜单项。最后，*FlightGear* 到目前为止依旧缺乏 ATC 程序。

已经有很多人在这些方向上努力了。如果你是一名程序员或者你觉得自己可以做出贡献，邀请你贡献进来。

致谢

显然，这篇文档如果没有上面提到的这些贡献者的努力是不可能写就的，他们让 *FlightGear* 更加真实。

首先，我非常高兴的看到 Martin Spott 加入文档写作。Martin 不仅提供了大量 Linux 相关的更新（特别是 OpenGL 相关），而且还为文档的写作提供了思路。

此外，还要特别感谢 Curt Olson，他收集了大量零散的 README 文件、Thanks 文件、网页和私人邮件，让我自由利用，以便完成此小册子。

还要感谢 Bernhard Buckel 写了多篇早期版本的指南，并贡献了很多思路。

Jon S. Berndt 通过几个版本的文档关键校对，指出不一致的地方，并提出改进建议。

同时，我从 Norman Vine 那里获得大量的帮助。也许没有他的解答我无法制服不同版本的 *Cygwin* 和 *FlightGear* 这一对欢喜冤家。

非常高兴，Mac 专家 Darrell Walisser 贡献了在 Mac OS X 下编译的相关章节。另外他还提供了很多 Mac 相关的提示和修正。

特别鸣谢他们提供的贡献和捐助：John Check（主要布局）、Oliver Delise（一些建议包括章节注释）、Mohit Garg（OpenGL）、Kyler B. Laird（修正）、Alex Perry（OpenGL）、Kai Troester（编译问题）、Dave Perry（游戏杆支持）和 Michael Selig（UIUC 建模）。

除那些我突然想不起来的名字以外，还要感谢下面这些贡献到此版本 FlightGear 指南中，修复了一些 “Bug”（随机顺序）：Cameron Moore、Melchior Franz、David Megginson、Jon Berndt、Alex Perry、Dave Perry、Andy Ross、Erik Hofman 和 Julian Foad。

译后记

中文翻译项目的地址：https://github.com/tonghuix/getstart-zh

因为《*FlightGear* 手册》涉及到大量民航基本术语和相关飞行技术，我在翻译时力求准确和专业，因此翻译过程中参考了一些民航相关书籍文献，互联网上的文章和我本人在模拟飞行方面的经验。其中由西南交通大学出版社出版，张泽龙主编的《私用飞行员教程》、《商用飞行员教程》，航空工业出版社出版的《飞行员航空知识手册》等是主要参考来源。翻译教程时，个别地方亦有参考中国民航出版社出版的《飞行快速心算》一书。此外在翻译过程中也查看了民航局的互联网公开资源等。除此之外还有参考一些非授权资源，如塞斯纳 172P 的飞行手册，Boeing 737-800 飞行手册、机组训练手册等。

此《*FlightGear* 手册》中文版会跟着 *FlightGear* 官方英文手册代码一起更新，随时增加新版本的新特性，以利更好的推广 *FlightGear* 这个自由开源的飞行模拟平台。

欢迎在翻译项目的 Github 页面提交对我翻译的指正和意见，我会第一时间做出修改。也欢迎愿意贡献到此手册翻译的人士，直接提交 Pull Request! 也可以通过电子邮件联系我 tonghuix fedoraproject.org

——译者：佟辉
2016 年 1 月于北京

索引