

Designing Interfaces
Patterns for Effective Interaction Design

Designing Interfaces

中文版



O'REILLY®

Jenifer Tidwell 著
De Dream' 译

 電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

Designing Interfaces 中文版



“非常优秀！这绝对是UI设计模式领域最好的书，也是近年来交互设计领域最好的新书之一。本书清楚、简洁、引人入胜，既包括基础知识，又讲述了复杂的概念，既适合初学者，又适合专家和研究人员。本书超越了很多琐碎的‘模式语言’——它们提到的是显而易见的各种细小折衷——而本书讲述了交互设计领域许多真正的两难选择。本书不只是收集了一组模式——事实上，这是一本关于现代视觉和交互设计的完整教程。任何设计师都值得一看。”

——Larry Constantine, IDSA, 获奖设计师及《Software for User》的作者之一

“本书采纳了界面设计领域的智慧和经验，为所有人提供了值得学习的内容。它已经成了我的设计工具集里不可缺少的一部分。现在，当我设计的时候有四个必备元素：我的手写板、Photoshop、Flash以及这本书。它帮助我组织头脑风暴，并为我的工作提供批评意见。我愿意向任何与创建界面有关的人士推荐这本书。”

——Paul Hoover, 微软移动PC部门的产品设计师

每一天，越来越多的人们依赖交互式的软件——系统、Web应用、手机，以及其他数字设备。当这些软件设计良好的时候，人们将更开心、更安全、更有效率。好的界面设计增进了用户忠诚度，减少了客户服务支出，可以帮助产品从竞争对手中脱颖而出。

你想设计出有吸引力、容易使用的界面，但又对自己的能力还不太确定吗？这本书将是你的帮手。本书从很多各不一样的资源吸取了设计智慧：多年的pre-Web应用设计、最好的交互式网站，以及移动设备（例如手机和iPod）。书中有很多你可以重用的设计想法。你不用再从头开始了。

这些设计想法表现为一组模式——对于常规设计问题的解决方案，并可以根据具体情况进行裁剪。每个模式包含切实可行的设计建议，供你即刻投入使用。每个模式还提供了各种全彩的例子。另外，每一章还讲述了交互设计和视觉设计中的各个关键概念。本书的主题包括：

- 应用的信息架构
- 表单
- 导航
- 图形编辑器
- 页面布局
- 颜色、排版、外观和感觉
- 地图、图表和表格

本书提供了许多有价值的资源，供交互设计师、软件开发人员、图形设计师，以及任何与创建带用户界面软件相关的人士阅读。当你在寻求解决方案的时候，学习某种特别技术的时候，或者在你只想得到一点有创意的帮助的时候，别忘了打开本书。

Jenifer Tidwell是MathWorks公司的一名交互设计师和软件工程师。MathWorks公司位于马萨诸塞州，波士顿附近，是一家技术计算软件提供商。从1997年以来，Jenifer一直在研究用户界面模式，而早在1991年，她就已经开始设计和建造复杂的软件系统和Web界面了。

图书分类：交互设计

策划编辑：方舟

责任编辑：周筠 何艳

项目管理：梁晶



Broadview®

www.phph.com.cn

网上订购：www.dearbook.com.cn

第二书店·第一服务

www.oreilly.com

ISBN 978-7-121-04797-8



9 787121 047978 >

定价：80.00元

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

O'Reilly Media, Inc. 授权电子工业出版社出版

此简体中文版仅限于中国大陆（不包含中国香港、澳门特别行政区和中国台湾地区）销售发行

This Authorized Edition for sale only in the territory of People's Republic of China (excluding Hong Kong, Macao and Taiwan)

ISBN 978-7-121-04797-8

9 787121 047978 >

内 容 简 介

本书清楚、简洁、引人入胜，以设计模式的形式总结处理界面设计法则，展现了常见设计问题的解决办法及其在实践当中的运用。每个模式包含了您可以立刻取用的实务建议，并用全彩方式展现了运用技巧。您能够从本书中获得设计建议、可替代的设计方案，以及何时不宜采用的警示，可以根据自身情况灵活运用。另外，每章的介绍部分描述了交互设计和视觉设计中的各个关键概念，这些叙述让您深入理解模式的精髓，如何更加科学地应用模式。

资深设计师可将本书用做汇集了设计构思的参考手册。初级设计师则可以按照书中的指导直接使用这些模式，找到一条通往界面和交互设计领域的快捷之路。

0-596-00803-1 Designing Interfaces Copyright © 2006 by O'Reilly Media, Inc.

Simplified Chinese edition, jointly published by O'Reilly Media, Inc. and Publishing House of Electronics Industry, 2008. Authorized translation of the English edition, 2006 O'Reilly Media, Inc., the owner of all rights to publish and sell the same.

All rights reserved including the rights of reproduction in whole or in part in any form.

本书中文简体版专有出版权由O'Reilly Media, Inc.授予电子工业出版社，未经许可，不得以任何方式复制或抄袭本书的任何部分。

版权贸易合同登记号 图字：01-2007-0662

图书在版编目（CIP）数据

Designing Interfaces中文版 / (美) 泰德维尔 (Tidwell,J.) 著；蒋芳译. —北京：电子工业出版社，2008.1

书名原文：Designing Interfaces

ISBN 978-7-121-04797-8

I. D... II. ①泰... ②蒋... III. 用户界面—程序设计 IV.TP311.1

中国版本图书馆CIP数据核字（2007）第117263号

责任编辑：周筠何艳

项目管理：梁晶

封面设计：Mike Kohnke 张健

印 刷：北京画中画印刷有限公司

装 订：

出版发行：电子工业出版社

北京市海淀区万寿路173信箱 邮编 100036

开 本：889×1194 1/16 印张：22.75 字数：500千字

印 次：2008年1月第1次印刷

定 价：80.00元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：
(010) 88254888。

质量投诉请发邮件至zjts@hei.com.cn，盗版侵权举报请发邮件至dbqq@hei.com.cn。

服务热线：(010) 88258888。

O'Reilly Media, Inc.介绍

为了满足读者对网络和软件技术知识的迫切需求，世界著名计算机图书出版机构O'Reilly Media, Inc. 授权电子工业出版社，翻译出版一批该公司久负盛名的英文经典技术专著。

O'Reilly Media, Inc.是世界上在UNIX、X、Internet和其他开放系统图书领域具有领导地位的出版公司，同时也是联机出版的先锋。

从最畅销的《The Whole Internet User's Guide & Catalog》（被纽约公共图书馆评为20世纪最重要的50本书之一）到GNN（最早的Internet门户和商业网站），再到WebSite（第一个桌面PC的Web服务器软件），O'Reilly Media, Inc.一直处于Internet发展的最前沿。

许多书店的反馈表明，O'Reilly Media, Inc.是最稳定的计算机图书出版商——每一本书都一版再版。与大多数计算机图书出版商相比，O'Reilly Media, Inc. 具有深厚的计算机专业背景，这使得O'Reilly Media, Inc. 形成了一个非常不同于其他出版商的出版方针。O'Reilly Media, Inc. 所有的编辑人员以前都是程序员，或者是顶尖级的技术专家。O'Reilly Media, Inc. 还有许多固定的作者群体——他们本身是相关领域的技术专家、咨询专家，而现在编写著作，O'Reilly Media, Inc. 依靠他们及时地推出图书。因为O'Reilly Media, Inc. 紧密地与计算机业界联系着，所以O'Reilly Media, Inc. 知道市场上真正需要什么图书。

联系博文视点

您可以通过如下方式与本书的出版方取得联系。

读者信箱：reader@broadview.com.cn

投稿信箱：bviougao@gmail.com

北京博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 吴家湾 邮科院路特1号 湖北信息产业科技大厦1402室

邮政编码：430074

电话：(027) 87690813 传真：(027) 87690813转817

若您希望参加博文视点的有奖读者调查，或对写作和翻译感兴趣，欢迎您访问：

<http://bv.csdn.net>

关于本书的勘误、资源下载及博文视点的最新书讯，欢迎您访问博文视点官方博客：

<http://blog.csdn.net/bvbook>

模式意味着重用

处在软件的世界里，一切都和现实世界有些相同，又有些不同。例如，用Tab页来组织内容，对内容进行快速索引，对空间进行扩展，这些与现实中Tab标签的使用几乎是完全相同的。而有些方面，例如软件屏幕的布局和纸质页面的布局就不尽相同：单拿屏幕尺寸来说，有点像横着摆放的纸张；另外，适合在屏幕上和平面媒体上使用的字体也是不一样的。

如果人拿到一叠纸，他会很自然一张张地翻过去，翻到最后，然后放下。然而，在软件世界里，如果有人到了这个界面之后无法返回，就会被困在那里，不知道该怎么办（在早些年的游戏《风云》中，快到结束的时候就有这么一个地方，害得很多玩家被困）。在这种情况下，应该怎么做呢？当然是一定要留一个出口，让用户可以回到安全的地方咯。

在每个限制了导航选择的页面上放一个按钮或链接，让用户能明白无误地离开该页面，回到一个熟悉的地方，这就叫逃生舱（Escape Hatch）模式。

模式意味着重用。例如，假设有一个网站的导航很复杂，为了避免访问者迷失在各种各样的链接里，我们可以应用逃生舱模式：在网页上设置统一的出口，通常是把首页的链接加在站点图标上，无论什么时候，单击站点图标就可以回到首页，重新开始，而且现在，这样做已经成为设计网站的惯例了。

在本书中，作者为我们收集了很多常用的界面设计模式，例如，如何组织内容，如何设计导航，等等。可能我们对其中的一些已经很熟悉了，也有一些是以前没有细想过的，在阅读本书的时候，正好可以一一梳理。总地来说，这些模式都是在实现级别常用的各种解决方案，你可以灵活运用于自己的界面设计当中。

不过，在得到任何解决方案之前，你需要关注问题。而那些问题通常是跟人有关的，例如，人的行为模式，人的心理特点，等等。所幸的是，作者在本书的开篇就专门总结了一章“和人有关”的模式，所以，我们在阅读和学习各种界面设计模式的时候，也不要忘了：“因为好的界面设计并非始于图片，而是始于对人的理解：人们喜欢什么，为什么人们会使用某种特定的软件，他们可能会怎样与之交互。对使用的人了解得越多，你就越能使他们移情，也就越能进行有效的设计。毕竟，软件对于使用它们的用户来说只是达到目的的一种手段，你越能帮助他们达到那些目的，他们就越高兴。”

感谢电子工业出版社北京博文视点资讯有限公司，也感谢本书的策划编辑方舟，他在背后为这本书做了很多工作。翻译和出版一本书并不是一件容易的事，不过，前面这些繁琐的工作我们都已经完成了，剩下的，就是你，一位好奇的读者，坐下来，慢慢享受阅读的时间了。

Windy (from De Dream')

2007年7月于北京

关于作者

ABOUT THE AUTHOR

Jenifer Tidwell是技术计算软件厂商MathWorks公司的一名交互设计师和软件工程师。她擅长设计和开发数据分析及可视化工具，最近正在为MATLAB的数据工具进行新的设计。全世界有很多研究人员、学生和工程师使用MATLAB来开发汽车、飞机、蛋白质和宇宙理论等。她在网站设计方面也很有名，很早的时候就是一名RIA（Rich Internet Application）技术的倡导者，并在2000年初协助设计和开发了Curl。

Jenifer在麻省理工学院接受技术教育，并在马萨诸塞州艺术学院学习设计，但她一直没有停止学习。她从1997年开始研究用户界面模式。Jenifer热爱摄影和写

作，并热衷于在新英格兰地区进行户外活动——骑车、划船、滑雪、攀岩。她的个人网站地址是：<http://jtidwell.net>。

封面说明 COLOPHON

O'Reilly特有的封面设计风格，主要来自读者的建议。我们自己的实验及发行渠道的反馈。唯有与众不同的封面，才能突显我们对技术主题独树一帜的阐述风格，避免读者被枯燥的主题压得喘不过气。

本书封面上的动物叫做鸳鸯（Mandarin duck，学名Aix Galericulata），是鸭科中最漂亮的鸟种之一，原产地在中国，现在生活在俄罗斯东南部、中国北部、日本、英格兰南部和西伯利亚地区。

鸳鸯的雄鸟有着华美艳丽的羽毛，色彩斑斓带着金属光泽的羽冠、栗色的两颊，长长的白色眉纹从鲜红的鸟喙一直延伸到脑后。雌鸟的羽毛没有这么鲜艳，呈灰色、白色、褐色或绿褐色，颈部前面有白色的羽毛。

它们生活在靠近河流和湖泊的森林地区，属于杂食鸟

类，所吃的食物随季节变化：在秋季进食果实和谷物，在春季进食昆虫、蜗牛、水生植物，在夏季进食蠕虫、蚱蜢、青蛙和一些软体动物。

鸳鸯的求偶以一种复杂而精细的求爱舞蹈开始，包括摇摆运动、模仿喝水的姿势，还有梳理羽毛等动作。每一只雄鸟都要互相比赛来赢得雌鸟的喜爱，并由雌鸟来决定最后的对象。鸳鸯的幼鸟跟随母亲生活，因为幼鸟的猎食者包括水獭、貉、貂、臭鼬、鹰和草蛇等，雌鸟常常会假装受伤引开这些敌人的注意力，从而来保护幼鸟。

鸳鸯并不是濒危动物，但它们的生存也会受到威胁，伐木工人不断侵蚀它们的地盘，猎人和偷猎者常常觊觎它们的羽毛。它们的肉味道不好，所以人类并不把它们当作食物来捕杀。

前言

PREFACE

从前，界面设计师们只有一个可怜的小破工具箱。

那个时候，我们只有几种简单的控件：文本输入框、按钮、菜单、小图标和模态对话框。我们小心翼翼地把它们按Windows或Macintosh的界面风格指南组织到一起，然后希望用户能理解由此产生的使用界面。当然，很多时候事与愿违。同时，我们也在为有限的屏幕、不多的颜色、低效率的CPU，还有慢吞吞的网络（如果能连接上的话）设计。那个时候的界面是灰蒙蒙的。

三十年河东，四十年河西，世界不一样了。现在进行设计，我们有了比以前多得多的组件和想法，有了更多的用户界面工具包，例如Java™ Swing、Qt、HTML、Javascript、Flash，还有数不清的开源选择。苹果和微软自己的UI工具包也比以前更丰富更漂亮。显示技术更先进了，Web应用常常看起来像嵌入它们的网站一样专业。在桌面UI习惯用法（例如拖拽）逐渐集成到Web应用的同时，一些Web用法也慢慢迁移到桌面应用中，例如蓝色的下划线链接、后退/前进按钮、大胆的字体和背景图片，还有好看的、不再灰蒙蒙的配色方案。

尽管如此，现在要设计出好的用户界面仍然不容易。假设你不是一名受过训练或专门自学过的界面设计师，如果你只是按原来的方式去使用UI工具箱，并且遵循已有的界面风格指南或模仿已有的应用，也许能设计出普普通通还算OK的界面。

但是，那样的界面已经不够了。用户的期望比以前更高了——如果不是一开始就很容易上手，用户就不会觉得它很好。即使界面符合所有的标准，也可能没有很好地理解用户习惯的工作过程，或者使用了错误的词汇，又或者让用户很难猜出软件到底是做什么用的。没有耐心的用户甚至连问题都不会向你提。更糟糕的是，如果你开发了一个不可用的网站或Web应用，失望的用户只要按一下按钮就可以转而投向你的竞争对手。因此，就算设计一个普通的用户界面，带来的不良影响也比以前更大了。

如果你在设计桌面和Web之外的产品，形势会更加严峻。因为在那些领域没有多少好的设计建议，掌上电脑、手机、汽车导航系统、数字电视录像机——设计师们还在根据一些基本的原则试图找出什么方案可行，什么方案不可行，没有成熟的设计（他们的用户常常可以忍受难用的界面，但是也不会忍受太久了）。

像电视、电话、汽车仪表盘这样的设备，以前曾经是工业设计师们的专有领地。但现在，这些设备变得越来越智能，它们由功能越来越强大的计算机驱动。为了响应市场的呼声，基于软件的功能和应用也越来越多。不管好不好用，它们已经进入了这个领域。在这种情况下，良好的界面设计和交互设计可能是十年之内设计师集体智慧唯一的希望。

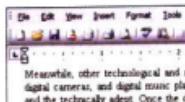
小片的界面，松散地结合

SMALL INTERFACE PIECES, LOOSELY JOINED

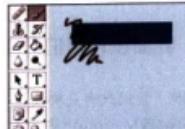
作为一名想要弄清楚最近这些年所有技术变化的界面设计师，我看到了界面设计工艺上两个大的影响趋势。一是界面习惯用法的增长；界面习惯用法是一些可以识别的界面类型或风格，关于它们的目标、动作和外观，每个习惯用法都有它自己的词汇。大家也许能认出下面图中所有的习惯用法，同时还有更多的习惯用法正在产生之中。



表单



文本编辑器



图形编辑器

	A	B	C
1	Time	XaY	XaY
2	0	-1.2071	0
3	5.12E-09	-201.13	-5.7E-05
4	1.54E-08	1.13	-9.8E-04
5	3.8E-08	-1.2071	-1.16E-03
6	2.18E-07	1.13	-2.56E-01
7	1.12E-06	-1.2071	-3.4E-19
8			
9			

电子表格软件



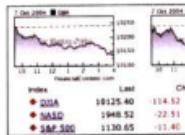
浏览器



日历



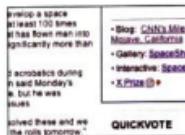
媒体播放器



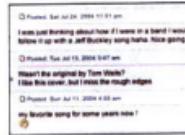
信息图表



游戏



网页



社交空间



电子商务网站

一些代表性的界面使用法

第二个影响是把这些界面组织到一起的规则开始日益放宽。例如，在一个界面上看到几个习惯用法混合在一起不再让人觉得诧异，又或者，这些控件的一部分和其他控件的一些部分同时共存于一个页面上。在线帮助页面，以前一直是超链接文本的格式。现在也有了一些交互性的脚本，例如动画或者到公告板的链接。界面上可能有自己的帮助文本，分布在表单或编辑器的中间。这在以前非常罕见。组合框的下拉菜单里不再只有标准的文本行，取而代之的可能是让人意想不到的元素，例如颜色网格和滑块（sliders）。Web应用看起来像以文档为中心的画图程序，但是没有菜单条，并且只把完成的工作保存到某个地方的数据库里。

Web页面的自由度似乎已经教会用户通过图片和交互性来扩展他们的期望。只要用户能弄清楚界面在做什么，现在要打破旧式Windows风格指南的规定就没什么问题了。

不过，那也是最难的部分。一些应用、设备、Web应用容易使用，但也还有很多不是这样的。遵循界面风格指南从来不能保证可用性，但现在设计师们有了比以往更多的选择（反过来，它们也会让设计难度更高）。那么，什么样的界面容易使用？

一些人会说：“容易使用的界面应该很直观。”噢，是的。不过“直观”似乎只是“容易使用”的同义词。

还有，“直观”这个词带有一点欺骗性。Jef Raskin曾经指出，当我们说到软件“直观”时，我们实际上是在说“熟悉”。鼠标对于没有见过它的人来说一点都不直观（尽管可能一只灰熊会这么觉得），人脑中没有什么先天的或本能的因素能知道它是什么，但一旦你花上10秒钟学习如何使用鼠标后，它就变得有熟悉感了，而且你再也不会忘记。蓝色的带下划线文本是这样，播放/暂停的按钮也是这样，等等，诸如此类皆是。

改述一下：“容易使用的界面，要设计得让人有熟悉感。”

现在我们有点感觉了。熟悉并不一定意味着某个给定的应用要跟已经定义好的产品流派相同（例如Word、Photoshop、Mac OS或随身听）。人类非常聪明，只要应用的各个部分容易识别，而且部分与部分之间的关系清楚，人们就能将已有的知识应用到一个前所未有的界面上，并且能弄明白它是怎么用的。

这就是模式的来源。本书列出了很多这类熟悉的界面组成部分，并且以一种你可以重用到各种不同上下文中的方式组织到一起。模式捕捉了一种常用的结构——通常是非常“本地化”的结构，就像新奇的组合框下拉元素一样——不会在细节上太过具体，这样能让你有发挥创意的空间。

如果知道用户对你的应用有何种期望，而且能从界面工具箱中仔细选择习惯用法（大规模的）、控件（小规模的）和模式（在整个范围内），你就能设计出某个让人觉得有熟悉感同时保持新颖独特的界面。

这样，你就两全其美了。

模式综述

ABOUT PATTERNS IN GENERAL

基本上，模式是某个对象的结构和行为上的特点，这些特点可以改进它的“可居住性（habitability，适应人的能力）”——这个对象可能是一个用户界面、一个网站、一个面向对象的程序，甚至一栋建筑物。模式让目标对象更漂亮或更容易被人们理解，也会让工具更有用、更可用。

因此，模式可以描述成某个已有设计领域的最佳实践。它们为设计张力（通常在模式文献中叫“约束”，Forces）提供普遍适用的解决方案。根据这样的定义来看，它们并不是新颖独到的。它们不是即取即用的组件，同一个模式的每种实现各不相同。它们不是简单的规则，也不是启发式的道理，也不会为你提供所有的设计决策依据——如果你在寻找一套完整的界面设计方法，那么你找错地方了，因为本书是一份界面模式目录！

因为模式的部分价值在于在不同的设计上下文中解决设计张力问题，所以本书只是把模式描述为不同设计问题提供的解决方案。例如，若需要把一大堆素材打包到一个小的空间里，设计师可以使⽤一个叫做卡堆栈（Card Stack）的模式。在该模式之外，需要设计师自己处理信息架构——如何把内容切分成小块、如何命名等。还有，卡堆栈完成之后应该是什么样子，也不是由模式决定的。使⽤Tab、左侧的列表或数状视图？这些都由设计师自己决定。

有一些非常完整的模式集合组成了“模式语言”。这些模式与可视化语言类似，因为它们涵盖了整个设计中用到元素的词汇（虽然模式语言更抽象，也更注重行为，而可视化语言讲的是形状、图标、颜色、字体等）。这个集合不那么完整，它包含的技术也没有达到传统模式的要求，但它非常简洁，可以管理，并且非常有用。

其他模式系列

OTHER PATTERN COLLECTIONS

最早的时候，“模式”一词源于建筑领域，而不是软件行业。克里斯多夫·亚力山大的《建筑模式语言》（*A Pattern Language*）和它的姊妹书籍《建筑的永恒之道》（*The Timeless Way of Building*，两本书都是牛津大学出版社出版的）确立了模式的概念，并介绍了一门包括250个模式的多层次模式语言。因为它的完备性、模式之间丰富的关联，以及它缘自人类对建筑世界的回应这个事实，所以它被誉为模式语言的金科玉律。

在20世纪90年代中期，《设计模式》（*Design Patterns*, “Addison-Wesley”出版，作者是Erich Gamma、Richard Helm、Ralph Johnson和John Vlissides）一书的出版，完全改变了商业软件架构实践。该书包括一组描述面向对象微架构的模式，如果你学习过软件工程，这可能就是那本把你引入模式世界的书。后来，许多其他作者也开始写书叙述软件方面的模式，这些模式确实让软件更能适应人了——只不过针对的是那些写软件的人，而不是使用软件的人！

第一组关于用户界面的模式是本书这些模式的祖先：“Common Ground”¹。后来又出现了很多其他模式和语言，最有名的是Martijn van Welie的“交互设计模式（Interaction Design Patterns）”²，还有Jan Borchers的《交互设计的模式方法》（*A Pattern Approach to Interaction Design*，“Wiley”出版）。最近，一部非常完备的介绍网站模式语言的书又诞生了，就是“Addison-Wesley”出版的《网站设计》（*The Design of Sites*）。我强烈推荐这本书，特别是如果你正在设计传统网站的话。如果你在进行Web或桌面应用设计，又或者你想在这两个领域有所突破，那么看看这些书籍和模式，你肯定会从中找到一些灵感。

关于本书里的模式

ABOUT THE PATTERNS IN THIS BOOK

由上所述，其实本书里没有什么新鲜内容。如果你曾经进行过任何Web或UI设计，或者哪怕只是对这两方面关注较多，你可能会对绝大部分的模式说：“噢，是的，我知道，就是那样的。”但是其中也有一些模式对你来说可能是新概念，还有一些虽然见过，也可能不在你的常规设计后备箱里。

这些模式适用于桌面应用，也同样适应于基于Web的应用，很多模式还可以用在如掌上电脑、手机、数字录像机等数字设备上。常规的网站可能也会从中得到好处，但我会在下节中重点谈论这个主题。

尽管本书不会事无巨细地描述之前说到的那些界面习惯用法，但它们是本书的组成部分。有三个章节会重点介绍常见的习惯用法：表单、信息图表，还有所见即所得的编辑器（WYSIWYG，就像那些文本和图形编辑器一样）。其他章节将讲述跨越多个习惯用法的主题，例如组织方式、导航、动作（actions），以及可视化风格。

本书的目标读者是那些懂得一定界面设计概念和术语的人士。最好能知道对话框、选择、组合框、导航条和空白等。书中并不会提到很多广为人知的技术（例如粘贴），因为大家都已经知道了。但是，冒着唠叨的风险，书里也讲述了一些常见的技术来鼓励大家把它们用在别的上下文里——例如，许多桌面应用应该更好地使用全局导航（Global Navigation），或者把它们和其他候选方案一起讨论。

本书并没有提供一份完整的界面设计过程，在进行设计时，合理的设计过程非常重要，尤其是以下几个特定的部分：

- 进行实地研究，来找出色目标用户和他们的行为
- 进行目标和任务分析，描述并阐明用户将会怎样对待你的产品
- 进行模型的设计，例如人物角色模型（Personas，一种用户模型）、情景（Scenarios，关于任务和交互情形的模型）及界面原型（Prototype，界面本身的模型）

1. http://www.mit.edu/~jtidwell/common_ground.html

2. <http://www.welie.com/patterns>

- 在不同的开发阶段进行经验测试，例如可用性测试和对真实用户使用产品的*in situ*（拉丁语，意为“现场”）观察
- 拥有足够的设计时间完成几个设计迭代，因为你通常要多试几次才能得到一份好的设计

关于设计过程的内容超出了本书讨论范围，而且其他地方有许多书籍和课程详细地讲到了它们，建议你学习一下那些内容，因为那些内容很不错。

不过，还有一个更深刻的理由来解释为什么本书不会给你设计界面的灵丹妙药。好的设计远远不止一些诀窍，它是创造性的过程，也会随着你的工作而发生变化——例如，在任何给定的项目下，你都有可能直到走进死胡同才会意识到其中的一些问题。

而且，设计并不是线性的活动。本书的大部分章节是按照规模（其次是它们在设计进程中的大致顺序）来排列的：首先对重大内容和范围进行决策，然后是导航、页面设计，最后是关于交互的细节，例如表单、画布。但你经常会发现需要在这个进程中来回往复。可能你在一个项目早期就已经制定某个屏幕应该是什么样子，然后它将成为一个“固定点”。你需要在它之前之后做一些工作来确定正确的导航结构（不，我们理想的状态并不是这样的，但是现实生活中常常会碰到这样的情形）。

也就是说，你可以采用以下这些方式来使用本书中的模式。

学习

如果你还没有丰富的设计经验，这些模式可以成为你良好的学习工具。你可能希望通过一遍来获得一些想法，或者在需要的时候回过头来参考某些特定的模式。正如扩大你的词汇面有助于帮助你在某种语言上的表达能力一样，扩大你的界面设计“词汇面”也可以帮助你更好地设计。

范例

本书中的每个模式至少有一个范例，有些模式有很多范例；它们也许可以成为有用的原始资料。你还可能在这些精彩范例中找到文字叙述中没有表达的闪光点。

术语

如果你向用户、工程师或经理们谈到界面设计，又或者你在写说明文档，那么可以用一些模式的名字来沟通和表达你的想法。这也是模式语言另一个著名的优点（例如，术语Singleton和Factory原来是模式的名字，现在已经在软件工程师中广为流传了）。

激发灵感

每个模式的描述都力图阐述为什么这个模式可以让某个界面更加容易使用或更有意思，如果明白了其中的理由，然后又想在本书的示例之外再发挥一下，那么你可以在有根有据的情况下得出一些创造性的想法，不必凭空猜测。

还有一点点提醒：模式目录不是检查清单，你不能指望用其中使用的模式来衡量一份设计的质量。每个项目都有它独特的上下文，在这种情况下，哪怕你是在解决一个常见的设计问题（例如怎样将很多很多内容塞到一个页面上），某个给定的模式对于该上下文来说可能并不是好的选择。没有任何参考资料能取代明智的设计判断，它们也不能取代良好的设计过程，因为好的设计过程将帮助你找到并修复设计中的错误。

你最终应该能将这样的参考资料置之脑后。在成为一名熟练的设计师以后，你将在实际情况中自然地使用这些设计想法，不必刻意去注意它们。模式将成为你知识体系的一部分。当然在那个时候，它们也会成为你设计工具箱的自然组成部分。

目标读者

AUDIENCE

不管你的用户界面设计能力如何，本书都可以为你提供帮助，特别是在以下领域进行设计的人士：

- 桌面应用
- Web应用，或者富网络应用（RIA，Rich Internet Application）
- 需要高度交互的网站
- 为手持设备、手机或其他消费电子产品设计软件
- 现成系统（Turnkey System），例如信息亭（Kiosk）
- 操作系统

上面的列表也包括传统的网站，例如公司或机构的网站页面，但我特意没有把重点放在网站上，关于网站的设计已经有了很多出版物和文献，在这里再详细叙述就有点多余。而且，它们中的绝大多数并不需要大部分模式中具备的交互性，也就是说，在一个只读网站和交互性网站之间，有着本质上的区别。

当然，上述设计平台彼此之间也有很大的差异，不过，我相信它们有更多的共同点。你将在这些模式中看到各种平台的范例，而且我是有意这么做的——它们经常使用同样的模式来达到同样的目的。

这本书不是设计101准则，它更像设计225手册。正如前面提到过的，希望你已经有了UI设计的基础知识，例如了解现有的工具箱和控件集合，希望你有拖拽和关于注意力焦点的概念，并懂得可用性测试和用户反馈的重要性。如果你缺乏这些知识，参考资料中有一些很不错的书，能帮助你更好地开始。

本书特别面向以下读者：

- 需要设计用户界面的软件开发人员。
- 需要设计更具交互性的Web应用或网站的网页设计师。
- 用户界面设计和可用性领域的新人行人员。

- 想要了解某些特定问题如何解决的有经验的设计人士。本书中的示例可以作为很好的灵感来源。
- 交叉领域的专家，例如技术文档编写人员、产品设计人员、信息架构师等。
- 想要进一步了解良好界面设计要素的经理们。
- 开源开发人员和爱好者。本书并不是什么“开源设计”，但是本书公开地为大家提供界面设计最佳实践作为参考。

本书组织方式

HOW THIS BOOK IS ORGANIZED

本书中的模式组成了几个主题章节，每个章节都有一个简短的介绍，包括了这些模式的概念。我想简单强调一下，其中一些概念可能需要一整本书来阐述它们。但是介绍部分将会给你一些关于模式的上下文知识。如果你已经知道了这些概念，那么可以在这里温习一下；如果不是，那么它们将给你提供一些可以进一步学习的主题。

第一部分的章节可以应用到几乎所有的界面，不管是桌面应用、Web应用、网站、硬件设备或任何你能想到的东西：

- 第1章，**用户做些什么**，讲述的是好的界面应该支持的常规用户行为和使用模式。
- 第2章，**组织内容**，讨论应用到高度交互界面的信息架构问题。这一章讲述了不同的组织模型、一个用户一次看到的内容数量，还有使用窗口、面板和页面的最佳方式。
- 第3章，**到处走走**，讨论导航问题。这一章描述了在界面中移动的模式——如何在页面之间、窗口之间及在大的虚拟环境下移动。
- 第4章，**组织页面**，描述页面元素布局和布置模式。它讲述的是如何简单地通过把设计元素放在正确的地方来进行与用户的交流。
- 第5章，**完成任务**，描述的是如何表现动作和命令，可以使用这一章里的模式来处理界面上的“动词”。

接下来的一组章节处理具体的界面习惯用法，可以一次把它们读完，但在实际的项目中可能不会全部用到。第6章和第7章中的模式是用得最广的（既然绝大部分的现代界面都流行使用树、表格和表单）。

- 第6章，**显示复杂数据**，包括常见的树、表格、图表、信息图表等模式。本章讨论了数据表现方式的认知知识，以及如何使用它们来进行知识和含义的交流。
- 第7章，**从用户获得输入**，处理表单和控件。除了模式之外，本章还提供了一个表格，用来把各种数据类型映射到可以表现它们的不同控件上。

- 第8章，**Builder**和编辑器。讨论常用在所见即所得的图形编辑器和文本编辑器中的模式和技术。

最后，第9章讲述设计过程的最后一步，当然它也可以应用到你设计的任何东西上。

- 第9章，修饰外观。讲述美观和最后的工序，完成设计。它使用图形设计原则和模式来讲述在界面的行为已经稳定的基础上，怎样及为什么需要打磨它的外观和感觉（Look and Feel）。

在为本书选择例子的时候，我考虑了很多因素。当然，其中最重要的因素是该例子是不是很好地展现了这个模式或概念，不过其他的因素包括总体的设计适合度、是否可以打印、平台可变性——是否可以适用于桌面应用、网站、设备等——及这些应用有多知名、读者是否能访问到。因此，考虑到这些方面，很多例子来自微软和苹果公司的软件、一些特定的网站及容易找到的消费类软件和设备。这并不是说它们通常代表了良好的设计。不是那样的，我也不是要故意忽略很多设计师的优秀设计，或者不那么知名的应用。如果你知道有一些例子符合上面这些准则，请把它们推荐给我。

问题和评论

COMMENTS AND QUESTIONS

我们已尽力核验本书所提供的信息，尽管如此，仍不能保证本书完全没有瑕疵，而网络世界的变化之快，也不可能保证本书永不过时。如果读者发现本书内容上的错误，不管是繁字、错字、语意不清，甚至是技术错误，我们都竭诚虚心接受读者指教。如果您有任何问题，请按照以下的联系方式与我们联系。

与本书有关的在线信息如下所列。

<http://www.oreilly.com/catalog/designinterfaces/> (原书)

<http://www.oreilly.com/book.php?bn=978-7-121-04797-8> (中文版)

奥莱理软件（北京）有限公司

北京市 海淀区 知春路49号 希格玛公寓B座809室

邮政编码：100080

网页：<http://www.oreilly.com.cn>

E-mail：info@mail.oreilly.com.cn

博文视点资讯有限公司（武汉分部）

湖北省 武汉市 洪山区 邮科院路特1号 湖北信息产业科技大厦1402室

邮政编码：430074

电话：(027)87690813 传真：(027)87690813转817

读者服务网页：<http://bv.csdn.net>

E-mail：

reader@broadview.com.cn (读者信箱)

bvtougao@gmail.com (投稿信箱)

致谢

ACKNOWLEDGMENTS

首先，我要向本书的技术审校人员致以衷心的感谢：Eric Freeman、Peter Morville、William Wake、Robert Reimann、Jeff Johnson、Martijn van Welie，还有Ron Jeffries，你们的建议无疑让本书增色不少。

要感谢的其他读者、审校人员和提供宝贵建议的朋友，包括Midtmoen Fease、Jan Stetson、Helen Rennie、Rhon Porter、Geoff Dutton、Steve Eddins、Lynn Cherny、Tom Lane、Joe Conti、Will Schroeder、Janice Kutz、Tim Wright、Ben Bederson、Robert Nero和Michael Del Gaudio，你们付出了不少时间和精力来协助完成本书，谢谢你们！非常感谢我在MathWorks的其他同事，特别是Chris Wood，我在忙着写这本书的时候你一直都很有耐心。

感谢最开始的审校人员，包括Ralph Johnson和伊利诺斯大学香槟校区的软件架构小组：Tankut Baris Aktemur、John Bordan、John Brant、Nicholas Bray、Danny Dig、Christos Evaggelou、Alejandra Garrido、Brian Foote、Munawar Hafiz、Thuc Si Mau Ho、Pablo Montesinos、Jeff Overbey、Weerasak Witthawaskul、Spiros Xanthos、Joseph Yoder（我肯定忘了一些人，你们在记录本上的名字太模糊啦）。

感谢Doug Hull、Tom Lane、Gerard Torenvliet、Alex Conn、Amy Groden-Morrison、Susan Fowler和Robert Nero，你们为很多关键的示例提供了补充材料和链接，谢谢你们的屏幕截图和建议。我希望本书能为你们带来荣誉。

感谢O'Reilly的Nat Torkington，是他发现了最开始的网站并且跟我联系，还有Mike Hendrickson，在写书的开始给了我很多指导，并提出了一些很好的想法，还有Andrew Odewahn和Mary O'Brien，我的编辑，你们在我写本书的草稿时给了我很多帮助。

谢谢访问我的UI模式网站的成千上万不知名的访问者。你们是本书存在的理由，没有你们的访问日志记录，我永远不会知道这个网站得到了多少访问。谢谢那些花时间写信和告诉我你们喜欢它的访问者，我希望本书达到了你们的期望！

非常非常感谢我的家人和朋友，特别是Rich，你一直在鼓励我把网站转换成一本书，并且在本书的写作过程中全心全意地支持我。

最后，我想感谢本地的冠军球队：2003年和2004年新英格兰爱国者橄榄球队，以及2004年波士顿红袜棒球队，是你们让我们相信，任何事情（任何事情！）只要努力就能成功。

甚至是写一本书。

目录

CONTENTS

前言	II	
01 用户做些什么	2	
达到某种目的的手段	3	
用户研究基础知识	5	
用户的学习动机	7	
模式	10	
1 安全探索	11	
2 即时满足	11	
3 满意即可	11	
4 中途变卦	12	
5 延后选择	13	
6 递增构建	14	
7 习惯	14	
8 空间记忆	15	
9 前瞻记忆	16	
10 简化重复工作	17	
11 只支持键盘	17	
12 旁人建议	18	
02 组织内容：信息架构和应用结构	20	
信息架构基础知识：切分内容	22	
物理结构	28	
模式	30	
13 双面板选择器	31	
把两个相邻的面板放在界面上，在第一个面板上显示一组对象，用户可以从中任意选择，然后在第二个面板上显示选中对象的内容。		
14 画布加调色板工具条	34	
把一个带图标的调色板工具条放在空白的画布旁边，用户单击调色板工具条上的按钮，在画布上创建对象。		
03 到处走走：导航、路标和找路	54	
不要迷路	55	
导航的代价	56	
模式	63	
21 清楚的入口点	64	
只在界面上显示几个人口，让它们面向任务，并具有自描述性。		
22 全局导航	66	
在每个页面上用一个小栏目来显示一组一致的链接或按钮，让用户可以通过它们来访问网站或应用的核心栏目。		

中心和辐条	68	33 中央舞台	103
把应用的各个栏目分割成几个迷你的小应用，每一个应用都有一个入口（从主页面）。并且可以回到主页面。		把最重要的UI部分放到页面或窗口最大的子栏目上，把一些辅助工具和内容放置在它周围的小面板上。	
金字塔	71	34 带标题的栏目	107
使用后退/下一个（Back/Next）的链接来关联一系列页面。把这种序列化展现方式和一个主页面结合起来，由该主页面链接到（并链接回）这个序列中所有的页面。		通过给每个栏目一个醒目的标题来区分不同栏目内的内容，然后把它们都排列在页面上。	
模态面板	74	35 卡片堆	109
只显示一个页面，在用户解决好当前的问题之前没有其他任何导航选择。		把不同栏目的内容组织成几个单独的面板或“卡片（Cards）”，并把它们垒成一堆，一次只显示一个栏目，用户可以使用Tab页或其他设备来访问它们。	
序列地图	76	36 可关闭的面板	111
在一系列页面的每一页上，显示一幅地图，把所有的页面按顺序排序在该地图上，包括一个“你在这里”的位置指示器。		把不同栏目的内容组织成几个面板，让用户每次可以打开和关闭单独的面板而不影响其他面板的显示。	
面包屑层级结构	78	37 可移动的面板	114
在层级结构的每个页面，显示所有父级页面的链接，向上追溯到主页面为止。		把页面上的工具或栏目组织到几个不同的面板上，让用户可以移动它们，形成自定义的布局。	
注释滚动条	80	38 右对齐/左对齐	116
让滚动条在滚动的同时，还可以作为一种内容的映射机制，或者说，作为一个“你在这里”的位置指示器。		在设计一个两竖栏的表单或表格时，让左边竖栏的标签右对齐，而右边竖栏的元素左对齐。	
颜色编码的栏目	82	39 对角平衡	118
使用颜色来标记一个应用或网站中某个网页所属的栏目。		用一种不对称的方式布置页面，但是，通过把视觉重量放在左上角和右下角来使页面保持平衡。	
动画转换	84	40 属性表	120
把一个突然出现或位置移动的转换用动画来显示，让它变得更自然。		使用两栏或表单格式的布局来告诉用户，一个对象的属性可以在这个页面进行编辑。	
逃生舱	86	41 响应式展开	123
在每个限制了导航选择的页面上，放置一个按钮或链接，让用户能明白无误地离开这个页面，回到熟悉的地方。		从一个最小限度的用户界面开始，通过在每个步骤显示更多的界面，引导用户完成一系列步骤。	
04 组织页面：页面元素的布局	88	42 响应式允许	125
页面布局基础知识	89		
模式	99		
32 视觉框架	100		
使用相同的基本布局、颜色、格式方案来设计页面，但是会让设计足够灵活来处理不同的页面内容。			

43	流式布局	128
	当用户调整窗口大小的时候，相应地调整页面元素的大小和位置，让页面一直保持填满的状态。	
05	完成任务：动作和命令	130
	扩展边界	133
	模式	136
①	按钮分组	137
	把相关的动作组织成一组按钮，彼此水平对齐或垂直对齐。如果超过三个或四个动作，那么可以多分几组。	
②	动作面板	140
	不使用菜单，而是把大量相关的动作放在一个UI面板上。该面板可以有丰富的组织方式，并且通常都是可见的。	
③	突出的“完成”按钮	144
	把完成一项任务的按钮放在视觉流的末尾，加大它的尺寸并为它提供一个合适的标签。	
④	智能菜单项	146
	动态改变菜单的标题，以便在调用之前明确地知道它的功能。	
⑤	预览	147
	在用户执行某个动作之前，为其显示执行结果的预览或小结。	
⑥	进度提示	149
	在那些需要很长时间才能完成的操作中为用户显示该操作当前的进度。	
⑦	可取消性	151
	提供某种方式快速取消一个耗费时间的操作，而没有什么负面影响。	
⑧	多级撤销	153
	提供一种方式可以很容易撤销用户执行的一系列操作。	
⑨	命令历史	156
	当用户执行某些操作时，保存一份可见的记录，记录做了什么操作、作用在什么对象上，以及执行操作的时间。	
宏	158	
	宏是一个操作，但是在该操作里组合了其他一些小的操作，用户可以通过把一系列操作放在一起来创建一个宏命令。	
06	显示复杂数据：树、表格及其他信息图形	160
	信息图形基础	161
	模式	173
54	总览加细节	174
	在放大的详细视图旁边放置一个总览图。当用户在总览图上圈出一个视野的时候，详细视图里将显示该视野范围里的内容。	
55	数据提示	176
	当鼠标滑过图形上一个兴趣点时，把该点的数据显示在一个工具提示或其他某种浮动窗口上。	
56	动态查询	178
	提供一些方式来快速过滤数据，并让这种过滤充满交互性。采用容易使用的标准控件，例如滑块和复选框，来定义数据需要显示的范围。只要用户调整了这些控件的值，显示结果也随之调整。	
57	数据刷	181
	让用户在一个视图里选择数据，然后在另一个视图里同时显示这些选中的数据。	
58	局部缩放	184
	在一个密集的页面上显示所有数据，其中有很多小小的数据项目。当鼠标移动时页面会变形。鼠标位置的数据项变大，变得可以阅读。	
59	斑马行	187
	在表格行上使用两种样式类似但颜色不同的背景，来间隔显示数据。	
60	可排序表格	189
	在一个表格里显示数据，并让用户根据表格列的内容对表格进行排序。	
61	跳转到对象	191
	当用户输入某个对象的名称时，直接跳转到该对象并选中它。	
62	新对象行	193
	在表格的最后一行创建新的对象。	
63	级联列表	195
	通过在每个层级上显示一个可以选择的列表来表示层级结构。对任何对象的选择都会将其子对象显示在下一个列表中。	

64	树状表格	197	73	自动完成	227
	把层级结构的数据放到各个列里，就像表格那样，并在第一列使用一种缩进的大纲形式来表现树结构。			当用户在文本输入框内部输入的时候，猜测可能的答案并在合适的时候自动把答案填写到输入框里。	
65	多Y值图表	198	74	下拉选择器	230
	把多个图表的线段堆积起来，一个接一个地从上到下摆放在相同的面板里，让它们共享同一个X轴。			通过使用下拉列表框或弹出面板，把菜单条的概念扩展成一个更加复杂的界面。	
66	大小小对象	200	75	图示选项	233
	用两三个维度创建很多小的数据图片，并按一到两个附加数据维度平铺到整个页面。要么平铺成连环画式的长条，要么平铺成二维矩阵。			用图片而不是文字（或在文字上附加文字）来显示可选项。	
67	树状地图	203	76	列表建造器	235
	用不同大小的矩形来表示多维数据和/或层级结构的数据。可以把这些矩形级联起来显示其层次结构，并用不同的颜色或给它们加上标签来显示更多的变量。			在同一个页面上显示源列表和目的列表，让用户在它们之间移动项目。	
07	从用户处调输入：表单和控件	206	77	良好的默认值	237
				无论在什么地方，只要合适，就预先为用户填写他们想要的输入值到各个字段中。	
	表单设计基础	207	78	错误显示在同一页	239
	控件选择	209		把表单的错误信息和表单放在一起，直接放在页面上，在页面顶端标记错误信息。如果可能的话，在产生错误的控件旁边进行提示。	
	模式	218	08	Builder和编辑器	242
68	容错格式	219			
	允许用户以多种格式和语法输入文本，并让应用系统具有智能解释文本的能力。		编辑器设计基础	244	
69	结构化的格式	220		模式	248
	和前面模式只使用一个文本输入框不同，这里使用一组文本输入框来反映所请求数据的格式。		79	就地编辑	249
70	填空	222		使用小的动态文本编辑器，让用户可以“就地”修改文本，把这个编辑器直接放在原始文本的上面，而不是使用一个单独的面板或对话框。	
	把一个或多个数据字段设置成散列的句子或短语形式，把那些输入变成“空白区域”，以便用户填写。		80	智能选择	251
71	输入提示	224		使软件足够聪明，自动选择一组相关对象，而不是让用户自己去选择。	
	在一个空白的文本输入框旁边，用一句话或一个例子来解释需要输入什么样的数据。		81	组合选择	253
72	输入提醒	225		在不同的屏幕区域使用不同的操作——或鼠标单击，例如组合对象的边缘相对它的内部，来确定应该选择这个组合对象本身还是允许它所包含的那些对象也被选中。	
	用提示信息预先填写文本输入框或下拉列表框，来告诉用户该怎么做/输入。				

82	一次性模态	255	90	角落处理	297
	当一个模态打开的时候，执行一次操作。然后自动切换回默认模态或前一个模态。			不是使用普通的直角，而是使用斜纹、曲线或镂空来装饰界面上的方框角落。让这些角落处理和整个界面保持一致。	
83	弹性加载模态	257	91	边界回应字体	300
	让用户通过按住某个键或鼠标按钮来进入某种模态。当用户释放它时，离开这个模态并回到之前的模态。			在描画边界和其他线条的时候，使用设计中某种主要字体所使用的颜色、粗细和曲线。	
84	强制调整大小	259	92	发丝	303
	为调整大小模态提供不同的行为，例如保持原始比例，以便在特别的环境下使用。			在边界、水平标尺和Texture上使用的一个像素宽的线条。	
85	磁性吸附	261	93	粗细字体对比	306
	让对象具有“磁性”，可以吸住靠近它们的其他对象。当用户推动一个对象到这种“磁性”目标对象附近时，它就会贴在目标对象上。			使用两种字体互相对照——一种细一些，颜色浅一些；另一种粗一点，颜色深一点——来区别不同级别的信息，并丰富视觉效果。	
86	对齐指示线	263	94	皮肤	308
	提供竖直方向和水平方向的参考线，帮助用户对齐各个对象。			为你的应用提供开放的外观和感觉架构，让用户可以设计他们自己的图形和风格。	
87	粘贴变种	266			
	在标准的粘贴操作之外，另外提供专门的粘贴功能。				
09	修饰外观：视觉风格和美感	268			
内容相同，风格不同	270				
视觉设计基础	279				
这对桌面应用来说意味着什么	287				
模式	290				
88	深色背景	291			
	用图片或渐变颜色作为页面的背景，这样能可视化地拉开和前景元素的距离。				
89	少一点色彩，多一些价值	294			
	选择一种、两种，最多三种主要颜色样式应用在页面上。通过选择这很少几种颜色的混合值（不同亮度）来建立一个颜色板。				

TP311.1/65

2008

O'REILLY®

Designing Interfaces中文版

Designing Interfaces

Jenifer Tidwell 著
De Dream' 译

电子工业出版社

Publishing House of Electronics Industry
北京•BEIJING

01

用户做些什么
WHAT USERS DO



本书的内容几乎全部和应用、Web应用、交互式设备外观和行为有关。但是第1章例外。本章没有屏幕截图，没有界面布局，没有导航，没有图表，没有任何可视化元素，完全没有。

为什么呢？毕竟那些才是你最开始拿起这本书的理由。

因为好的界面设计并非始于图片，而是始于对人的理解：人们喜欢什么，为什么会使用某种特定的软件，他们可能会怎样与之交互。对使用的人了解得越多，你就越能对他们移情，也越能进行有效的设计。毕竟，软件对于使用它们的人来说，只是达到目的的一种手段，你越能帮助他们达到那些目的，他们就越高兴。

每当有人使用一个应用或者任何数字产品的时候，他们都在和机器进行对话。可能是文本的对话，通过命令行或电话语音菜单进行；也可能是一种隐含的对话，就像画家和她的画笔及画布那样——工匠和他所建造的作品之间那种给予和接受。和社会性的软件之间则可能是一场通过代理进行的对话。不管怎样，用户界面作为对话的中介，它帮助用户达到任何他/她思想中的目标。

那么，作为用户界面设计师，你得为那样的对话编写脚本，或者至少定义它的术语。而如果打算为某场对话提供脚本，你必须同样对人有着良好的理解。用户的动机和意图是什么？他们希望使用的词汇、图标和应用姿态是什么？应用怎样才能为用户设置适当的期望？用户和机器怎样才能在最后结束一次对彼此都有意义的对话？

这才是界面设计领域最重要的一块：“了解用户，因为他们不是你！”

所以本章我们将讨论人的因素。在介绍中将会简短地提到一些基本概念，然后接下来讨论模式。这些模式和书中的其他模式不同。它们描述的是人的行为——而不是系统的行为——也就是你设计的软件需要支持的那些行为。支持这些人类行为的软件将帮助用户达到他们的目标。

达到某种目的的手段 A MEANS TO AN END

每个人在使用工具、软件或别的什么的时候，总有一个理由，例如：

- 寻找信息或对象
- 学习知识
- 进行交易
- 控制或监视
- 建造某种东西
- 和其他人交谈
- 娱乐

广为人知的习惯用法、用户行为和设计模式都可以支持这些抽象的目标。交互设计师们已经知道，比如，如何帮助人们在海量的网络书籍中寻找某些特定的信息。他们知道怎样安排任务，让它们更容易完成。他们也在学习如何支持文档、演示和代码的构建。

设计界面的第一步是要找出用户真正要达到的目的。例如，填写一张表单本身几乎从来不是什么目的——人们填写表单是因为他们要在线购物、更新驾驶执照、安装网络打印机等。¹他们是在进行某种事务。

提出正确的问题有助于把用户的目标和设计过程结合起来。用户和客户通常会与你谈论他们想要的功能和解决方案，而不是真正的需要和存在的问题。当一名用户或客户告诉你他想要某个特定的功能时，记得问他为什么——来确定他最直接的目标。接着，针对这个问题的答案，再问为什么。然后再问，一直问到远远超出这个现有问题的边界为止。²

如果你已经有了明确的需求，为什么还要追问这些问题呢？因为如果你喜欢进行设计，很容易陷入某个有意思的界面设计问题上。可能你很擅长用合适的控件来设计表单，使表单要求用户提供合适的信息，且布局良好、美观大方。但是界面设计真正的艺术在于：解决正确的问题。

所以不要太热衷于设计那张表单。如果有某种方法可以很快完成任务，同时用户却完全不必用到那张表单，那就把它整个去掉。这么做可以让用户离他的目标更近，花费更少的时间和精力（可能对你来说也是一样的）。

让我们用“为什么”的方法深入挖掘一些典型的设计情景。

- 为什么一位中层经理要使用邮件客户端？噢，当然——收邮件。那么为什么她需要收发邮件呢？为了同别人交流。当然，其他方法也可以达到交流的目的：通过电话、走廊谈话或者正式的文件。但是很显然邮件满足了一些其他方式所不能满足的需要。那么是哪些需要？为什么这些需要对她来说重要？保护个人隐私？保存谈话内容？社会习俗？还有吗？
- 一位父亲来到一家旅游代理网站，输入他们一家准备夏天去休假的城市，想要找到不同时间内的机票价格。他从结果中学到了很多，但他的目标不是浏览和查看各种选择。为什么？他的目标实际上是完成一桩事务：购买机票。同样，他可以通过许多别的网站来完成，或通过电话向旅游代理订票。这个网站比它们好在哪里呢？更快？更友好？得到更多优惠？

1. 参见Eric Raymond的短文：“无知的奢华，一个关于开源的恐怖故事”，关于他使用Linux打印功能的惨痛经历，地址在<http://www.catb.org/~esr/writings/cups-horror.html>, <http://org/~esr/writings/cups.html>。
2. 这和有名的根本原因分析（root cause analysis）技术的原理一样。当然，根本原因分析是一种查找组织失败原因的工具，而这里，你使用“五个为什么”（多几个少几个都可以）来理解用户的日常行为和功能请求。

- 一位手机用户想要更快地查找他的手机号码本。作为设计师，你可以想出几个聪明的办法来让用户少按几次按钮。但是他为什么需要这么做呢？原来是他在开车的时候要打很多电话，而他又不想让视线离开路面，越少越好——他想在打电话的同时（尽可能地）保证安全。理想的情况下，他最好是完全不必看手机。一个更好的解决方案是语音拨号：他要做的只是说出名字，然后手机自动拨打。
- 有些时候目标分析并不那么直观。一个滑雪网站可能会提供信息（供访问者学习），一个在线商店（完成事务）、一些Flash短片（提供娱乐）。假设一个人访问网站想要购买滑雪板，但她又被滑雪技巧的文章所吸引——从完成事务的目标切换到了浏览和学习。可能她还会回过头去购买，也可能不会。还有，这个网站提供的娱乐功能是否能同时让20岁的人和35岁的人都满意？如果觉得在这里不自在，35岁的访问者会不会到别的地方去购买滑雪板？还是他并不在乎？

把各种各样的使用者建模成一个没有面目的实体——“用户”——带着一个面向任务的目标，查一些简单的用例，这么做似乎很容易，但不一定能反映用户的实际情况。

要得到好的设计，你需要考虑很多“软”的因素：不经大脑的直接反应、个人偏好、社会环境、信仰、价值观等。这些因素都会影响到对应用/网站的设计。在这些“软”的因素当中，你可能会找到一些关键的特征或设计因素来让你的软件更成功、更引人注目。

因此要保持好奇心，悉心研究。找出你的用户真正喜欢什么，他们的真实想法和真实感受。

用户研究基础知识 THE ABSICS OF USER RESEARCH

要获得这些信息，唯一的方法是根据经验去发现。要开始一份设计，你需要去刻画各种将要使用目标产品的用户（包括前面提到的“软”因素），而最好的方法是走出去，和他们接触。

当然，每一个用户群体都是独一无二的。例如，一种新式手机的目标用户就和某种科学软件的目标用户有很大的区别。甚至就算一个用户同时使用手机和软件，他对两者的期望也会不同——使用科学软件的研究人员可能会容忍用比较粗糙的界面来换取强大功能的做法。同样是这个人，如果发现新手机的用户界面难以使用，他可能会在几天之后把手机低价卖掉。

同样，每一位用户本身也是独一无二的。某个人觉得难用，其他人可能不会这么觉得。其中的秘密在于，去找出对你的用户来说普遍的情况，也就是说，得找到相当多的单个用户来把这个别的个人癖好和公共的行为模式区分开来。

具体而言，你需要学习：

- 他们使用软件的目标；
- 他们实现这些目标时需要完成的具体任务；
- 他们平常使用的语言和词汇；
- 他们使用同等软件的技巧；
- 他们对你设计的目标对象的态度，以及不同的设计对这些态度的影响。

我没法告诉你特定的目标用户是什么样子。你需要去研究他们将怎样和你设计的软件交互，以及该软件有多适合他们的生活环境。虽然这可能会很困难，但还是要去努力描述你的潜在用户，描述他们将怎样使用你的软件，他们使用的理由和原因。你可能会得到一些截然不同的答案，这些答案代表着截然不同的用户群体，这样是可以的。你可能忍不住想摊开双手说：“我不知道哪些人是用户”，或者“每个人都可能是用户”。这对设计完全没有用——没有对他们忠实具体的描述，设计将脱离实际。

不幸的是，这个发现用户的阶段将在设计的早期花费许多时间。这样做代价不菲。但是常常物有所值。因为这样可以得到机会去解决正确的问题——你将在一开始就走在建造正确产品的路上。

不过，所幸的是，现在有很多书籍、课程和方法可以帮助你完成这项工作。虽然本书不打算详细讲述用户研究，但还是提供一些方法和主题供你考虑。

直接观察

访谈和现场用户访问将让你直接置身于用户的世界。你可以向用户询问他们的目标，他们经常需要完成的任务和动作。这样的研究通常是在现场完成的，也就是用户可以实际使用软件的地方（例如在办公室或家里）。可以进行结构化的访谈（也就是预先拟定好问题列表），或者非结构化的访谈，在此种情况下你需要捕捉临时涌现的主题。访谈可以做得很灵活，次数可多可少，时间可长可短，形式可以正式可以非正式，可以通过电话进行或面对面谈话进行。这些都是非常好的机会，让你了解许多自己不知道的东西。记得问为什么，多问几次。

案例研究

案例研究可以帮助你深入了解一些具有代表性的用户和用户群。有时候你可以使用这个方法来探查极端用户，确定软件的边界。这种方法在需要重新设计现有软件时特别有用。你也可以用它进行纵向研究——用数周、数月甚至数年时间考察用户的使用背景。最后，如果为单个用户或网站进行设计，你可能想尽可能多地了解实际使用的上下文。

调查问卷

通过调查问卷能够收集到大量的用户信息。可以从对这些调查问卷的反馈里得到很多统计数据。由于在进行问卷调查时没有直接的人工接触，这种方法可能会丢失大量的额外信息——问卷中没有问到的，就不会有答案，不过能得到目标用户群体某些方面的清晰轮廓。仔细设计问

卷是非常必要的。如果想得到可靠的数字而不是笼统的感觉，毫无疑问你需要把问题写对，选对调查对象，并且正确分析调查结果——这可是一门科学。

人物角色

人物角色不是一种数据收集手段，但他们确实可以帮助你正确处理所收集的数据。这是一种对目标用户群体建模的设计技术。为每个主要的用户群体，都创建一个虚构的人物，来捕捉该用户群体中最重要的方面：他们试图完成哪些任务，他们的最终目标，他们在问题域的经验水平，他们的计算机能力等。这种技术让你始终保持关注重点。随着设计过程的推进，你可以问自己这样的问题：“这个人物角色确实会这样做吗？不然，她会怎么做？”

其实还不止如此。你可能注意到了其中一些方法，例如访谈和问卷调查看起来很像市场活动。的确这样。焦点小组（Focus Group）方法也会很有用（虽然比不上前面那些），而且市场细分的概念和我们这些使用的目标用户定义很相似。在这两种情况下，都要尽你的最大可能去了解用户。

设计师与市场人员的区别在于，作为一名设计师，你在试图了解软件的使用者，而一名市场专家希望尽可能了解软件的购买者。

去了解用户和系统交互之下的真实问题并不容易。用户并不总是有合适的语言或善于自己总结来解释他们实现目标时的真实需要。你需要花费很大的努力去从他们的叙述中找出有用的设计想法——自我讲述的观察常常有些细微的偏差。

这些技术中有一些非常正式，另一些则比较随意。正式的、定量的方法非常有价值，因为它们是基于科学理论的。正确应用的话，这些技术就可以帮助你看到世界的真实情况，而不是你想像的情况。如果你进行用户研究的时候马虎随意，不考虑用户自我选择之类的偏差，很可能得到的数据并不能如实反映你的目标用户——这只会给你的设计造成长远的损失。

不过如果你没有时间进行正式的研究，那么随便去接触几个用户也会比什么都不做要好。和用户交谈很重要。如果能为用户着想，并且能想像这些人将如何使用你设计的软件，那么你就能得到更好的设计。

用户的学习动机

USERS' MOTIVATION TO LEARN

在设计过程开始之前，考虑一下你的整体方案。想想你会怎样设计总体的交互风格——个性。

当你同别人就某个给定的主题展开对话时，你会根据自己对对方的了解来调整要说的内容。你可能会考虑他对这个主题有多关心，他已经知道了多少，他的接纳程度，还有最起码的，他是不是对这

次对话感兴趣。如果你对其中某一项的估计有误，那就会出问题——他可能会觉得没兴趣，没面子，没有耐心，甚至完全抵制。

在设计的时候我们可以从上面的情形中得到借鉴。例如，在界面上使用的主题相关词汇，应该和用户的知识水平相当；如果一些用户不知道某个词汇，那么请为他们提供一个方法来学习不熟悉的术语。如果他们不太懂计算机，别让他们使用复杂的控件或不常用的界面设计习惯用法。如果他们不太感兴趣，请尊重这一点，不要让他们付出很多努力来获取一点点回报。

其中一些因素可能会渗透在整个界面设计当中。例如，用户是在期望一个短的、焦点集中的过程来获得某个特定的东西，还是在希望得到一个范围更自由的探索式对话？换句话说，界面的开放性如何？开放性太弱，用户会觉得受到约束，不满意；太开放，他们又会觉得不知所措，不知道下一步该做什么，没有准备好这种程度的交互。

因此，你需要选择给用户多少自由。一种极端的情况可能是软件安装向导：除了下一步、上一步、取消之外，用户基本上没有机会进行别的操作。它高度集中、具体，但是非常有效——而且令人满意，只要它能有效且高速完成。另一种极端的情况可能是像Excel那样的软件，一个“open floor plan”形式的界面展现了大量功能。而且在任何时候，用户接下来可以做872件不同的事。这样的软件也很好，因为熟练的用户可以用这样的界面完成很多工作。同样，它也是令人满意的，虽然这次是基于完全不同的理由。

这里甚至有一个更根本的问题：用户愿意花多大的努力来学习使用你的界面？

对这个问题的估计很容易过于乐观。也许他们每天在工作中使用它——在这种情况下他们自然乐意去学习，但这种情况并不多。也许他们某些时候会用，因此只会学到能用就行。也许他们只会看到一次，一眼就过。坦白一点：你能期望大部分的用户成为熟练用户吗？还是他们中的大多数人都只会停留在新手阶段？

为熟练用户设计的软件包括：

- Photoshop
- Dreamweaver
- Emacs编辑器
- 代码开发环境
- Web服务器的系统管理工具

相反，下面这些是为偶尔使用而设计的：

- 旅游中心或博物馆的信息亭
- 设置桌面背景的Windows或Mac控件
- 在线商店的购买页面

- 安装向导
- 自动提款机

上述两个列表之间的区别很大：对于用户知识的假设遍布在这些界面当中，体现在它们屏幕空间的使用、是否采用标签、使用怎样的控件技巧、是否提供帮助等方面。

第一组的应用有很多复杂的功能，但它们不会让用户一步一步完成任务。它们假设用户已经知道该怎么做，并为有效的操作而不是可学习性进行优化；它们可能是以文档为中心的，或是列表驱动的（其中一些是命令行应用）。它们通常有整套的书籍和课程，其学习曲线会很陡峭。

而第二组的应用刚好相反：功能有限，而且一路提供操作帮助。它们提供简单的界面，假设不需要了解文档或列表中心的应用基础知识（例如菜单条、多选框等），它们通常会提供向导的形式，不需要用户特地集中注意力。这里的关键是用户不需要有学习这些界面的动机——因为通常没必要去学习。

看过这些极端的例子之后，现在来看看处于中间状态的应用：

- Microsoft Office
- 邮件客户端
- 网络浏览器
- 手机应用
- 掌上电脑操作系统

事实的真相是，大部分应用都属于这种中间情况，需要为两类用户提供服务——既要帮助新用户学习（满足他们的即时需要），又要让经常使用它们的中间用户快速完成工作。这些应用的设计师们可能知道，不能指望用户通过3天的课程去学习邮件客户端的使用。而界面同样要支持重复使用。人们很快就学会了基本知识，达到自己满意的熟练程度，并且在他们不需要完成更具体的目标时不用进一步学习。

Alan Cooper先生提出了“独占姿态”（sovereign posture）和“暂时姿态”（transient posture）这两个术语来讨论这些方法。独占姿态的应用把用户当做工作搭档：用户在其中花费很多时间，需要全神贯注，需要进行学习，并把这些应用放大到全屏来使用。暂时姿态的软件在需要的时候出现，用完后就不用管了。这两种姿态大致对应于我前面提出的两组极端情况，但也不完全一样。请参看*About Face 2.0: The Essentials of Interaction Design*来了解更多关于软件姿态的知识。

也许有一天你会发现自己处在这两个极端之间。自然你希望人们从一开始就可以使用你的应用，但同时也想尽可能支持经常使用的用户或专家用户，那么就应该去努力找到两者之间合适的平衡。第2章中提到的组织型模式，例如多级帮助（Multi-Level Help）、有趣的分支（Intriguing Branches）、需要时显示（Extras on Demand）等可以帮助你支持两方面的用户。

模式

THE PATTERNS

尽管每个人都一样，但人们的行为还是可以预测的。设计师们已经进行了长年的网站访问研究和用户观察，认知科学家和其他的研究人员也花了数百小时来观察人们如何行事，以及他们自己是怎样评估自己行为的。

因此，当你观察人们使用软件或执行其他任何你希望在新软件中支持的动作时，用户的行为是可以期望的。在用户观察中常常可以见到下面列出的行为模式。如果你也看到了它们，那很好——尤其是当你在寻找它们的时候。

对于模式爱好者来说，这里还有一个忠告：这些模式和书里的其他模式不同，这些模式描述的是人的行为，不是界面元素；它们的描述方式也和其他模式不同，其他模式是以标准的结构化方式描述的，而这些模式只是一些文字叙述。

再强调一次，和那些不支持这些模式的界面相比，一个可以支持这些模式的界面能更有效地帮助用户达到他们的目标。而且，这些模式并不只是关于界面的，有时候这整一套东西——界面、底层架构、功能选择、文档——需要针对这些行为进行考虑。不过作为一名界面设计师或交互设计师，在团队中你是最需要关注这些的人，也可能是最适合为用户争取利益的人。

- | | |
|---|--|
| <p>1 安全探索
Safe Exploration</p> <p>2 即时满足
Instant Gratification</p> <p>3 满意即可
Satisficing</p> <p>4 中途变卦
Changes in Midstream</p> <p>5 延后选择
Deferred Choices</p> <p>6 递增构建
Incremental Construction</p> | <p>7 习惯
Habituation</p> <p>8 空间记忆
Spatial Memory</p> <p>9 前瞻记忆
Prospective Memory</p> <p>10 简化重复工作
Streamlined Repetition</p> <p>11 只支持键盘
Keyboard Only</p> <p>12 旁人建议
Other People's Advice</p> |
|---|--|

1

安全探索 Safe Exploration

“让我进行探索，不会迷路，也不会陷入麻烦。”

如果某个人觉得她可以在一个界面上探索而不会产生可怕的后果，和那些不能这么做的人相比，她可能会学到更多的知识——也会对这个界面更加肯定。好的软件允许人们去尝试某些不熟悉的操作，退回，试试别的，这时候没什么压力。

这种“可怕的后果”并不需要特别可怕，只要一点点小麻烦，就足以让用户打消尝试的念头了。单击关闭弹出的窗口，重新输入被误清除的数据，当某个网站突然播放大声的音乐时猛地关掉笔记本上的声音开关——这些都会让人不愉快。在设计各种软件界面时，可以给用户留下试验性的通道来让他们探索和尝试，同时别让用户付出任何代价。

下面是一些例子：

- 一名摄影师在一个图片处理软件上尝试某些滤镜。他觉得自己不喜欢这些滤镜处理的结果，就单击了几次“撤销”，回到自己处理之前的状态。然后他又尝试另一个滤镜，接着再试一个——每次都能从操作中回到原来的状态（这个模式的名字是多级撤销（**Multi-Level Undo**），在第5章中描述了它的用法）。
- 一名来到某个公司网站的新访问者，因为相信后退按钮可以把她带回首页，她单击了几个链接来看后面是什么内容。没有多余的窗口或弹出窗口，后退按钮可以一如既往地退回去。可以想像，如果某个应用在后退按钮上提供一种不同的功能——或者，如果某个应用提供了一个看起来像后退，但其实不是后退的按钮——混淆就产生了。用户在导航的时候分不清方向，然后她可能会放弃这个应用。

一个手机用户想要尝试一些看起来还不错的网络功能。例如实时得到世界级比赛的比分。但他有点不想去试用，因为上次他使用某个网络服务时，才试用了几分钟就被收了一大笔钱。

2

即时满足 Instant Gratification

“我想现在就完成某件事，是现在，不是以后。”

人们喜欢立刻从他们的行为看到结果——这是人类的天性。如果一个人开始使用一个软件，然后在最初的几秒中内就得到了成功的体验，他会觉得很开心。他很可能会继续用它，哪怕将来它的难度会增加。和要花上一些时间来弄清楚状况相比，前一种情况会让他对这个软件更有信心，对他自己也更有信心。

支持即时满足的需要会产生多个设计分支。例如，如果你能预测到一名新用户会做的第一件事，那么你需要对UI进行设计，让这件事非常容易完成。如果用户的目标是创建某个东西，那么先显示一个新的画布，然后提供调色板。如果用户的目標是要完成某个任务，那么为他提供一个明显的起点。

同时，即时满足也意味着你不应该把介绍性的功能隐藏在任何需要阅读或需要等待的功能之后。例如注册、长串的指令、载入很慢的屏幕或广告。这些都是妨碍用户立即完成第一个任务的绊脚石。

3

满意即可 Satisficing

“这样就够了，我不想花更多的时间来做得更好。”

当人们看到一个新的界面时，他们不会事无巨细地阅读每个细节，然后作决定：“嗯，我想这个按钮是我要

达到目标的最佳选择。“实际情况完全相反，用户会快速扫描这个界面，挑任何一个他第一眼看到，也许会达到目标的元素，然后尝试——哪怕那可能是一个错误的选择。

术语“满意即可”（Satisficing）是“满意”（Satisfying）加上“够用”（Sufficing）的结合体，是在1957年由社会科学家赫尔伯特·西蒙（Herbert Simon）提出的。他用这个词来形容人们在各种经济和社会状况下的行为。当了解到所有可能的选择需要付出时间和努力的时候，人们愿意接受“够用”而不是“最佳”的选择。

一旦你了解到分解复杂界面需要付出的脑力劳动时，满意即可实际上是非常理性的行为。Steve Krug在他的书*Don't Make Me Think*（“New Riders”出版社）里指出，人们不愿意付出哪怕一点点多余的思考——而且事实上这样是完全可行的。但是如果界面提供了一到两个非常明显、用户一眼就能看到的选项，他会去尝试。如果选对了，那很好；如果选错了，也没关系，因为要回到刚才的地方并试别的会很容易（假设该界面支持安全探索模式）。

对于设计师来说，这就意味着：

- 要让标签简短、文字通俗易懂，而且可以快速阅读（标签包括菜单项、按钮、链接和任何文字类的元素）。人们会快速扫描并进行猜测，要让用户第一次就能猜对。如果猜错了几次，他可能会倍感挫折，这对于双方来说都不是好的开始。
- 使用界面的布局来表达含义。第4章“布局”解释了具体怎样进行布局设计。用户在视觉上分解颜色和界面元素外形，与必须阅读的标签相比，这些线索将更有效地分解页面。
- 让界面容易导航，尤其是勿促地作了一个错误的选择后还能够回来。可以为用户提供逃生船（Escape Hatches，第3章）。在常见的网站上，很容易使用后退按钮，所以设计容易使用的前进/

后退导航对于Web应用来说非常重要；当然，对于安装的程序和设备来说，这一点也同样重要。

- 要记住，复杂的界面对新用户来说，意味着繁重的认知负担。视觉复杂性常常诱使非专家用户采用满意即可的策略——他们只寻找第一个可行的选择。

满意即可就是为什么用户在使用某个软件一段时间以后养成各种奇怪习惯的原因。很久以前，某个用户可能已经学会了一种完成任务的方式，然后即使该软件的后续版本又提供了另一种更好的方式（或者它原来一直就在），这名用户看不到再次学习的必要——毕竟，学习需要付出劳动——他就一直使用那种效率不高的老方式。这并不是不理性的选择。打破旧的习惯，学习新的方法需要时间和精力，一点点改进可能不值得用户这么做。

4

中途变卦

Changes in Midstream

“我在中途改变主意了。”

在一些偶然情况下，人们会在正在完成某事的过程中改变主意。一个人可能走进房间，原本想找到她遗落在房间的钥匙，但是她进去以后，发现一张报纸，然后就看起来。又或者她原本访问Amazon是想要读一些产品评论，后来却买了一本书。可能她只是注意力转移了，也可能是经过考虑才这么做的。不管是哪种情况，总之用户就在使用界面的途中改变了她的目标。

这种情况对设计者来说，意味着你应该为人们中途变卦提供机会。让他们可以进行其他选择。如果没有充分的理由，别把用户锁死在一个没有全局导航，不能到达其他页面或其他功能的环境里。当然，那种需要限制选择的情况确实存在，参考第2章的向导（Wizard）模式、第3章的中心和辐条（Hub and Spoke）模式及模态面板（Modal Panel）模式。

同样，你也可以设计出容易“开始，在中间停止，以后再回来接着在原来的位置继续进行”的过程——这是一个叫做重新进入（Reentrance）的特性。例如，一名律师可能开始在PDA上输入信息到一个表单中，这时一名客户进来了，律师不得不暂时关闭PDA，准备一会再回来完成这个表单的输入。PDA必须保存刚才已经输入的信息。

要支持重新进入，可以让对话框记住之前已经输入的数值（参考第7章的Good Defaults即良好的默认值模式），这些对话框也不一定要是模态的。如果不是模态对话框，用户可以把它们拖拽到一边备用。建造器类型的应用——文本编辑器、代码开发环境、画图程序等可以让用户一次开始多个项目，然后允许用户在一个项目上工作的时候把其他的放在一边。

surveymonkey.com有时在每一页在线调查表单中提供一个按钮：“以后完成”，该按钮将记录到这个时候为止已经完成的选择，关闭当前页面，让用户将来再回来完成该调查。

5

延后选择 Deferred Choices

“我现在不想回答这些，快点让我完成就行了！”

这个模式来自人们即时满足的愿望。如果你在一个用户准备完成某件事的时候问他几个没有必要的问题，他经常会宁愿先跳过这些问题，以后再回过头来回答。

例如，一些基于Web的公告板有冗长而复杂的用户注册过程。屏幕显示名称、E-mail地址、隐私设置、头像、个人描述等。这个清单还可以延续很长。“我只想发个贴而已”。用户痛苦而又委屈。为什么不跳过这些问题，只回答最少的必要问题，然后让用户将来继续

填写（如果还有将来的话）其他部分？否则他可能得花上半个小时，来回答长篇累牍的问题，找到合适的头像。

另一个例子是在DreamWeaver或其他网站编辑器中创建一个新的项目。确实有一些地方需要一开始就决定，例如项目名称，但你可以很容易把其他选择放到以后去做——完成之后将放到服务器的什么位置？我现在还不知道呢！

有时候用户就是不想回答这些问题。还有时候用户没有充分的信息来回答这些问题。要是一个编写音乐的软件包在你开始写作之前就向你询问标题、关键字和一首新歌的节拍，会怎么样呢？（请参见Apple的Garageband，看它是怎么设计的。）

这个模式对于界面设计的含义很容易理解，不过，实现的时候并不是那么容易：

- 不要让用户在一开始就面临太多选择。
- 在必须使用的表单上，清楚地标注必须填写的字段，同时不要让这些必须填写的字段数目太多。让用户可以跳过可选问题，继续进行。
- 有时候你可以把少数几个重要的问题或选项从其他选项中挑出来。显示这份简短的列表，把详细的问题列表隐藏起来，参见第2章中的需要时显示（*Extras on Demand*）模式。
- 尽可能使用良好的默认值模式（第7章），为用户提供几个合理的默认答案来开始接下来的过程。但是要记住，预先填写的答案仍然需要用户的关注，以防他们想要改变这些默认值。这里仍然需要付出一些小小的代价。
- 让用户将来可以回到需要延后填写的字段，并且，要让这些字段出现在显眼的位置。有些对话框会给用户显示一段短的提示，例如“你将来可以单击编辑项目的按钮来改变它”。网站把用户填写

- 了一半的表单或其他数据保存起来，例如购物车里没有付款的商品。
- 如果某个网站需要用户注册来提供一些有用的服务，用户在第一次体验网站服务的时候不愿意注册——受到吸引，过来看看，然后网站就询问他们的个人资料。事实上，Turbo Tax 2005允许用户在创建用户名之前填写整张税务表单。

6 递增构建 Incremental Construction

“让我改改。还是不对，让我再改一下。这下好多了。”

当人们要创建某个东西的时候，他们并不总是一下就完成它。就算是一名专家用户，也不会在开始之后有条不紊地进行创建，得到一个完美的产物，然后结束这次创建。

完全不是这样的。相反，她会从一些小的片断开始，进行工作，退回来观察，进行测试（如果要创建的对象是代码或其他可运行的东西），修正其中的错误，然后开始建立其他部分。或者如果实在不喜欢现有的成果，她可能会重新来过。创建某个东西的过程总是一阵一阵的。有时候有进展，有时候又会退回去，它经常是增量式的，通过一系列细小的变化而不是大的改动来完成。有时候它是自顶向下的，有时候又会自底向上。

建造器类型的界面需要支持这种类型的工作。要让一次完成一小部分工作更容易。应用要能支持快速的变化和保存。在这些应用里，反馈很关键：在用户工作的时候，持续告诉用户整个目标对象的样子和它们的行为。如果是代码、仿真或其他可执行物的构建，让“编译”的过程尽可能快，然后让操作的反馈看起来非常即时——在修改和看到结果之间尽量不要有延时，有也要让它减到最少。

当一些好的工具在支持创造性的活动时，这些创造性的活动可能会引发用户的“心流”（flow）状态。这是一

种全神贯注全心投入的状态，在这个过程中，人们会忽视时间的流逝，忽略其他的干扰，从而让人持续工作好几个小时——这个过程的乐趣本身就是它的回报。艺术家、运动员、程序员都对这种状态深有体会。

可以肯定的是，不好的工具会让使用的人分神。如果用户需要等待半分钟才能看到她修改后的增量结果时，她的注意力就被分散了，心流的状态被打断了。

如果你想对心流状态了解更多，可以阅读Mihaly Csikszentmihalyi的书，他在这个领域有多年的研究。

7 习惯 Habituation

“那个操作在别的地方都可以，为什么在这里不行？”

当一个人重复使用某个界面时，一些常用的操作将演变成人的反射性动作：用Control+S来保存，单击后退按钮离开一个页面，按返回按钮关闭模态对话框，使用图标来显示和隐藏窗口，就像踩汽车刹车。用户不需要对这些动作进行过多的思考，他们已经养成了习惯。

这种倾向显然会帮助人们成为某个工具的专家用户（它也有助于人们建立一种心流的感觉）。因此，正如你能想到的那样，习惯在很大程度上提高了效率。但也可能成为用户的负担。如果某个操作成为了习惯，而用户试图在一个不支持它的环境下使用时——或者更糟糕的是，造成了某种破坏性的后果时，用户就像傻了。他必须马上再思考这个工具一次（我刚才做了什么？怎样才能达到我原来的目标？），然后他可能不得不撤销这个操作引起的修改。

例如，Control+X, Control+S是Emacs文本编辑器的“保存文件”操作顺序。

Control+A把文本输入光标移动到某行的开始。这些对 Emacs用户来说就是操作习惯。当用户在Emacs上输入 Control+A, Control+X, Control+S时, 它将执行一些完全不会带来破坏性后果的操作。

现在来看看如果在Microsoft Word里面进行这些习惯性的顺序操作会怎么样?

1. Control+A: 全部选择

2. Control+X: 剪切刚才选定的内容 (此时就是剪切整篇文档)

3. Control+S: 保存文档 (哎呀, 天哪!)

这就是为什么在各个应用之间一致性这么重要的原因。

当然, 在一个应用之内一致性也同样重要。一些应用是非常糟糕的, 因为它期望某些操作会执行Action X动作, 然而在某个特定的模型下, 它又会执行Action y动作。千万不要那样。毫无疑问, 用户将在这种地方犯错误, 而且, 他们越熟练——也就是说, 他们越习惯——就越有可能犯那些错误。

这也是为什么确认对话框在提醒用户意外情况时起不到什么作用的原因。当模态对话框弹出的时候, 用户很容易单击确定, 或返回按钮来关闭它们 (如果确定按钮是默认按钮)。如果用户在进行修改时总是弹出这种对话框, 例如删除文件, 那么用户会习惯性地快速关闭它们。如果有一天真的遇到问题, 这个对话框就没什么用了, 因为它在用户的认知中已经不能引起注意了。

我至少见过一个应用, 它把确认对话框的按钮在不同时刻随机排列。这样, 确认的时候得花精力来确定该单击哪个按钮。这并不是确认对话框的最佳表现方式——实际上, 最好在大多数情况下完全不要使用确认对话框——不过至少这样的设计创造性地去除了习惯的影响。

空间记忆 Spatial Memory

“我敢发誓, 这个按钮一分钟以前还在这里。它哪里去了?”

当人们使用东西或文档时, 他们经常通过回忆它们在哪里来再次找到它们, 而不是它们的名字。

就拿Windows、Mac或Linux桌面来说吧。很多人把文档、常用软件和一些其他东西放在桌面上。这证明人们喜欢使用空间记忆法在桌面上寻找东西, 而且这个方法也非常有效。例如, 人们想出他们自己的分组方法, 或者回忆起“那份文档在右上角, 就在xxx的上面”(自然, 在现实世界人们也是这样。很多人的桌面处于一种“有序的混乱”状态, 看起来乱糟糟的, 但是它的主人可以很快找到他要的东西。如果有人帮他们清理, 那才是害了他们呢)。

很多软件把它们的对话框按钮——确定、取消等——放在预定的位置上, 部分原因是由于它们的空间记忆依赖性。在复杂的应用系统中, 下面这些元素也可以由记忆它们的相对位置找到: 工具条上的工具、层级结构中的对象等。因此, 你应该小心使用像**Responsive Disclosure** (响应式展开, 第4章)这样的模式, 把某个东西添加到界面上通常不会引起麻烦, 但是把控件重新排列会打乱空间记忆, 从而难以找到。这得看具体情况, 如果你不确定的话可以做一些用户测试。

空间记忆和习惯之间的关系很密切。和习惯一样, 空间记忆也是一个在应用中及在应用间保持一致性会带来好处的理由。人们会期望在类似的地方找到类似的功能表现。

空间记忆解释了为用户提供自定义的区域来存放文档和其他对象为什么能带来好处的原因, 就像前面提到的桌面一样。这样的区域并不总是实用, 尤其是在对象数目巨大的情况下, 它适用于数目不多的情况。当

人们自己排列这些对象时，他们喜欢记住它们的位置（千万不要在他们没有要求的情况下重新排列它们的顺序）。可移动的面板（Movable Panels）模式（第4章）讲述了一种解决办法。

还有，这也是为什么动态改变菜单有时候会带来麻烦的原因。人们已经习惯了在菜单的顶部和底部看到特定的菜单项。“好心地”重新排列或压缩菜单项可能与用户的使用习惯不符，引发错误。

顺便说一句，菜单和列表的顶部和底部都是很特别的区域，从认知的角度来说，与处于菜单中间的项目相比，顶部和底部的项目会获得用户更多的注意力。记忆也更持久。如果它们发生了改变，那么对用户来说就太糟糕了。

前瞻记忆 Prospective Memory

“我把它放在这里，提醒我以后来处理。”

前瞻记忆是一个著名的心理学现象，然而在界面设计中似乎没有得到足够的重视。不过我觉得应该重视它。

当我们计划在将来完成某事的时候，就要用到前瞻记忆了，然后我们会设置一些提示物来提醒自己。例如，如果第二天需要带一本书去上班，你可能会在前一天把它放在大门前的桌子上。如果你需要迟一点回复某封邮件（而不是现在），你可能会把那封邮件留在屏幕上作为一个物理提示。又或者，如果担心错过会议，你可能会用Outlook或Palm设置一个提醒闹钟，在每个会议开始前5分钟响起。

基本上，几乎每个人都会这么做，这是我们在复杂的、高度计划性的多任务生活环境中的应对手段之一：我们使用“现实中”的知识来帮助我们加强有限的记忆。我们需要把事情做好。

有些软件确实支持前瞻记忆。前面提到过的Outlook和Palm OS，就直接有效地实现了它：它们里面有日历（其他很多软件系统也有），还有能够发出声音的闹钟。但是，还可以用前瞻记忆来做点别的什么呢？

- 个人用的便签，就像可视化的“即时贴”
- 留在屏幕上的窗口
- 直接插入文档中的注释（例如“待完成”）
- 浏览器书签，来保存那些以后还会访问的网站
- 存放在桌面而不是文件系统其他位置的文档
- 收件箱里（或打上标记）的邮件，而不是那些存档的邮件

人们使用各种各样的东西来支持被动的前瞻记忆。不过要注意，上面这个列表中没有哪项是有意那么设计的。它们的设计目的是灵活性——以及任由用户自主组织信息的态度。一个好的邮件客户端允许你建立任何名称的文件夹，而且它也不关心你对收件箱里的邮件进行什么操作。文本编辑器不关心你输入的是什么，或者大号黑体红色的文本是什么意思；代码编辑器不关心你在方法开头的地方有一个“待完成”的注释。浏览器不知道你为什么要把一些特定的书签保存起来。

在很多情况下，这种放任不管的灵活性正是你真正想要的。让人们自己用工具去创建他们的提示系统。不要想着设计一个非常智能的系统来处理这些。例如，不要只因为一个窗口可能会闲置一会儿，没有人用它，就假定这个窗口需要关闭。总的来说，不要“好意”清空那些系统认为可能没用了的文件和对象：一些人也许故意把它们留在那里！还有，除非用户要求，请不要自动组织或排序那些文件/对象。

作为一名设计师，你可以用前瞻记忆来主动做点什么吗？假设有人把一个完成了一半的表单临时关掉，你可以把已经填好的数据保存起来，帮助她下次回忆都填到哪里了。与此类似的是，很多应用能够“想起”最近

几次编辑的文档或对象。可以提供类似书签的兴趣对象清单——旧的或新的都可以——然后让这个清单容易阅读，容易编辑。

这里有个大一点的挑战：如果用户开始某项任务，然后在没有完成的情况下把它们放在一边，想想怎样把一些半成品放在旁边而不是让窗口继续打开，来表示这些没有完成的任务。还有一个挑战：用户怎样才能从不同的地方把各种不同的提示（邮件、文档、日历等）集中放到一个地方？来点创意！

10

简化重复工作

Streamlined Repetition

“我得重复多少遍？”

在很多软件中，用户有时候发现他们得一遍一遍地重复某些操作。这些重复的操作越方便越好。如果可以把这些重复操作减少到一次击键或一次单击鼠标——或者，更方便一些，所有的重复操作只需要几次击键或几次鼠标单击。那么，你就可以把他们从烦闷的重复劳动中解放出来了。

在文本编辑器（Word、邮件编辑器等）里的查找和替换对话框就可以很好地应用这种技巧。用户在这些对话框中输入旧的词组和新的词组，然后只需要每次单击一下替换按钮就可以了。而且，一次替换一个地方是在用户需要检查每次操作的时候才需要——如果他们相信确实需要替换所有的地方，那么只需要单击一次“全部替换”就可以了。一个动作，全部搞定。

这里还有个更常见的例子。当你需要用一次单击进行一些任意次序的操作时，Photoshop可以让你把这些“动作”录下来。例如，如果需要对20张图片进行缩放、剪裁、加亮，然后保存，你可以把对第一张图片进行处理的这四个步骤录下来，然后单击这些动作的播放按钮来处理剩下的19张图片。参见第5章中的宏（Macros）

模式来了解更多。

在脚本环境中，这种行为甚至更普通。Unix和它的衍生版本允许你把任何输入写成脚本，提交到Shell。可以重新调用和执行单个命令，也可以是更长一些的，有Control+P和返回的命令。可以选择任何命令集合到命令行中，把它们放到for循环里，然后敲一下回车就运行它们。又或者，你可以把这些命令放在一个Shell脚本中（或者Shell脚本的for循环里头），然后用单个命令来执行。在这些环境里脚本非常强大，而且随着它变得更复杂，它就演变成某种意义上的编程了。

其他的变种包括“复制-粘贴”功能（免得在许多地方重新输入同样的内容）、用户在软件或桌面定义的快捷方式（免得到文件系统目录中去查找这些应用）、浏览器书签（以免再次输入URL），甚至快捷键。

直接观察用户可以帮助你找出需要支持哪些重复性的任务。用户往往不会直接告诉你这些。甚至他们都意识不到自己正在进行的这些重复操作可以通过合适的工具来帮助完成——可能他们重复操作的时间太长了，已经注意不到这些了。通过观察他们的工作，我们可以发现这些他们自己觉察不到的地方。

无论在什么情况下，这个模式是要为用户提供某种简单的完成重复任务的方式，否则那些重复的劳动会浪费很多时间，很无聊，也容易出错。

11

只支持键盘

Keyboard Only

“别让我用鼠标。”

有些人确实在使用鼠标的时候有困难。还有一些人不愿意在鼠标和键盘之间切换，因为这样切换会浪费时间和精力——他们宁愿把双手一直放在键盘上。还有

一些人看不到屏幕，而他们的辅助设备常常只用键盘API和系统交互。

由于这些用户的存在，一些软件就设计成可以完全通过键盘来操作。它们同时往往也支持鼠标，但是没有哪个操作是只支持鼠标的——只使用键盘的用户不会错过任何功能。

下面是几项用来实现只支持键盘模式的标准技术：

- 你可以在菜单条上定义快捷键、加速键和各种操作的记忆方式，例如Control+S是保存的快捷键。可以参见具体平台的风格指南来看看哪些是标准的定义。
- 从清单中选择，特别是多选项的时候，可以使用箭头键和辅助键（例如Shift），不过这还要看你的平台支持什么。
- Tab键经常用来移动键盘的焦点（也就是当前控制键盘输入的地方），从一个控件移到另一个控件，然后用Shift+Tab来后退。这有时候叫做“Tab遍历”。很多用户希望能在表单格式的界面上使用它。
- 大多数标准控件，包括单选按钮和组合框，都可以允许用户从键盘改变他们的输入值，例如使用箭头键、回车及空格键。
- 对话框和网页经常会有一个默认按钮——它代表的动作是“我已经完成了这项任务”。在网页上，默认按钮常常表示提交或完成，而在对话框里常常是确定或取消。当用户在这些页面或对话框上按下回车键时，就是调用上述操作。然后它将把用户转移到下一页或回到前一个窗口。

还有更多的技术，表单、控件面板、标准Web页面都很容易从键盘驱动。图形编辑器，还有其他更加空间化的界面就更难一些，但也不是不可能。参见第8章的弹性加载模式（Spring-Loaded Mode）模式，这个模式解

说了一种在图形编辑器中使用键盘操作的方法。

只使用键盘模式在数据输入应用中特别重要。在这些应用中，数据输入的速度很关键。用户也不能忍受每次从一个字段转移到另一个字段时或从一页移动到另一页时还要把手从键盘移动到鼠标的麻烦（事实上，很多表单甚至不需要用户按Tab键在控件之间转移焦点，它们会自动进行）。

12 旁人建议 Other People's Advice

“其他人是怎么说的？”

人是一种社会性生物。尽管我们自己的想法有时候很强烈，但其他人的想法也常常会影响我们。

看看在线“用户评论”的快速增长：Amazon上的书评、IMDB上的电影评论、Photo.net和Flickr上的图片评论，还有数不清的在线商店让用户为他们的产品提供评论。像Ebay那样的拍卖网站，就把用户的评价转换成了实际出价。Blog（博客）给人们提供了无限的空间来为任何东西写评论，从产品到编程，甚至政治问题。

人们做任何决定的时候都会受到其他人意见的影响，不管那些意见是直接的还是间接的。在网络上查找，完成事务（我需要买这个产品吗？），玩游戏（其他玩家都在这里做了些什么？），甚至新建什么东西的时候也是如此——在有别人帮忙的时候，人们的效率更高。如果不是这样，至少也会觉得更开心一些。

这里有一个更巧妙的例子。程序员们使用MATLAB来完成一些科学和数学方面的任务。每几个月，开发MATLAB的公司会举行一些编程竞赛，只要几天，参加

竞争的选手就会写出最好的代码来解决困难的科学问题。最快、最准确的代码将会获胜。巧妙之处在于，每位选手都可以看到其他人的代码，而且竞赛鼓励抄袭。这里其他人的“意见”是间接的，以共享代码的形式出现，但是它的影响很大。最后，获胜的代码不完全是原创的，但是无疑会比某一个单独的选手能写出的要好（在很大程度上，这是一种开源软件开发的微观世界，由很多强大的社会动力来推动）。

不是所有的应用和软件系统都可以适应这种社会性的组件，它们也并不是都有必要。但是如果它能增强用户体验，就可以考虑一下。而且，和把一个基于Web的公告板变成普通网站相比，这样会让你更有创意。你怎样才能有建设性地劝说用户？你怎样才能把它结合到用户常用的工作过程之中？

如果是一项创作性的任务，也许你可以鼓励人们把他们的作品贴在公共空间展示。如果是为了找到某些事实或目标，也许你可以让人们更容易看到别人在进行类似查找的时候都找到了些什么。

这本书中提到的模式中，第2章里的多级帮助（*Multi-Level Help*）表述了这个观点，一个在线的支持社区是一个应用完整帮助系统的一部分，而且是很有价值的一部分。

02

组织内容：信息架构和应用结构

ORGANIZING THE CONTENT:
INFORMATION ARCHITECTURE AND
APPLICATION STRUCTURE



在这个时候，可能你知道用户想从你的应用中得到什么。也许你知道要使用哪种习惯用法或界面类型，例如图形编辑器、表单、Web类型的超文本，或者媒体播放器——或者某种把其中几个结合起来的想法。如果你真的很能干，那么你已经写了一些典型的情景来描述人们怎样使用抽象的元素来完成他们的目标。你很清楚这个应用将给人们的生活带来什么样的价值。

那么现在怎么办呢？

你可能会开始勾画界面的草图。很多喜欢可视化思考的人在这个阶段就是这么做的。如果你就是这样的人，需要在完成主体设计之前先用草图来演练，那么就这么做好了。

不过如果你并不是天生用图形来思考的人（有时候即使你是），那么先不要画草图。那样可能会把你的思路限定在最开始画出来的几个设计上。在你把应用的总体组织方式设计出来之前，需要再在一段时间内保持灵活性和创造性。

高层组织方式是一个困难的话题。它有助于从多个角度来思考问题，所以这个介绍部分将深入讲述其中的两个方面。我发现这两个方面都很有用。

第一个方面是“切分内容”，鼓励你去努力把应用系统的内容从它的外在表现上完全分离开来。不要去想什么窗口、树状视图、链接。你应该更抽象一些，去想像这些动作和对象怎样才能和你的主题最顺畅地结合起来。以后再决定使用哪些具体窗口和控件。显然，在设计多模态的应用时，这种分离是有用的，例如，若在Web和Palm上表现同样的内容，它们的物理表现形式一定会区别很大）。但它也同样适应于全新的应用设计或重新进行设计。这种方法迫使你先考虑正确的方面：内容组织方式和任务流。

第二个方面，用“物理结构”把内容用页面、窗口、面板等元素表达出来。事实上，很难将内容的组织和表现完全分离开来。设备和网页的物理形式会给设计带来严格的限制，在桌面应用上，窗口结构是一个重大的设计决定。所以本章有一部分内容专门讲述它。

信息架构基础知识：切分内容

THE BASICS OF INFORMATION ARCHITECTURE: DIVIDING STUFF UP

在前言里，我提到过界面习惯用法¹。你可能还记得，它们是人们已经比较熟悉的各种界面类型，包括文本编辑器、表单、游戏、命令行、数据表格等。这些习惯用法很有用，因为它们可以让你用一些熟悉的元素开始设计，而不需要从头开始。而且，一旦一名初次访问的用户认出这些熟悉的地方，他也会对这个界面有初步理解。

不管要建造什么应用，你可能已经决定了要用哪些界面习惯用法。但是，当你在用这些习惯用法组织你要展示的内容时，可能就不会那么清楚了。如果你的应用很小，小到可以用一个页面或面板来表示，那么，好，你可以继续了。如果要处理大量的功能、工具、内容，也是没有问题的。既然在高科技行业常常是功能影响销售的，自然需要在这些界面上塞上满满登登的内容。

如果以前建造过网站，你可能知道“信息架构”（Information Architecture）这个词。信息架构是需要一开始考虑的。需要弄清楚怎样把这些内容和功能建构起来：怎样组织它们的关系，怎样标注，怎样引导用户通过界面达成他们的目标。就像一名建筑师一样，你是在规划人们居住的“信息空间”。

但是现在的软件应用和以前的网站不同了。用“名词”和“动词”的方式来思考一下。在网站和许多其他媒体（书籍、电影、音乐、平面媒体）上，你在和名词打交道。你排列它们，展示它们，对它们分类，给它们添加索引。用户知道怎样处理文本和图片，以及其他类似的对象。但是，根据它们的定义，应用软件就是用来完成某件事情的：写作、绘画、进行事务，和其他人交互，以及进行跟踪记录。你是在和动词打交道。你需要对界面进行组织，来让用户知道接下来该做什么（或者至少知道该去哪里寻找）。

大多数应用程序（以及许多网站）是根据下面这些方式来组织的，其中一些使用名词，另一些使用动词。

- 对象列表，例如一个满是邮件的收件箱
- 动作或任务列表，例如，浏览、购买、出售、注册
- 某种主题类别的列表，例如，健康、科学、技术
- 工具列表，例如日历、地址簿、写字板

可以根据下面这几个互相关联的因素进行选择：软件的本质和所在的领域（主题）；用户的领域知识；用户的计算机熟练程度；还有最重要的是，你的软件有多贴近用户已经建立的领域心智模型（心智模型是用户基于以前的经验或了解，包括类别、词汇表、过程、原因和结果等，它表示用户对某个东西的认知程度，也就是他认为他已经知道的那部分）。

1. 术语“习惯用法”（Idiom）来自Scott McCloud的*Understand Comics*，在这本书里，它用来形容一种发展出自已类型的词汇表、手法和内容的工作流派。另一个术语可能是“类型”（Type），在Malcolm McCullough的*Digital Ground*里面用来形容构架性的形式和惯例。

很多UI设计上的问题可以追溯到这个地方的糟糕决策，或者更恶劣的是，把几种类型的组织方式混在一起——例如把工具和主题类别混合到某个含糊不清的导航条上。

另一方面，有时候可以把几种组织方式混合起来。例如，一些有趣的小规模UI创新就来自在一个菜单上同时提供名词和动词。这么做的可用性取决于它的上下文。还有，可以在把这种区分应用到软件顶层分类的同时，也应用到它们内部的各个分类级别。不同的界面部分需要不同的组织方式。

再次说明，这并不是什么高精尖技术。你在别的地方也可以看到这些概念。但是某些时候，很容易默认选择一种区分方式而不必仔细考虑哪种是最好的。通过把它们列出来，让它们可视化并且可以进行讨论。这一点对于本书讲述的许多模式和组织模型来说都是通用的。

让我们在近距离看看这四种分类方式，来了解一下它们最适合的情况。

对象列表

LIST OF OBJECTS

在很多情况下，使用这种组织方式的理由很明显。一组邮件、歌曲、书籍、图像（参见图2-1中的iPhoto示例）、搜索结果、金融事务——我们每天都在软件中遇到它们。从这些列表中，可以得到类似的界面习惯用法：用来编辑的表单、用来播放的媒体播放器、用来查看内容的网页。

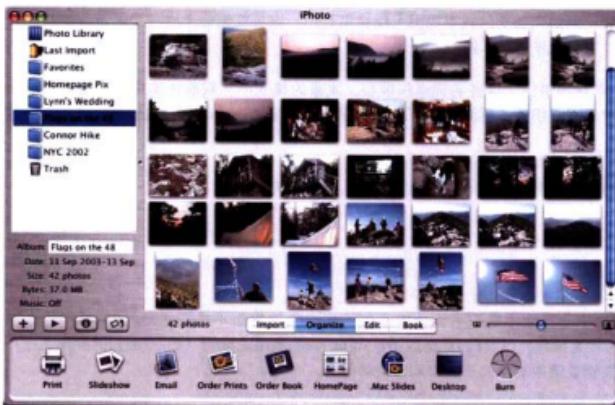


图2-1 iPhoto里面的照片列表。这些照片根据相册进行排序，并以表格的形式显示出它们的缩略图

也可以在可选择的列表、表格、树或其他类似的地方看到这样的对象。一些用户界面非常有创意。手机的电话本是其中一种极端情况，它的屏幕大小有限，只能顺序排列，一次只能显示几条信息，来供你快速浏览。而TiVo把它们的电视节目显示在一个多级菜单里，你必须通过几次单击来进行遍历。另外，最先进的邮件客户端可以让你以各种复杂的方式进行排序和过滤。当建立这些类型的界面时，要保证让设计相应扩展，并小心地用界面提供的功能去适应用户的能力和需要。

在这样的界面中组织和展示对象有很多方面要注意。那是你作为信息架构师的下一个任务，下面是一些最常见的模型。

- 线性模型，通常会进行排序
- 二维表格，通常也会排序，并让用户可以通过列的标题进行排序，或根据某些规则进行过滤
- 可以把项目归类到某种类别（以及子类别）中的层次模型
- 可以体现彼此关系的层次模型：父/子、容器等
- 空间组织方式，例如地图、图表或类似桌面的区域，让用户可以把某些东西按他们的需要放在那里

事实上，所有这些模型（除了二维表格）可以用在用来分隔界面的四种区分方式上：对象、任务、类别、工具。具体怎么用取决于人们希望用软件来做什么，哪种模型最符合他们的心智模型，最符合问题域的自然组织方式——如果有的话。

例如，如果需要为城市里的公共汽车设计时刻表，自然的组织方式可能是按照公共汽车或路线号码进行组织。线性的路线列表是可行的，但不是每个人都知道他们想要乘坐的公共汽车线路号码，因此某种空间组织方式，例如交互式的城市地图可能会更有用。你可能也会考虑一个区域层次结构，这些区域里的车站，以及经过这些车站的线路。

第6章“显示复杂数据”详细讲述了这些名词组织模型。本章的模式中，双面板选择器（Two-Panel Selector）模式通常用来组织这类界面，还有单窗口深入（One-Window Drilldown）模式也是这样的。

然后，一旦用户选择了某个对象，他们将会对它做什么？请继续往下看。

动作列表

LIST OF ACTIONS

这种方式是以动词为中心，而不是以名词为中心的。这一类的界面不是问用户“你将操作什么对象？”而是“你想进行什么操作？”这样的界面包括从TurboTax的高层决策树（图2-2是它的界面之一）到某个正在编辑的文档或选中对象的一长串动作菜单等等。

这样做的好处在于，它们都是用很朴素的语言来描述的，人们只要从字面上来理解就可以了。如果对应应用软件的领域有很好的了解，并且定义了合适的服务，这样设计出来的服务会容易使用。即使对新用户也是这样。

如果遇到大量的动作，这样设计就会成问题了，用户可以使用的动作可能有很多。动作太多比对象太多更容易让用户不知所措。

Welcome to TurboTax for the Web

Start a new 2003 tax return
Click here to start your 2003 tax return.
We'll walk you through the process step by step. Remember, you can come back to finish at any time and you don't pay until you're ready to file.

Continue my 2003 tax return
Click here to sign back in and continue a 2003 tax return that you've already started.

图2-2 <http://turbotax.com>上的一种友好的任务组织形式，用动词“Start”（开始）和“Continue”（继续）来描述，并提供了有帮助的解释说明

桌面应用程序用菜单和工具条来显示大量的动作，绝大多数用户都能明白这些熟悉的习惯用法，至少他们大体上知道。采用单窗口深入（One-Window Drilldown）模式的应用程序可以显示整页的菜单。只要这些菜单不会太长。另外，画布加调色板工具条（Canvas Plus Palette）模式讲述了一种在可视化建造器中非常典型的创建动作组织方式。事实上，整个第5章都在讲述界面上各种放置、排列和组织动作的方式。



但是小型设备（例如手机和PDA）界面的设计师们会受到一些有趣的限制。他们能做的常常是提供几项功能的单键选择：如果运气好的话一次可以显示三个，但通常只有一个或两个。对他们来说，关键是在任何给定的交互时刻确定哪些动作是最常用的，然后将这些动作分配到一到两个“软键”或按钮上（见图2-3）。小心谨慎地进行优先级排序是非常好的磨练方法，就算对Web和桌面应用也一样。

图2-3 这部手机包括一个电话本线性条目列表，在屏幕的底端，可以看到两个软键（它们下面的按钮说明文字是可以变化的）写着“Exit”（退出）和“View”（查看）。左边的按钮几乎用在所有应用的退出功能上（因此用户会习惯这个按钮）。而右边的按钮文字将随着你正在进行的任务而改变——它通常是每个应用中最常用的那个动作。

所有其他可能的动作都隐藏在菜单里头，可以通过中间带T型图标按钮进行访问。这种常用功能和不那么常用的功能划分是需要时时显示（Extras On Demand）模式的一个示例。由于一次显示所有的菜单是不可能的事情，因此设计师在确定哪个动作最重要时，要进行艰难的抉择。

主题类别列表

LISTS OF SUBJECT CATEGORIES

网站和在线信息一直都采用主题类别来对内容进行分类。因为有大量的内容供访问者浏览，按主题或类似的类别来组织它们会最有效率。但是成功的类别组织方式就像任务列表一样，取决于你对“用户第一次访问的时候他们会怎样看待界面”有多少了解。同样，你需要对应用领域有深入的了解，才能对应用用户已经建立的心智模型。

很多应用都不是这么组织的。主题类别在组织名词方面比较好，而在组织动作方面不太适用，面向动作的软件通常也不采用这种方式。也就是说，帮助系统——应该是应用系统设计的一部分——常常使用这种方式。另外，如果应用程序确实把知识检索和动作联系到一起（例如医疗应用和地图软件），采用这种方式会很方便。

图2-4显示了一个非常流行的例子。iTune音乐商店采用专辑、歌手和流派的方式来组织它的成千上万首歌曲。这同时也是它的用户在不搜索的时候对iTune目录的浏览方式。如果iTune采用其他方法来组织会怎么样呢？例如歌手们在2004年格莱美颁奖时头发的长度。这样的组织方式不会适合大部分人在音乐世界的心智模型（基本上如此）。他们将没法找到想要的歌曲。

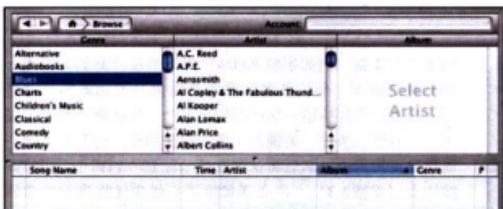


图2-4 iTune音乐商店用专辑、歌手和流派组织歌曲。iTune还添加了播放列表这个类别。这样的组织方式，再加上熟悉的媒体播放器习惯用法，就是iTune这个应用的核心。

无论在什么情况下，按主题类别组织可能会对你的应用程序很有用。相关的组织类型包括按字母顺序、年代顺序、地理分布，甚至受众类别（例如经常访问的用户和第一次访问的用户）。参见*Information Architecture for the World Wide Web* (“O'Reilly”出版）这本书来获取更多的信息。

工具列表

LISTS OF TOOLS

操作系统、掌上电脑和手机都提供了很多工具或子程序，供大家在它们的框架内部使用。一些应用程序（例如微软的Money）也是以这种方式工作的。还有部分网站也提供一些自己的Web应用程序，例如向导或游戏。

再次声明，要用好这种组织方式，需要用户对这些工具有清晰的、可以预先知道的认识。日历和电话列表很好识别。MS Money里面收支平衡工具和投资界面也是这样。如果你的网站提供一些新奇的命名古怪的小工具，而它们又和主题类别混在一起的话，用户往往不会去了解它们（除非他们有很强的动机）。

出于一些理由，基于工具的组织方式不适合用在各种类别混合使用的情况下，例如把工具和动作、任务及对象混杂在一起。混杂在一起让人们对于它们能做什么不知所措——不管是在菜单项、列表对像、按钮或链接上——尤其是那些还不知道工具名称的新用户，更是如此。你能观察到用户常常没法弄清楚这些东西，所以要特别当心。观察的时候，你应该留意他们觉得混淆的各种小迹象，但是不要直接去询问。

在另一方面，如果用户的目标不是要完成某个重要任务，而只是试探和玩耍，那么也许可以采用

这种策略。好玩的名字可能会吸引他们的注意力，让人们去单击它们看看它们到底是什么（参见本章有趣的分支（Intriguing Branches）模式）。但是在这种情况下，可预知性不是特别重要。而在绝大多数软件中，可预知性非常重要。

怎样才能组织一个工具列表的表现形式？由于通常一次不会出现太多，常见的组织方式是线性组织。PalmOS和其他很多小型设备使用网格来显示（见图2-5），这实际上是一个线性列表。为了容易查找，通常它们会按字母顺序排列，或者按期望的使用频率进行排列。如果这样的工具很多——用户也可能添加很多——的时候，你可以把它们按类别进行分组，就像Windows的开始工具条一样。也有些系统让用户把工具摆放在任何他们希望的位置。

在第3章的导航中，有一个模式叫做中心和辐条（Hub and Spoke），它常用来表示一个应用及其周边工具的组织模式。

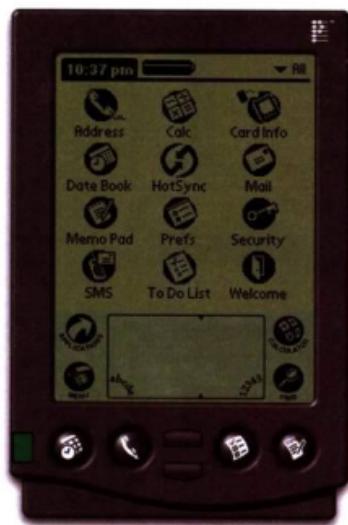


图2-5 PalmOS应用程序的屏幕是一个简单的线性工具列表

物理结构

PHYSICAL STRUCTURE

一旦有了一份设计的初步打算，你需要把它翻译成窗口、页面和空间的物理结构。这是人们对应用程序认知的第一个方面，特别是桌面应用。它们能容纳这里讲到的各种窗口排列方式。

我已经多次听到这样的辩论：一个应用程序应该使用多窗口、单一窗口加上几个平铺的面板，还是一个内容看起来像网页的窗口？要把这几种形式结合起来吗？参见图2-6。

也可能你已经知道该使用哪一个——你所采用的技术常常已经告诉你了。手持设备、手机及大部分其他的消费型电子设备根本就不会给你多窗口或多面板的选择。就算你可以这么用，也意味着不是什么好主意，因为用户将发现，若没有鼠标，实在是难以在上面这些模式上导航。

如果是桌面软件或大屏幕Web应用，你确实会有更多的选择。然而，没有什么必须遵守的规定来确定什么才适合任何给定的设计。但是下面的各节将给你一些参考。在决定之前，分析一下你的用户将进行哪些任务——特别是，他们是否需要同时在两个或更多的UI区域进行工作。他们需要在编辑面板B上的某个对象时参考面板A上的内容吗？他们需要对A和B进行并列对比吗？他们需要让面板A在任何时候都可见，从而可以用来监测某个东西吗？让你对用户任务的理解来驱动你的设计。

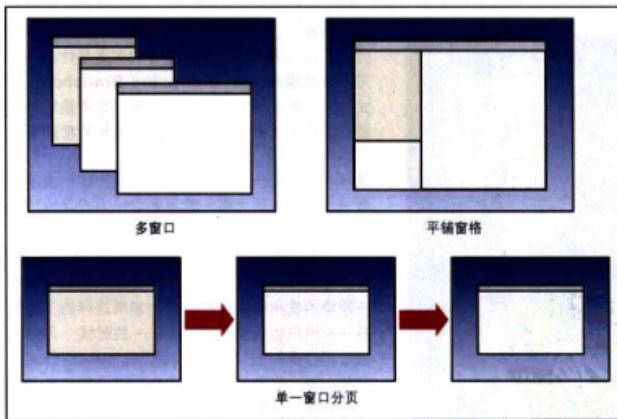


图2-6 三种不同的物理结构

多窗口

MULTIPLE WINDOWS

多窗口有时候是正确的选择，但这种时候并不多。它们在用户想要自定义屏幕布局的复杂应用中表现最好。不经常使用的用户，以及有时候经常使用的用户，也许会觉得多窗口让人恼火或引起混淆。

有时候，因为屏幕上一次显示太多的窗口，用户就会找不到想要的那些窗口。另一方面，如果用户确实需要看到两个或更多的窗口并行显示时，你可以采用这种方式或平铺窗格模型。

单一窗口分页

ONE-WINDOW PAGING

简单的Web应用最适合采用单一窗口分页模型，因为它一次显示一页。毕竟，这就是Web最开始的工作形式，而且人们对这种模型非常熟悉。还有，因为它能有效组织空间——除了能看到的内容之外，屏幕上再也没有别的了——所以它是小型手持设备和手机的首选模型。（总之，这个时候不能采用平铺窗口或多窗口的形式）。参见单窗口深入模式了解更多的相关知识，它演示了如何把一个层级的列表驱动或以任务为中心的界面组织到单一窗口模型中。

平铺窗格

TILED PANES

很多应用程序和Web应用在一个窗口中使用平铺的窗口。对于那些想要看到许多内容而不必管理窗口的用户而言，它非常好用。无数的窗口和对话框都是以双窗格结构的方式设计的，随着Outlook和类似应用的流行，三窗格的方式也越来越常见了。人们直觉地了解到了这样的知识：在一个窗格中单击，在另一个窗格中查看。

不过，平铺的窗格会占用大量的屏幕空间。有时候我不得不在窗格变得太多的时候切换到多窗口模型，而且用户也不能一次在窗口中展开所有的窗格。

本章的第一个模式，即双面板选择器（Two-Panel Selector）模式就描述了一种有效利用平铺窗格的情况。可以在使用这个模式的同时再使用画布加调色板工具条（Canvas Plus Palette）模式。有些网站把小的交互内容模块放到其他常规页面上，这些小模块本身可能像单一窗格的窗口，但在页面上它们是平铺显示的。

在讨论集中界面还是开放界面的时候，平铺模型和多窗口模型一起构成了第1章中提到过的“open floor plan”的想法。使用平铺或多窗口的方式让人们可以一次访问几个对象，并且他们需要自己在不同窗口或面板之间管理自己的注意力。有时候某些技术限制了你不能使用平铺或多窗口的方式，还有些时候，你需要选择不要太依赖它们，而是使用一个简单的窗口来引导用户在预先设计好的路径中穿行。

所以，一切都需要权衡。当然，事实往往如此。最终，重要的是你的用户——不管是初级用户、中间用户还是专家用户——是不是能使用你建造的界面并乐在其中。用设计来演练，画出图片，建立界面原型。让你自己、你的同事，还有最重要的是，让用户自己来对它们进行试验。

模式

THE PATTERNS

本章的模式包括前面讲到的两种应用设计方法。其中一些模式结合了内容结构和物理结构。通过它们，我们可以了解到这些出色的结合，例如以下四种模式：

- 13 双面板选择器
Two-Panel Selector

- 14 画布加调色板工具条
Canvas Plus Palette

- 15 单窗口深入
One-Window Drilldown

- 16 可选视图
Alternative Views

接下来的几种模式没有讲述太多关于物理表现形式的内容，而是针对抽象的内容。向导模式讲的是如何在任务中线性化一种路径，它可以用任何数量物理表现形式来实现。需要时显示模式和有趣的分支模式则讲述了分隔内容的其他方法。

- 17 向导
Wizard

- 18 需要时显示
Extras on Demand

- 19 有趣的分支
Intriguing Branches

本书中提到的许多模式在不同程度上涉及了界面的可学习性。多级帮助模式讲述了如何把帮助直接集成到应用系统中，从而为大量用户和应用情形提供了可学习性。

- 20 多级帮助
Multi-Level Help

双面板选择器 Two-Panel Selector



图 7.7 Mac Mail

它是什么

把两个相邻的面板放在界面上，在第一个面板上显示一组对象，用户可以从中任意选择，然后在第二个面板上显示选中对象的内容。

什么时候使用

这个模式用在展现对象列表、类别列表，甚至动作列表的时候。收件箱里的邮件、网站栏目、歌曲库或图片库里的项目、数据库记录、文件——这些地方都可以应用这个模式。这些列表里的每个项目都有一些有趣的内容和它关联起来。例如邮件正文或文件大小、日期的详细数据等。你希望用户看到列表的整个结构，但同时也希望用户能以他选择的先后顺序，按自己的需要到达这些项目。

从物理上讲，你的显示设备要足够大，来同时显示这两个分隔的区域。很小的屏幕例如手机将无法应用这个模式，但Blackberry那样的屏幕可以。

为什么使用

双面板选择器模式是一种习惯的用法，而且它很常见、很有效。人们很快就学会了，他们如果在一个面板中选择一个项目，就能在另一个面板中看到它的详细内容。这样的知识可能来自邮件客户端、Windows文档管理器或一些网站。不管是哪里，它们都应用了类似的概念，从而看起来差不多。

由于两个面板相邻摆放，用户可以很快把他们的注意力换过来换过去，一会看着列表的整个结构（“我还有多少未读邮件”），一会查看一个对象的详细信息（“这个邮件的内容是什么”）。和其他物理结构例如两个单独的窗口或单窗口深入模式相比，这种紧密的集成有几个突出的好处。

- 它减少了体力开支。两个面板距离很近，用户的
眼睛不需要进行长距离的穿梭，而且可以用过一

- 次鼠标单击或按键来改变选择的项目，而不是首先要在窗口和屏幕之间选择（这可能又需要一次鼠标单击）。
 - 它减少了可视化的认知负担。当一个窗口弹出到最上面，或当一个页面的内容完全改变时（在单窗口深入模式会发生这样的事），用户突然得花额外的注意力到现在要看的东西上：如果窗口本身一直不变，例如在双面板选择器出现这种情况，用户可以把注意力集中在一个较小的变化范围内。
 - 它也减少了用户的记忆负担。想想邮件的例子：阅读邮件时，如果屏幕上只有邮件的正文时，就没有什么东西会提醒他这个邮件放在哪里。如果他想知道，必须进行回忆或者通过导航回到收件箱里的邮件列表。但是如果这个列表已经在屏幕上上了，他只需要看一眼，而不必记住什么。在这里，该列表充当了“你在这里”的路标功能（参见第3章路标模式的介绍）。

如何使用

把可以进行选择的列表放在上面或左边的面板上，显示详细内容的面板放在下面或右边。这样做利用了绝大多数用户的视线流动方向，因为他们有着从左到右的语言阅读习惯（如果用户习惯的语言是从右到左，记得把这两个面板的位置反过来）。

当用户选中列表中的一个对象时，在第二个面板中立即显示它的内容或详细信息。应该用一次单击完成选择。但是也别忘了为用户提供通过键盘改变选择的方

式，特别是上下箭头键——这样既减少了体力消耗，也减少了浏览需要的时间，而且还能支持只使用键盘的可用性要求（参见第1章中的支持模式）。

让已经选中的对象在视觉上突出显示。绝大部分的GUI工具有一些特别的方式来显示选中对象，例如，将选中对象的前景色和背景色反转过来。如果那样不好看，或者你使用的GUI工具不支持这一特点，试试给选中的对象换一种颜色和亮度——这样有助于和没有选中的那些对象区别开来。

可选列表看起来应该是什么样子？视两种情况而定：内容内在的结构特点，或者需要完成的任务情况。例如，大部分文件系统会显示路径层次结构，因为文件系统就是这样组织的。动画和视频编辑软件使用交互性的时轴。GUI Builder可能只是简单地使用它的画布本身，选中的对象把它们的属性显示在画布旁边的一个属性表（参见第4章）里。

可以考虑使用以下几个在本章介绍部分“对象列表”一节中介绍过的模型：

- 线性列表，通常是排序的
- 二维表格，也是排序的，它通常还让用户通过列标题或其他条件的过滤器进行排序
- 某种层次结构，把项目分组成几个类别（可能还有子类别）
- 某种展现关系的层次结构：父/子、容器，等等
- 某种空间组织方式，例如地图、图表，以及类似桌面的区域，让用户可以按自己的需要放置对象

你也可以在双面板选择器模式中使用可排序表格（Sortable Table）和树状表格（Tree Table）这样的信息展现模式（在第6章可以找到这两种模式）。再加上更简单的组件例如列表和树。卡片堆模式（第4章）和双面板选择器模式紧密相关，还有总览加细节模式也是这样（第6章）。

示例

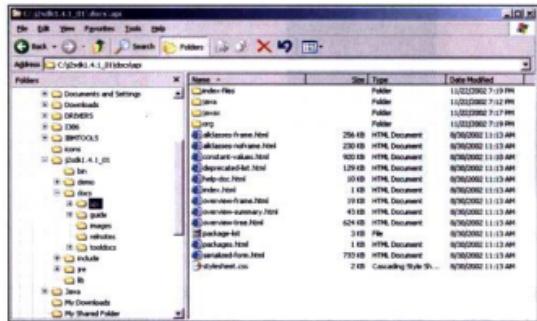


图2-8 Windows文件管理器可能是人们最熟悉的双面板选择器。它的内容通过可以选择的树进行层次化组织，反过来，Mac的邮件应用（见图2-7）采用了一个可以选择的表格。它是一种严格的线性组织方式。在这两个界面中，深色背景表示选中的对象。

当“选择-显示”的概念延伸到多个面板时，要使用某种层级信息架构来辅助导航，你就有了级联列表（Cascading Lists）模式（也在第6章）。

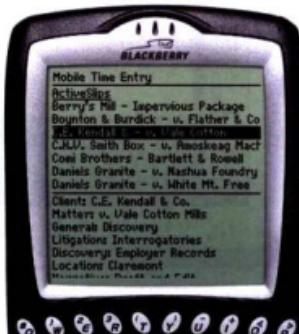


图2-9 Nortel的Mobile Time Entry应用是一个很少见的在手持设备中使用双面板选择器的例子。Blackberry的屏幕空间正好显示这两个大小合适的面板：当你在上面的面板中选择了一个对象时，它的内容将出现在底下的面板上（这两个面板都可以通过Blackberry的滚轮进行滚动，在右边可以看到该滚轮）。

实际上，这种界面非常有效。使用这个时间计费系统的律师们很容易就能发现他们想要的项目——这两个面板放在一起提供了充分的上下文，以及充分详细的信息，来快速准确地识别某个项目。

画布加调色板工具条

Canvas Plus Palette

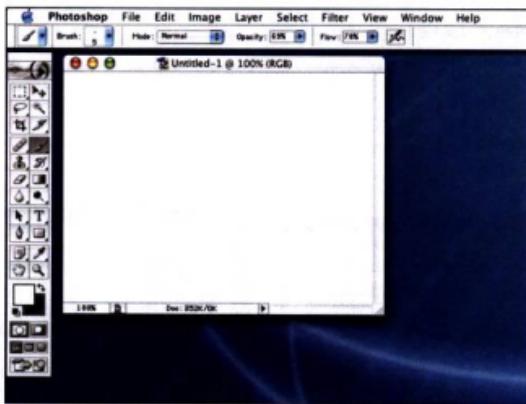


图2-10 Photoshop

它是什么

把一个带图标的调色板工具条放在空白的画布旁边；用户单击调色板工具条上的按钮，在画布上创建对象。

什么时候使用

在任何图形编辑器上都可以应用这个模式。这个模式的典型使用情形包括创建新对象，并把它们排列在某个可视的区域里。

为什么使用

这一对面板——调色板工具条用来创建对象，画布用来摆放对象——也非常常见，几乎每位桌面软件用户都看到过它们。这也是从熟悉的物理对象到虚拟屏幕世界的自然映射。调色板工具条也利用了可视化识别的好处：最常用的图标（画笔、手形图标、放大镜等）在各种不同的应用系统中一次又一次地得到重用，而且每次都是同样的用法。

如何使用

把一个大的、空白的区域提供给用户作为画布。它可能放在自己的窗口中，例如像Photoshop这样（见图2-10），放在一个平铺的面板上，或和其他工具一起，放在一个页面上。不管采用什么样的物理结构，都可以使用这个模式，只要你能在画布旁边看到调色板工具条即可。

调色板工具条本身应该是一个图标按钮或者看起来像按钮的网格。如果图标的含义很模糊，还可以附上文本；有一些GUI builder的工具条就把GUI工具的名称列在它们的图标旁边。Visio也是这样，用工具条把复杂的可视化构架工作划分到具体的领域中。但是图标化的外观很有必要，因为这样可以让用户看出它是一个工具条。

把工具条放在画布的左边或者上面（可能有着从右到左语言习惯的人更喜欢它在画布的右边而不是左边，如果需要考虑这点，请进行一些可用性测试）。可以把工具条图标划分成几个小组。你也可能希望使用卡片堆（Card Stack）模式，例如用Tab来表示这些分组。

大多数工具条按钮都应该在画布上创建一个图形对象。但有些builder也把它们和其他意图成功地集成到工具条上了。例如缩放、索索等。很早以前就有这样的做法了：MacPaint把它的模态（mode）加到了调色板工具条上（见图2-12），人们发现箭头、手状图标及其他图标也可以这样做。不过要小心，我建议你不要把其他动作掺杂在创造性工具条上——这样可能会让用户产生混淆。

通过调色板工具条，不同的应用程序有不同的对象创建用法。有的只用拖拽的方式，有的使用单击工具按钮然后单击画布的方式，还有些使用一次性模态（One-off Modes，参见第8章）和弹性加载模态（Spring-Loaded Modes）模式，以及其他一些小心设计的用法。我发现，可用性测试在这里非常重要，因为与用户的期望常常有着很大的差别。

示例

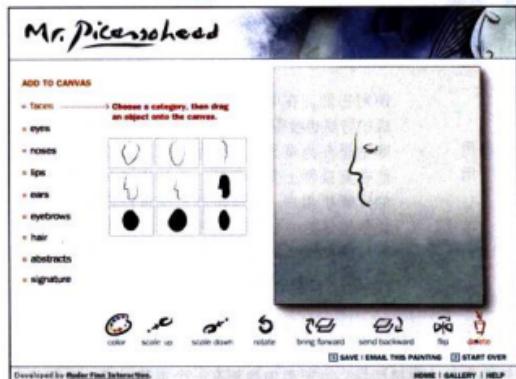


图2-11 在应用画布加调色板工具条模式时，你不必用上以文档为中心的桌面应用上的所有招数。这个Web应用，Picassohead先生，就是这个常见模式的有趣变形。工具条本身只是一个图标网格，它看起来也完全不像一组按钮；按钮分成了几组，并根据不同的类别放在不同的Tab上（这里也应用了双面板选择器模式）。当你单击文字“Eyes”、“Noses”或“Lips”的时候，工具条跟着变化，并显示这些对象。画布上本身没有什么东西，但并不是白色的。它的意图对第一次访问的新用户也很清楚，因为这是一大块开放空间，四周由它的边界包围起来。参见<http://mrpicassohead.com>。

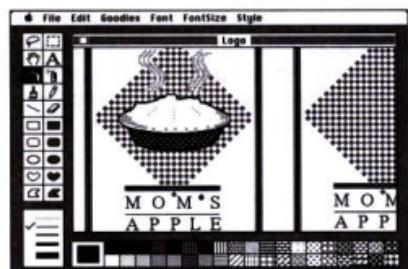


图2-12 回到从前，让我们来看看一个让这个模式流行起来的界面：MacPaint。这个模式从1984年以来就没有多大的变化——基本的元素都有了，空间布局也跟现在的软件一样，例如Mr. Picassohead和Photoshop。事实上，Photoshop和其他可视化建造器一样，20年来一直使用着当时MacPaint使用的图标。该屏幕截图来自<http://mac512.com>。

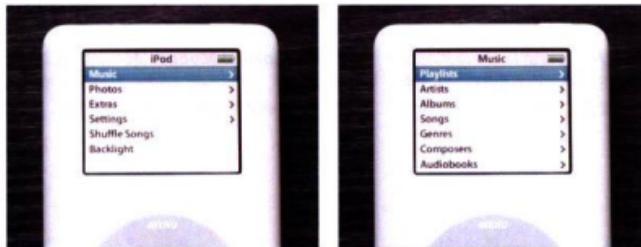


图2-13 两个iPod菜单

它是什么

在一个单一的窗口中显示应用程序的每个页面。当用户层层深入一组菜单项或某个对象的详细资料时，用新的页面完全替换当前窗口的内容。

什么时候使用

用户需要在你的应用程序中浏览很多内容页面或面板。它们可能按线性组织方式排列，出现在一个比较随意的超链接网络里，或者更常见的是，在一个菜单层级结构中。地址簿、日历、基于Web的邮件客户端，还有其他很多熟悉的应用都使用了这个模式。

你可能会遇到以下这两种约束：

- 你正在设计的设备有着严格的显示空间限制，例如手持设备（见图2-13）、手机或电视。在这些小规模的屏幕上，总的来说，双面板选择器和屏幕的窗格都是不切实际的，因为没有足够的空间来

使用它们。在电视屏幕上从一个面板到另一个面板的导航也会很困难，因为电视没有鼠标。

- 哪怕是在为桌面或笔记本屏幕进行设计，你也可能在复杂性上受到限制。你的用户可能不是熟练的计算机用户——同时打开很多应用窗口可能会让他们迷惑，或者他们不能从容地面对复杂的屏幕和需要技巧的输入设备。信息亭的用户就是这样。还有刚刚使用计算机的新用户也是这样。

为什么使用

保持简单。当所有内容都在一个屏幕上或一个窗口上的时候，每个阶段的选项会很明确，用户也知道，他们只需要把注意力集中在这里就可以了。

还有，有了单窗口和简单的后退/前进页面模型，现在所有人都知道怎样使用Web浏览器。人们已经习惯了当他们单击某个按钮或链接的时候，当前正在查看的页面将会发生变化，而当他们单击“后退”按钮的时候，将会回到刚才的页面。

可以使用多窗口来显示用户经过的不同空间——在一个主窗口中的单击将产生第二个窗口，再单击这个窗口又得到第三个窗口，等等。但那样做可能让人觉得混乱。（尽管他们可以看到几个并排的窗口，并把它们放在自己想要的位置，但是）就算是老练的用户也会一不小心就丢失了窗口焦点。

对于这里说到的这几种高度密集的模式和技术，单窗口深入模式就是一种可选方案。就像前面提到的那样，双面板选择器可能不合适，或者对于典型的用户来说，它让屏幕布局或交互变得过于复杂。平铺窗口、可关闭的面板、可移动的面板和级联列表模式也有空间和复杂性方面的问题。它们在小型的屏幕上不适用，而且它们也让针对计算机新手用户的界面变得太复杂了。

如何使用

你只有一个窗口可以使用——一个小设备的屏幕，一个大屏幕，一个浏览器窗口或一个和其他应用程序一起使用的桌面应用窗口。把你的内容划分到不同的面板，让它们优雅地放置到该窗口中：不能太大，也不能太小。

在这些面板上，设计出明显的入口来进入其他的UI空

间，例如带下划线的链接、按钮或可以单击的表格行。当用户单击其中之一时，就把当前的面板内容换成新的内容。这样，用户就可以深入挖掘到这个应用的深层内容上去了（见图2-14）。

用户怎么回来呢？如果你设计的设备有后退/前进按钮，那就可以用它们来解决这个问题。如果没有，那么创建这些按钮，并把它们放在窗口一个固定的位置上——通常是在左上角的位置，例如浏览器就是这么放的。你应该把表示“完成”或“取消”的控件放在面板上，让用户完成一项任务或选择的地方。这些控件可以给用户一种结束的感觉，因为“完成”了。

要记住，因为单窗口应用系统没有图形化的结构描述，也没有它们在这个结构中的位置信息，这样的应用将迫使它的用户依赖这些空间的组成方式。简单的线性或层级结构是最好的（见图2-15）。在可用性测试中，确定人们可以使用这些结构而不会迷路。面包屑（**Breadcrumbs**）和序列地图（**Sequence Maps**）这两个模式也许可以提供一些帮助，参见第3章“导航”。

中心和辐条模式的实现也常常使用单窗口深入模式，尤其是在Web和小设备屏幕上。同样，参见第3章。

示例

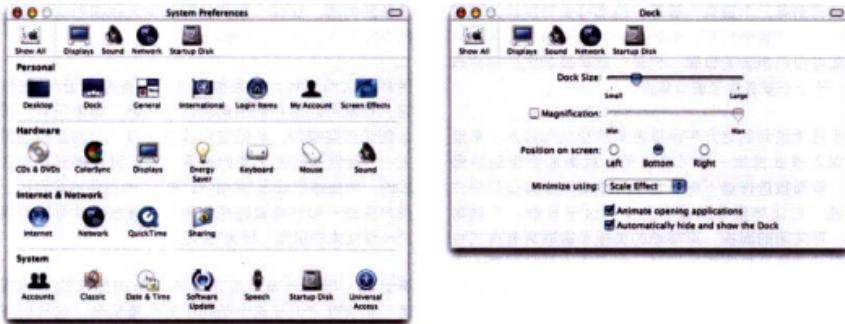


图2-14 Mac OS X系统的属性工具（System Preferences Tool）把所有内容放在一个窗口里。主面板在左边，可以深入进去的面板（可停靠）在右边。这里只有一级可以深入进去的面板。用户通过左上角的“Show All”按钮回到主面板。

Mac的屏幕通常很大，Mac OS X的用户熟练程度往往并不相同。系统属性（System Preferences）的设计师选择单窗口深入模式可能不是因为较小的屏幕，而是因为子面板数量与其多样性。主面板占用了大量空间，这个功能也可能并不适合采用平铺的双面板选择器窗口。

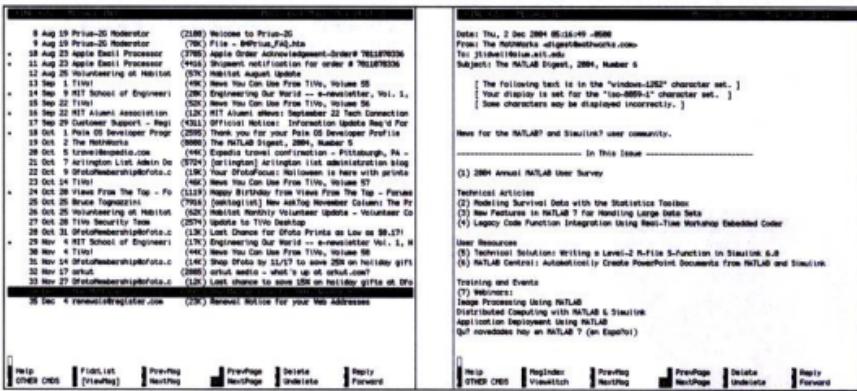


图2-15 Pine邮件客户端是一个轻量级的只支持文本的用户界面，完全由键盘操作。大于号键和小于号键用来在应用的层次结构中导航：主屏幕、邮箱列表、选中邮箱（左边）内部的邮件列表、选中邮件的正文（右边），以及邮件的附件。

因此Pine包括了UI空间的深度层次结构。但它在一个窗口中表现不错。把这个屏幕截图和双面板选择器模式中Mac Mail比较一下——两者都是良好的用户界面，但它们有着不同的设计。我们可以猜测，有两个迫切的需求使Pine采用了单窗口深入模式进行设计。要在每一个基本文本的终端窗口中运行，只能通过键盘操作。



图2-16 boston.com上一篇文章的标准显示格式和打印友好格式

它是什么

让用户在使用默认视图之外，还可以使用另外的可选视图。该可选视图对于默认视图来说，不光是装饰性的区别，还有结构上的区别。

什么时候使用

如果你在建造网页、编辑器、地图应用或其他任何显示格式化内容的应用，那么可以应用这个模式。人们在很多条件下使用它。也可以用在这样的情况下：可能你已经提供了某些可以自定义的特性——文字大小、语言、排列顺序、缩放级别等。但是这些轻量级的变化不足以支持人们通常需要进行的操作。

为什么使用

尽管你非常努力，但还是不能在一項设计中满足所有可能的使用情景。例如，打印就是一个典型的问题，因为信息显示的需求不一样了——导航和交互部件需要去掉，而剩下的内容需要应用新的格式来满足打印

纸的需要。参见上面的新闻文章例子（图2-16）来看看到底是什么情况。

根据不同的使用情景，有如下几个理由来使用可选视图模式：

- 应用在用户端的不同技术——一位用户可能在桌面使用，而另一位用户用的是PDA，还有一位用户使用的是屏幕阅读器，例如JAWS。
- 用户对于速度、可视化风格和其他因素的偏好不同。
- 为了深入了解，临时用不同方式查看某个东西的需要。

如何使用

选择一些该应用程序（或页面）的正常操作模式难以支持的使用情景，为它们设计专门的视图，并在同一个窗口或同一个屏幕上将这些视图作为默认视图的可选视图。

在这些可选视图中，可能需要增加一些信息，移除一些信息，但主要的内容应该或多或少保持不变。如果你要去掉一些界面元素——例如用来支持打印机或屏幕阅读器——那么应该考虑去掉那些次要的内容，减少或去掉图片，只留下最基本的导航元素。

如果要把内容塞到PDA或手机屏幕，情况稍微有点不同。你可能不得不重新设计导航本身。不是像你能在桌面应用和电视应用上那么做一样，把一大堆内容显示在一个屏幕或一个页面中，你可能需要把它们切分成好几个页面，每个页面刚好放到一个小屏幕上显示。

在主界面上某个地方放一个“切换”功能用来切换不同的视图。它不用非常突出：Powerpoint和Word（见图2-19）把它们的视图模式按钮放在左下角，这是在任何界面上来说都是非常容易忽视的位置。大多数应用

使用图标按钮来切换可选视图。同样，也要确认切换回默认视图非常容易。在用户切换过来切换过去的时候，保存应用系统当前所有的状态——选择，用户在文档中的当前位置，没有保存的更新，Undo/Redo的选项等。如果这些状态丢失了，用户会觉得很意外。

记得用户的应用系统常常会在不同的使用情景下保存用户的可选视图。换句话说，如果用户决定切换到一个可选视图，应用系统将会把该视图作为下次显示的默认视图。网站应用可以通过Cookie来实现这一点。桌面应用能够跟踪每个用户的偏好，数字设备上的应用（例如手机应用）可以简单地集中上次调用的时候使用的的是哪个视图。

网页还可以采用可选CSS页面的方式来实现可选视图模式。例如，这就是一些网站处理打印友好视图的方式。

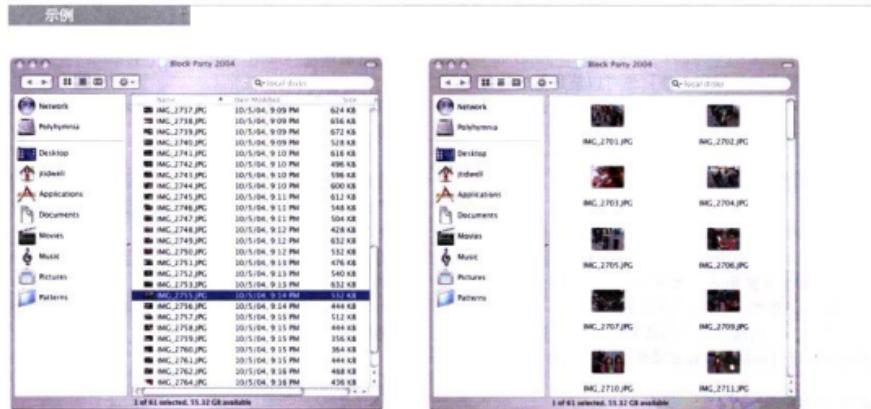


图2-17 Windows文件管理器和Mac的Finder都允许文件系统中的文件以几种不同的可选视图显示。这个示例显示了两种：一个排序的多栏列表（参见第6章的可排序表格模式），以及一个图标网格视图。

每种视图都有优点也有缺点。如果要管理大量信息，表格视图将会非常有用，例如用户可以通过在列标题排序来找到对象。但是图标视图在用户寻找自己记得的图片方面表现更好。这些视图解决了不同的使用上下文需求，而且用户也可能在同一个应用情景下前后切换这些视图。

prinz Microsoft PowerPoint -18.ppt
File Format: PDF/Adobe Acrobat - [View as HTML](#)
...With a **spring-loaded mode**, the user has to do something active to mode, essentially eliminating the chance that they'll forget what ...
[class.cailliet.edu:831/lectures/L8/ppt](#) - [Similar pages](#)

prinz Zulande
File Format: PDF/Adobe Acrobat - [View as HTML](#)
...**Zuland** "Spring-Loaded Mode" mit einer physischen Feder geklappt
...Zuland ist der Name des Zuges, der die **Machine-Mode Page** auf [www.soft.uni-ing.de/~zvogel/Zulande.htm](#) hat.
[Similar pages](#)

图2-18 Google的搜索结果不但可以是普通的HTML网页，还可以是PDF、Word和Powerpoint文档。要是你的机器上没有安装Word和Powerpoint程序怎么办呢？这个技术问题展示了可选视图模式的一种应用方式：“转换”成HTML。如果你实在不想下载一个大的Powerpoint幻灯片文件，只想马上浏览一下它的HTML转换结果，怎么办呢？这就是一种用户偏好。

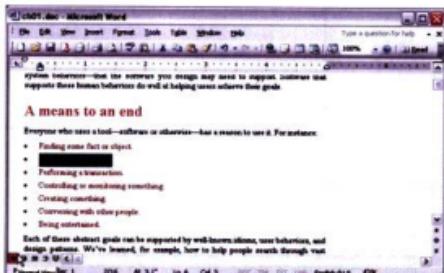
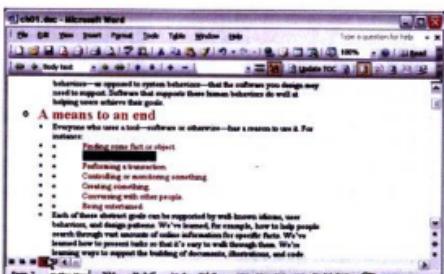
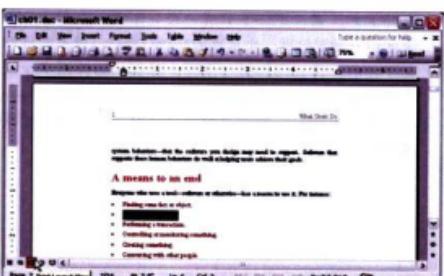


图2-19 当然，我们安装了Word和Powerpoint应用程序。这两个成熟的所见即所得编辑器可以创建出相当复杂的文档——Powerpoint幻灯片有顺序，有模板，可能其中一些幻灯片有注释。从一张幻灯片到另一张幻灯片之间还有好玩的跳转，甚至还有支持交互的代码。这个幻灯片文档的作者可能同时需要看到所有这些内容。而偶尔打开这个幻灯片的人不需要，同样，这也是不同的使用上下文。

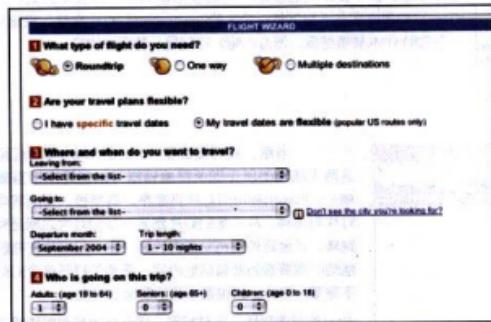
Word的视图包括：正常视图，用于绝大多数的编辑工作；一个打印布局视图，用来显示文档在打印页面上的效果；一个阅读模式的视图，仅供阅读；一个大纲视图，显示文档的结构。有些用户可能使用大纲视图来深入了解文档——例如，如果你打开了一份长长的不熟悉的文档，你可能切换到大纲视图来看到这份文档的总体目录。这两个应用都把可选视图按钮放在文档左下角的角落里。



要注意，这些Word例子里，用户切换不同的视图时，选中的文本将保持不变。文档的位置在视图切换时也不会变化。当然，不同的工具条会有所变化，缩放的级别可能会变化，还有其他一些内容——特别是脚注和注释——只会显示在打印布
局视图里。



可选视图

图2-20 来自`http://expedia.com`的航班向导

它是什么

向导是通过一步一步地引导用户完成任务的界面。在界面上一步步引导用户按预定的顺序完成任务。

什么时候使用

如果你在为一个又长又复杂的任务设计用户界面，而且它对于用户来说是陌生的——不是他们经常遇到或希望熟练掌握的那种界面。你有理由确定，你们这些设计UI的人比它们的使用者更了解如何才能更好地完成任务。

适合这种方法的那些任务要么有分支，要么很冗长而且单调无趣——它们包括一系列需要用户决策的步骤，然后走向不同的后续步骤。

要点在于，用户必须愿意在这种时候交出控制权。在很多情况下都没什么问题，因为人们在确定要做什么的时候，决策就是一种负担：不要让我考虑，告诉我接下来要做什么就好了。想想在一个陌生机场的经历——通常要是能有指示机场大致布局的路线标记，

那么你会觉得容易很多。你并不需要付出大量的劳动来学习机场是怎么设计的，因为你根本就不关心这个。

但是在其他一些情况下，就会有麻烦了。专家用户经常发现向导限制得很死，不能变通。这在需要支持创造性劳动的软件中尤其如此——写作、艺术创造、编码等。在用户确实想要学习系统的时候也是如此，向导不会告诉用户他们到底在做什么，应用系统的状态也不会随着不同的决定而改变。有些人会觉得这样很让人恼火。所以，要多了解你的用户！

为什么使用

分而治之。通过把任务分成一系列步骤，在每个步骤里用户只需要处理一个单独的“心智空间”，就有效地简化了任务。你必须通过一个预先规划的路线图来组织这项任务，从而不需要用户了解整个任务的结构——他们要做的就是按顺序执行每个步骤，相信他们如果遵循指示，就会成功完成。

对任务进行分解 “CHUNKING” THE TASK

把组成任务的操作分成几个部分或几组操作。你可能需要把这些部分按严格的次序排列，也可能不需要：可能只是为了方便而把任务分成1、2、3、4等几个步骤。

一次在线购买的切分可能包括以下屏幕：产品挑选、支付信息、支付地址、送货地址。它们的顺序并不重要，因为后面的选择并不依赖前面的选择。把相关的选择放在一起只是简化了人们填写这些表单的工作。

你可能想把任务在选择点进行切分，那样的话，用户的选择将会动态改变接下来的流程。例如，在一个软件安装向导中，用户可能决定安装可选包，这些可选包的安装则需要后面进行更多的选择。如果他们决定不采用自定义安装，那些步骤就会跳过。动态UI在表现这类有分支的任务时表现不错，因为用户永远不必看到那些和他的选择不相关的步骤。

不管在哪种情况下，设计这类UI最困难的部分在于权衡这些切分后的步骤数量和它们的粒度。如果只有两步，那会显得很傻，如果有十五步，也会漫长得让人受不了。另一方面，每个步骤不能太复杂，否则会得不到这个模式带来的一些好处。

物理结构 PHYSICAL STRUCTURE

每个步骤在单独的页面上，用前进和后退按钮进行控制，这是该模式最显然也最普通的实现方式。虽然这种方式并不总是明智的选择，因为分割到每个独立UI空间的步骤不能显示上下文——也就是说用户看不到前面和后面分别是什么。但是这种方式的向导也有一

个好处，它们可以让每个步骤拥有一个页面，可以在这个页面提供图标和说明文字。

如果采用这种方式，就需要允许用户在进行过程中随时向前或者向后移动。如果软件不能让你改变前面的选项而必须重新开始，那就太让人受不了了。因此，后退按钮是多页面向导的标准配置：把它们用上，而且还要确定该软件支持后退行为。另外，很多用户界面提供一份可选择的路径图或总体视图来表示所有的步骤，这样可以部分得到双面板选择器模式带来的好处（和那个模式相比，向导实现了一个预先制定的次序——哪怕只是建议——而不是完全随机的选择）。

如果你决定把所有的步骤放在一个页面上，那么可以使用第4章中将要提到的几个模式：

- 带标题的栏目（**Titled Sections**）。在标题上有固定的编号，因为一眼就可以看到所有的步骤，所以这个模式最适合用在分支不多的任务上。
- 响应式允许（**Responsive Enabling**）。在这个模式中，所有的步骤都显示在一个页面上，但每一步都必须等前面一步完成之后才能操作。
- 响应式展开（**Responsive Disclosure**）。在这个模式中，你想等用户完成前面一个步骤之后才把后面这个步骤显示在页面上。个人所见，我想如果向导的步骤不多，这是一种最优雅的方式了。它是动态的，很紧凑，而且容易使用。

不管你如何安排这些步骤，良好的默认值（**Good Defaults**）模式（参见第7章）都会非常有用。如果用户想要把这个过程的控制权移交给你，那么他也会希望你能为那些不太值得在意的选择提供默认值，例如软件的安装路径。

（在*The Design of Sites*一书中，作者把这个概念放在一个“过程漏斗”的模式底下。他们的模式更针对网站设计，例如像网站购物那样的任务，但是概念是一样的。）

图2-21 TurboTax是一个Web应用，它把几个步骤放在一个类似向导的界面上。每个主要的步骤都是一个单独的页面，而且就像通常的向导那样，它在这个页面的底端提供了“后退”和“前进”按钮，或者“完成”按钮（在这些屏幕截图上看不到它们，但是请放心，它们就在那里）。无论在什么时候，你都可以从顶部的序列地图（Sequence Map）上看到自己当前处在哪个步骤上。在这里没有使用良好的默认值模式。可能是因为这里涉及敏感信息——个人财务数据，不过我们可以发现，不少用户在很多字段里面填了“0”。



图2-22 这个Expedia的例子显示了一个应用了带标题的栏目（第4章）结构的向导；TurboTax的例子用的是页面模型，在每个后续的步骤里替换了浏览器窗口当前的内容。这里用的是卡片堆（Card Stack）模式（也在第4章），把所有的步骤放到一个非常有限的空间里：当用户从日历中选择一个日期，然后单击“下一步”的时候，第二个面板打开了。当这个步骤完成之后，用户往前移动到第三个面板。用户也可以在任何时候通过单击黄色的标题条回到前面的步骤。

还要注意，这个例子中窗口右边面板上预览（Preview）模式（第5章）的使用，它的标题是“你的路线”。这个模式跟踪了用户的选项，并且把这些选择进行小结显示出来了。这样很方便，因为一次只有一个向导页是可视的，用户看不到他所有的选择。参见<http://thetrain.com>。



图2-23 Windows 2000中的颜色对话框

它是什么

把最重要的内容显示出来，并把其他内容隐藏起来。让用户通过一个简单的操作访问隐藏内容。

什么时候使用

一个页面要显示很多内容，但是其中一部分内容不太重要。你希望把用户界面变得简单一些，但是不得不把所有内容放到某个地方。

为什么使用

简单的用户界面往往好过复杂的界面。对新用户或那些不需要所有功能的用户来说特别如此。让用户选择什么时候看到整个用户界面——他们比你更有资格进行判断。

如果你的设计可以让80%的使用情形更容易，而剩下的20%至少是可能达到的（需要用户稍做一些努力），那么这样的设计已经可以满足期望了。

如果应用得当，需要时显示模式可以为你节约大量的界面空间。

如何使用

狠下心来把用户界面删减到只有最常用的、最重要的那些项目。把剩下的部分放到它们自己页面或区域里去。并且，在默认的情况下，把这些区域隐藏起来：在新得到的简化界面上，放置一个醒目的按钮或链接到这些隐藏的内容上，例如“更多选项”。很多界面使用向右或向下的箭头作为这个链接或按钮标签的一部分，其他一些界面则使用“...”，特别在这个按钮将打开一个新窗口或新页面的时候。

该隐藏区域应该有另一个按钮或其他方式让用户关闭它。记住，用户在大部分时间里都不会需要它。只要把进入和退出这个额外页面的操作变得很醒目就行了。

在一些界面上，Windows直接打开相应的细节部分，然后当用户不用它的时候收缩起来。参见可关闭的面板（Closable Panels）模式（第4章）看看这种情况。不同的桌面用户界面提供另一种机制：例如，一个用来填充颜色的下拉框，包括一个标题为“更多填充颜色...”的项目，打开这个项目，将看到一个单独的对话框。

Experts warn of possible Web attack

Seeing a rise in hacker activity that could be a prelude to a broad Internet attack, security experts urged computer users to protect their machines by installing a free patch. Internet security firms issued similar warnings, saying they had seen increased chatter in online discussion groups and chat rooms. "We are expecting something sooner rather than later," said Dan Ingvaldsen, engineering director at Internet Security Systems in Atlanta.

[FULL STORY](#)

图2-24 Narratives经常使用需要时显示模式来把文章的摘要和全文区别开来。一位读者可以扫过前面的文字，例如CNN的这篇文章，然后决定是否阅读文章的其他部分（通过单击“全文内容”或标题本身）。如果他们不想看到内文，也没问题——因为他们已经阅读了最重要的部分。



图2-25 这是Windows 2000中的文件搜索功能。单击“搜索选项”将会打开一个显示额外选项的矩形框。和这类似的，单击搜索选项框的标题条，还有它向上的箭头符号，将会关闭这个选项框。在这里没有显示的是另一个级别的需要时显示模式：当用户取消“高级选项”的checkbox时，下面那些相关的checkbox都会消失。这种应用有点像响应式展开（Responsive Disclosure）模式（第4章），这个模式讲的是用户进行选择时，内容时而显示时而消失的副作用，和需要时显示（Extras on Demand）模式不一样的是，该模式需要一个显式的动作来打开或关闭相应的内容。

有趣的分支 Intriguing Branches

A political earthquake in the land of earthquakes (News)

By [gutted](#)
Fri Jul 25th, 2003 at 09:08:32 PM EST

While the rest of the world focuses on the deaths of the [Brothers Hussain](#), the rumblings of a [political earthquake](#) are threatening to bring California's government to its knees. On Thursday, Lieutenant Governor Cruz Bustamante, prompted by a petition signed by more than 1,600,000 people, called a snap election to recall the state's unpopular Democratic Governor, Gray Davis. It is the first recall of a Governor in the United States since [1921](#).

[Full Story](#) (185 comments, 2811 words in story)

图 2-26 来自 <http://kuro5hin.org>

它是什么

把到有趣内容的链接放在一些意想不到的地方，然后给它一个好玩的标签来吸引那些好奇的用户。

什么时候使用

用户沿着某种线性的路径前进——一段文本叙述、一项设计好的任务、一张幻灯片、一个Flash电影等。不过，有时候你希望在主要的注意力流之外提供一些额外的内容。这可能是一个故事的附加信息，例如图2-26里显示的那样。也可能是一些支持性的文本——示例、对概念的解释、术语的定义——或者一整段帮助文本。还可能是一项隐藏的功能，例如一个“复活节彩蛋”。

不管在哪种情况下，你都需要以一种优雅的方式把内容展现出来，从而让那些想要很快完成任务的人快速前进，而仍然能被那些希望看到它的人看得到。

为什么使用

人们都有好奇心，如果看到某个东西似乎有点意思，而且他们又有时间，想看看它是什么的时候，他们会去试试看。Web冲浪之所以流行起来，就是因为这种天生的好奇心和通过链接查看未知内容的意愿在起作用。对有趣分支模式进行有技巧的应用，让它变得有

意思，会让你的界面更加好玩。

创建有趣分支的一种传统用法是内联（inline）的链接（也叫做“嵌入式链接”（embedded links）），这种方式在Web上已经广为应用了。但是功能性的应用也许可以提供一种更有趣的用法。大家都知道，用户会有意忽略那些特别标记为“帮助”的东西。但是如果你把帮助性的内容放在某个链接（或者按钮、图标）后面，换一种方式来标注，例如“学习更多……”会怎么样呢？你可以利用用户好奇心的天性，让他们到达那个地方，学到一些他们需要的知识。

如何使用

从对用户的深刻理解开始。什么东西会让他们觉得有意思？界面上的什么地方是他们可能花时间去进一步探索更多的地方，哪些又是他们只想快速完成任务的地方？

为那些可能吸引用户的附加内容创建“入口”。这些入口可能是带下划线的链接（甚至在桌面应用里也是这样），大的标题、按钮、菜单项、图标、可以单击的图形区域——就看你认为怎样标记它们会引起好奇心了。这里头大有学问。如果有任何疑问，可以对一些有代表性的用户进行可用性测试。

对那些特别晦涩的启示（Affordances），例如图标或图形，你可能想要添加工具提示或其他某种类型的简短描述，来告诉用户单击的时候他们可能会到达什么地方。（不过，对于复活节彩蛋来说，没有任何提示本身就能带来某种乐趣）。

还有，需要为用户提供某种明显的方式回到他们刚才的工作流中。这个模式的意图是吸引用户阅读分支内容，然后回到他们先前的主要任务上：别让他们进入一种没有退路的状态。弹出窗口应该提供“关闭”按钮，浏览器界面那样的新页面应该提供后退链接或后退按钮。

示例



图2-27 Gmail的设置页面提供了一些明显和帮助有关的链接，但是它们采用了建议的表达方式，而不是“帮助”。看这里，一个“Learn more”的链接在键盘快捷方式标题的下面。这也是一种上下文敏感的帮助方式，和弹出菜单、帮助按钮、功能键一样。不过，和“Help”不一样，“Learn more”是一个主动性的词，而且它在页面上清楚可见，不像菜单项和功能键那样藏在后面。人们可以假定它会打开另一个网页，隐藏对它的操作完全是可以预计的。



图2-28 在Flickr浏览照片常常是对各种有趣分支的练习——它用普通Tag的方式在其他照片流、图片专辑和分组图片中提供了侧向分支（side trip）。这样做的结果是，产生了一种有吸引力（也非常受欢迎）的用户体验。



图2-29 一个普通一点的例子是Windows下面的Adobe PDF阅读器，一个粉红色的按钮“从你的桌面创建一份Adobe PDF文档”会把你带到Adobe网站上的一个页面，来解释怎样使用不同的产品来完成这样的任务。事实上，这个按钮的颜色和内容每隔几分钟就会变一次，每次都会显示一种不一样的效果。在某个应用程序里显示这样的链接，还是一种不多见的方式——绝大多数的应用不会使用价值很高的工具栏空间来做这样的事。

不过这些按钮对Adobe来说，谈得上是一种自我服务，因为它们鼓励你去了解其他的Adobe产品和服务。不幸的是，它看起来有点像广告。不知道在交叉销售和帮助用户完成那些必须完成的任务方面，它的实际效果会怎么样。

和具体内容
相关的提示

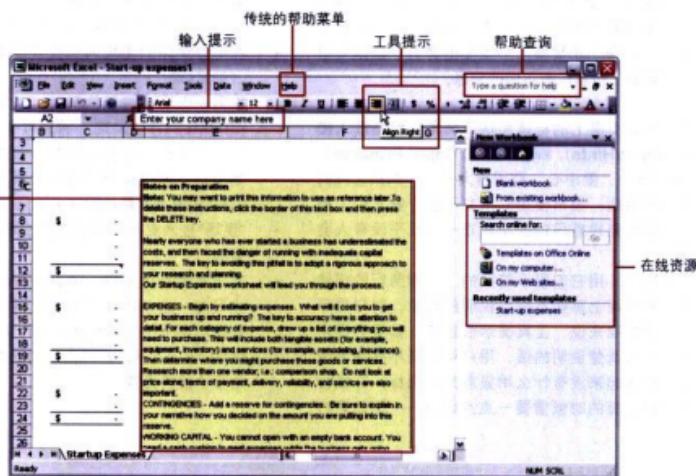


图2-30 Excel的不同帮助技术，这些不同的帮助技术都已经集成到了UI里面

它是什么

混合使用轻量级和重量级的帮助技术来支持用户的不同需要。

什么时候使用

你的应用很复杂。有些用户可能需要完整的帮助系统，但是你知道，大部分用户不会花时间去使用这个帮助系统。你希望尽可能地为那些没有耐心的用户，以及/或者那些偶尔使用的用户提供支持。特别是，如果你的应用主要面向“中级-专家”用户的话——怎样帮助用户从新手成为专家呢？

为什么使用

几乎任何软件产品的用户都需要各种级别的支持来完

成他们想完成的任务。那些第一次使用软件的人（或者在第一次使用的时候）和经常使用的人需要不同的支持。甚至在第一次使用的用户当中，承诺级别的学习风格也有着巨大的差异。有些人希望读一份指南，有些人不希望。大部分人觉得工具提示（tooltip）很好用，但也有少数人觉得它们很烦人。

在不同级别提供的帮助文本——甚至在它们看起来不像“帮助系统”的时候——也可以帮助到全部有需要的人。很多好的帮助技术把这些帮助文本放在容易获得的地方，但不会一直在用户的眼前晃啊晃的，所以不会引起用户的反感。不过，你的用户需要熟悉这些技术。例如，若没有注意到或者打开了一个可关闭的面板，他们可能永远也不会看到里面的东西。

在多个级别上创建帮助，包括一些下面列出的技术（不过没必要每个都用上）。把它看做一个连续的整体，每一个比前一个需要用户更多的努力，但是可以提供更详细的信息。

- 直接放在页面上的标题和指示文本，包括输入提示（Input Hints）和输入提醒（Input Prompts）（第7章）。要小心，别用得太多了。如果这样的帮助很简短，经常使用的用户不会介意它们，但是别使用整段整段的文字叙述——几乎没有人会看。
- 工具提示。用它们显示简要的，一到两行的描述来说明界面上那些不能自说明的功能。对只显示图标的功能来说，工具提示很重要，如果有提示对它进行清楚说明的话，用户可以毫不犹豫地使用一个看起来没有什么明显意义的图标。在鼠标移动到上面的时候需要一点点延时——例如两秒

钟——这样可以免去大部分用户可能的不便。

- 当用户选择或移动到某个界面元素时，动态显示稍长一些的描述。需要在页面上专门找一个区域来显示它们，而不是使用小小的工具提示方式。
- 在可以关闭的面板内部提供篇幅更长的帮助文本（见第4章）。
- 在另一个窗口显示的帮助。通常是通过浏览器，以HTML的方式呈现，有时候也在Winhelp或Mac Help中见到。这样的帮助一般是在线手册（一本整本书），通过菜单项或帮助菜单，或者对话框和HTML页面上的帮助按钮来访问。
- “现场”技术支持，通常是通过邮件、网络或电话提供的。
- 非正式的社区支持，通常是Web方式的。这种方式只对一些使用非常频繁，需要投入很多精力的软件适用——例如Photoshop、Linux、Mac OS X、MATLAB——不过用户会认为这样的社区支持是非常有价值的资源。

示例

MATLAB是一个复杂的多层软件应用，为所有这些层次提供帮助。下面每个例子都来自这个软件。

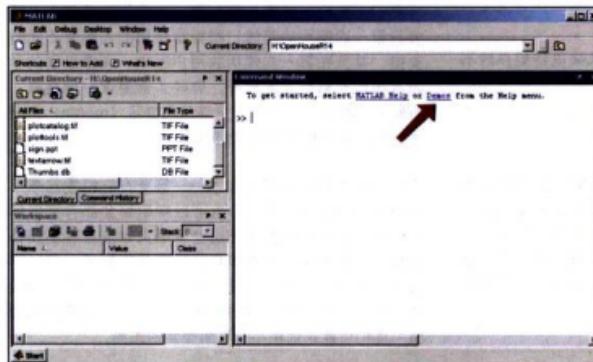


图2-31 当你启动MATLAB的时候，显示的命令行会立刻把你引导到帮助文档：“要开始使用，请从帮助菜单选择MATLAB帮助或演示”。你也可以看到快捷工具栏上有标题的那些项目，其中“How to Add”和“What’s new”这两个按钮将在另一个窗口显示帮助文本。

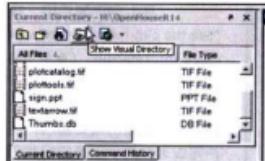


图2-32 主窗口工具条上的工具按钮都有工具提示。由于MATLAB是一个为中级用户和专家用户设计的应用，它把很多不太常用的功能打包到了很小的空间里，因此很有必要用工具提示来学习或记住这些按钮是用来做什么的。

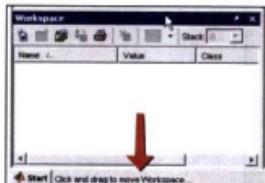


图2-33 在一些地方使用了移动时帮助，例如可移动面板的状态条。在这个状态条上显示的句子“用单击和拖拽来移动工作空间”，比工具提示要稍长一些，更重要的事，状态条不会那么突出自己——如果用户想忽略它，比工具提示还要容易。

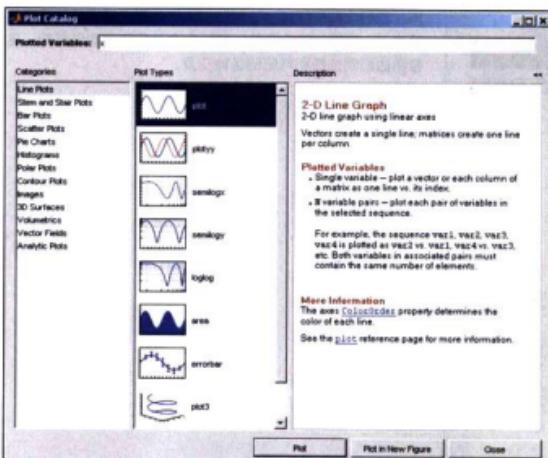


图2-34 在MATLAB界面上别的地方，选择一个对象的时候可能会引出一个简要描述，出现在可以关闭的面板上。在该窗口上，选择一个plot类型会引出描述性的面板来显示一个简短的帮助页面。这个帮助文本比通常需要的稍长一些（用户也可以关闭这个面板，如果他想关闭的话），但它更即时——因此和一个单独窗口的帮助相比，也更有可能用得到。

图 2-35 在上面的窗口中，在“See the plot reference page for more information”(查看 plot 参考页面获得更多信息)中单击带下划线的单词“plot”，将打开一个完整的帮助窗口。用户可以通过这个工具访问整个 MATLAB 的手册。

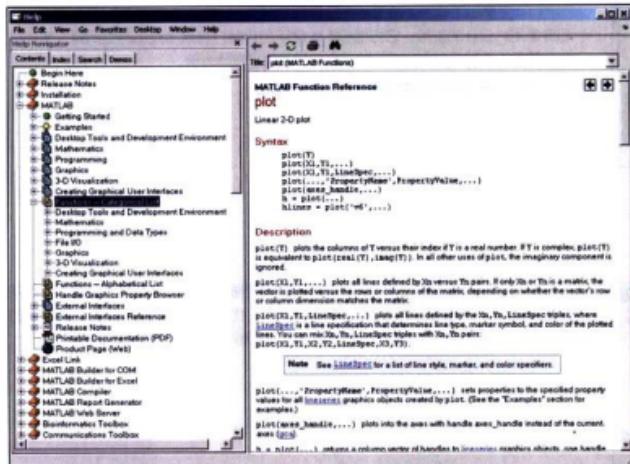


图 2-36 MATLAB 的用户也可以通过电话和网络得到技术支持。我们现在已经超出了软件设计的范围，但它仍然是产品设计——用户体验会沿着计算机中安装的软件向外延伸。这包括他们同公司和网站的各种交互。

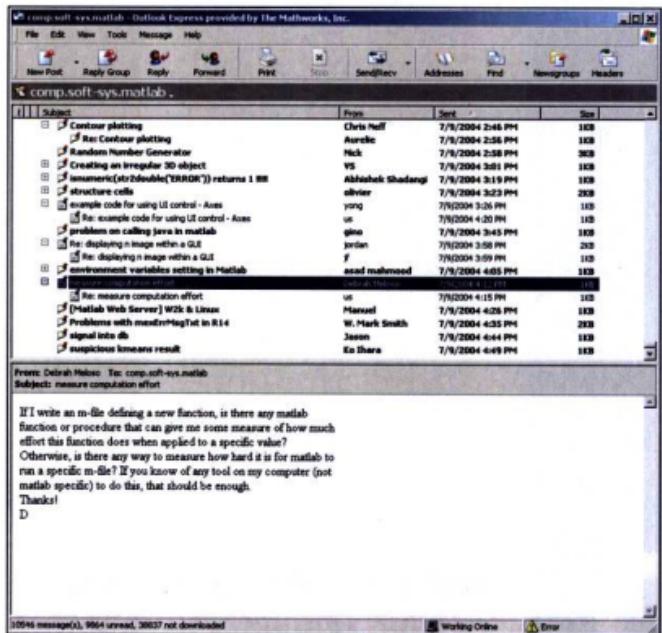


图2-37 最后,如果其他所有帮助资源都用尽了,用户还可以到广的用户社区去获得建议。在MATLAB的例子上,用户可以在一个专门的新闻组上询问和回答问题,这个新闻组就是comp.soft-sys.matlab(基于Web的讨论同样可以达到该目的)。像这样的社区只会出现在需要用户投入大量精力的产品上,可能由于他们每天上班或在家时都会使用它,也可能因为他们对它有着情感上的联系,就像很多人对游戏、Mac机或开源软件情有独钟那样。

03

到处走走：导航，路标和找路

GETTING AROUND:

NAVIGATION, SIGNPOSTS,
AND WAYFINDING

本章的模式将讲述导航问题。用户怎么知道自己现在在哪里，接下来要去什么地方，以及怎样才能到达那里呢？

因此，我把它叫做“问题”。在网站或Web应用里面导航就像交通一样。你不得不通过交通方式去你要去的地方，但是旅途很沉闷，有时候让人烦躁，花在路上的时间和精力就是一种浪费。就不能做点别的来打发时间吗？例如玩个游戏或完成点工作什么的。

最好的旅途就是——其他东西也一样——完全没有什么旅途。需要的东西就在手边，不用跑来跑去，那才是最方便的。同样，把尽可能多的工具放在界面上伸手可及的地方就会很方便，特别是对于那些正在向专家用户转化的中级用户来说（例如，那些已经对应用很了解了的人）。有时候你确实需要把那些不太常用的工具放在另外的对话框里，这样才不会把它们散落得到处都是。也有时候你需要把它们分组放在几个不同的页面上，让界面更清楚明白。这些都是可以的，只要用户在“路上”的距离不长。

因此，少就是多。让我们先来快速看看术语，然后再回到这个概念上来。

不要迷路

STAYING FOUND

假设你已经建立了一个大型的网站或应用，因为它规模很大，你不得不把它分成几个栏目、子栏目、专门的工具、页面、窗口及向导。在这种情况下，你将怎样帮助用户导航呢？

路标（**Signposts**）可以帮助用户找到他们所处的周围环境。常规的路标包括页面和窗口标题、网页标识（Logo）、其他有标记的设备、Tab、选择指示器。像全局导航、颜色编码的栏目、序列地图、面包屑层级结构、注释滚动条这样的模式（本章都会提到），将告诉用户他们现在身处何处，并经常会告诉他们再走一步就可以到达哪里。它们避免用户迷路，并且帮助他们计划后面的操作。

人们找到某种方式来达到他们的目标，这就是“**Wayfinding**”（找路）。这是一个自说明的术语，不需要什么解释。但人们怎样才能做到这一点完全是一个研究课题——来自认知科学、环境设计、网站设计的专家们就在研究它。下面这些常识性的特征可以帮助用户“found”（找到路）。

良好的标记

明白无误的标签可以预先显示你正在寻找的目标，并且告诉你该去哪里：你希望的标记都出现在它们应该出现的地方，永远不会让用户在某个需要决策的地方孤立无援。可以通过预先演练正在设计的界面来针对这一点进行检查，要覆盖所有的主用例路径。在每个用户需要决定下一步怎么做的地方，让它提供相应的标记或标签。

例如，你会在餐馆的后面寻找洗手间，或者在人行道和篱笆交叉的地方寻找入口。与此类似，你会在一个对话框的右边或底部寻找取消按钮，在网页的左上角寻找网站的LOGO。不过要注意，这些线索常常是由文化决定的。一个对这种文化不熟悉的人（例如一个从来没有用过某种操作系统的人）不会注意到它们。

地图

有时候人们从一个标记到另一个标记，或者从一个链接到另一个链接访问，但实际上并不知道他们在大背景中处在什么位置（曾经试过在一个陌生的机场穿行吗？可能和这种情况差不多）。不过有些人可能希望对整个空间有一个整体概念，特别是他们经常呆在这个空间里的时候。在标记不明确或建筑物很密集的空间里，例如市区附近，地图可能是他们唯一的导航辅助工具。

在本章里，清楚的入口点（**Clear Entry Points**）模式就是结合了良好的标记及环境线索的例子——链接必须在页面上突出显示。一个序列地图（**Sequence Map**）显然是一张地图。你可以使用总览加细节（**Overview Plus Detail**）模式（第6章）来同样为虚拟的空间显示地图。模态面板（**Modal Panel**）也是某种意义上的环境线索，因为一旦离开模态面板，你就会被带回到刚刚进入模态面板的地方。

在这里我把虚拟空间和物理空间进行了比较。不过，在虚拟空间里可以提供某种导航绝招，而物理空间在这一点上无能为力，绝招就是：逃生舱（**Escape Hatch**）模式。不管你当时身处何处，单击这个链接，就会回到熟悉的页面。这种感觉就像是随身携带了一个虫洞（*wormhole*，细小的时空管道），或者一双《绿野仙踪》里有魔力的红宝石鞋一样。

导航的代价

THE COST OF NAVIGATION

当步入一个不熟悉的房间时，你会四下打量。在很短的时间里，你就会对房间有初步印象，家具摆设、光线、出口以及其他线索；很快，你就会对这个房间进行一些假设，比如它是做什么的，是否能满足你进入这个房间的需要。接着你需要做一些你原来打算好要做的事。在哪里？怎么做？你也许能很快回答这些问题——或者不能，又或者你被这个房间中其他好玩的东西吸引住了（忘了原来的目的）。

和这种情形类似，打开一个网页或一个窗口都会带来一种认知上的代价。同样，你需要盘算这个新的空间：考虑它的外观、它的布局、内容、它的出口，以及怎样完成你到这里来想完成的工作。所有这些都需要时间和精力来完成，上下文切换迫使你重新集中注意力，根据你的新环境进行调整。

就算你对刚刚进入的这个窗口（或房间）已经很熟悉了，它仍然需要付出代价。不是一个很大的代价，但它会累积起来——特别是当你对显示一个窗口或载入一个页面所需时间进行计算的时候。

不管是网页、窗口、对话框，还是设备屏幕都是这样。不管是用按钮还是链接，用户下一步要去哪里的决定都是一样的，他们需要阅读标签，对图标进行解码，也需要在单击自己不太确定的链接或按钮时跨过信心的障碍。

让距离保持简短

KEEPING DISTANCES SHORT

一旦知道从一个页面跳转到另一个页面需要付出代价，你就能理解为什么减少这种跳转的次数很重要。通过选择常规的导航路径，把它们转变成容易的单次跳转，这就是本章的两个模式——全局导航（Global Navigation）和金字塔（Pyramid）模式——将带给你一些具体的方法，以便在复杂的应用和网站中实现这个目的。其他章节里的模式，例如卡片堆（Card Stack）模式（第4章），也可以帮助你把更多操作打包到一个单独的页面里，从而省去了跳转到新页面的麻烦。

不过可以从应用的结构中得到一些真正的效率（好处）。一种恶心的设计是，每次当用户需要完成一个简单日常任务的时候，都要迫使他们进入层次很深的子页面和对话框（如果把他们带到那里，然后告诉他们因为某些条件无法完成，接着再把他们送回第一步，就更糟糕了。这种状况我们都遭遇过）。

你能把应用设计成最常用的80%使用场景不需要或只需要一次上下文切换吗？

在某些应用中，这很难做到。你也许会想，有些工具必须呆在网站结构两个层次下面的对话框里，或其他某个地方。某个工具太大了不能放到你的主要页面上吗？试着压缩一下看看：去掉一些控件，把标签缩短一些，使用某些专门的能节约空间的控件（参见第7章）。在主要页面上包括各种各样的东西太分散啦？同样，压缩一下，把它们挪开，放到不得事的地方。你很需要那个向导吗？也许它的功能放在一个页面而不是很多页面上更合适。来点创意！

既然我告诉了你要遵守的规则，也会告诉你怎样打破这条规则。

有时候，确实有必要把一些功能藏到更深的页面里，需要多几个跳转也没有关系，例如那剩下的20%的内容（另外80%你已经让它们很容易访问了）。也有可能对你的应用来说，简洁的表达方式比减少一两次跳转更重要。需要时显示（Extras on Demand）模式（第2章）就是一种把很少用到的功能封装到一个额外的“门”后面的做法（同样使用了80/20原则）。同样，你需要有自己的判断并在有任何疑问的时候进行可用性测试。

现在的设计师们似乎对这些深信不疑。绝大多数的桌面应用都采用一个主窗口的形式，多个Tab页，或者平铺，带有分类工具条和面板。为每个功能使用单个的窗口或对话框已经不流行了，这样很好。Web应用也使用更多的客户端确认、逐步展开和其他一些技术来展示可变内容，而不需要每次有变化的时候再次刷新页面。

不过，有时候你还是需要在某个应用中用到四层的对话框（见图3-1）。

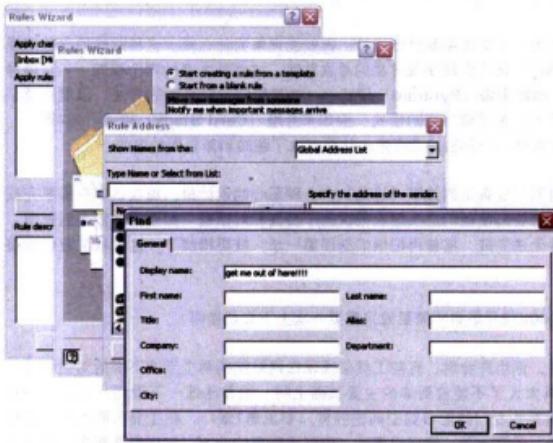


图3-1 Microsoft Outlook里面四极深的对话框

拼写检查：对话框、窗口、页面还是屏幕？

从表面上看，似乎网站和应用毫无疑问地在导航上的区别很大。其实不是这样。不管是网站、Web应用，还是让你从一页跳转到另一页的硬件设备，它们都经常让你在窗口或对话框之间移动。是的，你可以一次看到几个窗口，但是实际上用户一次只能把注意力放在其中一个上面，同时，在窗口与窗口之间转移注意力的成本也很高。

因为从这一点上来看，页面和窗口是差不多的，所以我打算从这里开始，在本章里简单地把这些虚拟空间叫做“页面”（Pages）。当你看到“页面”这个词的时候，可以假定我所讲的内容也同样适用于窗口、对话框及设备屏幕。

太多跳转

TOO MANY JUMPS

为了通过减少跳转的次数来演示我要讲述的内容，我将讲述一个简短的案例分析，一个实际上不是那么糟糕的交互式网站。任务可能很普通，尽管网站的设计者们显然没有为这个任务来优化网站。接下来你会看到我讲的是什么意思。

我正在访问Amtrak的网站来找出从波士顿坐火车到纽约大致需要的花费。因为出发时间和到达时间都很灵活，所以我打算看看几种可能的选择来找到一个最便宜的方式。很高兴的是，它们的主页上有一个“快速费用寻找”的工具——我只要一步就可以得到费用信息！而我所有要做的工作只是填写这个小小的表单（见图3-2），单击Go按钮，就行了，对吗？

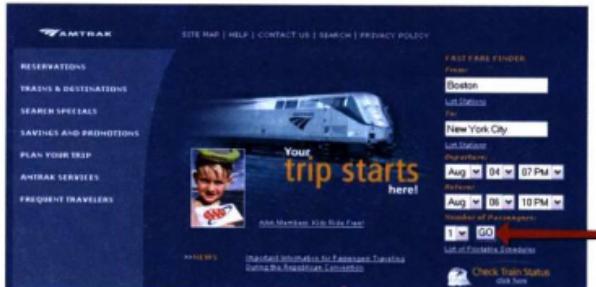


图3-2 Amtrak的主页

1. 单击Go按钮。

估计还不行，看起来他们让我再次填写同样的信息（见图3-3）。

This screenshot shows the "Schedules and Fares" section of the Amtrak website. At the top, there are links for TRAIN STATUS, RAIL LINKS, ACCESSIBILITY, REVIEW RESERVATIONS, and LOGIN. A message says "Select a valid station from the station list below. Thank you." Below that is a "Schedules and Fares" heading. It includes a note for new customers to create a member profile and links for logging in or going to Florida. A message states that for travel plans involving more than a simple one-way or round trip, users should use the "Multi-City Trips" page. The search form has "Departure Station" set to "Boston South Station (BOS), MASSACHUSETTS" and "Arrival Station" set to "New York Penn Station (NYP), NEW YORK". The "Number of Passengers" field is set to 1. The "Departure Date" is Aug 4 at 07:00 PM, and the "Return Date" is Aug 6 at 10:00 PM. A red arrow points to the "Show Schedules" button. At the bottom, there is a checkbox for "Show Acela Express and Metroliner only." The Amtrak logo is at the very bottom.

图3-3 日程和费用

2. 单击显示日程的按钮。

关闭，但是费用没有显示出来（见图3-4）。我猜想自己必须选中其中一个选项，而这三个选项可能都有不同的价格。我选了第一个，看它的价格是多少。

The screenshot shows the Amtrak website's "Your Departure and Return Options" section. It lists three train choices for the trip from Boston South Station, Massachusetts to New York Penn Station, New York. The first choice, "Acela Express" (train 219), is selected with a radio button. A red arrow points to this selection. The second choice is "179" and the third is "67". Each row includes service information, departure and arrival times, and a "Accommodations" column with a "F" and "B" icon. Below this, another table shows the return trip from New York Penn Station back to Boston South Station, with one option selected.

Choice	Service	From	To	Departs	Arrives	Accommodations
<input checked="" type="radio"/>	219 Acela Express	Boston-South Station, MASSACHUSETTS (BOS)	New York-Penn Station, NEW YORK (NYP)	6:45pm 08/04/04	10:15pm 08/04/04	F B
<input type="radio"/>	179	Boston-South Station, MASSACHUSETTS (BOS)	New York-Penn Station, NEW YORK (NYP)	7:25pm 08/04/04	11:30pm 08/04/04	B U
<input type="radio"/>	67	Boston-South Station, MASSACHUSETTS (BOS)	New York-Penn Station, NEW YORK (NYP)	9:45pm 08/04/04	2:01am 08/05/04	B R

From	To	Departs	Arrives	Accommodations
New York-Penn Station, NEW YORK (NYP)	Boston-South Station, MASSACHUSETTS (BOS)	9:07pm 08/06/04	1:08am 08/07/04	B U

图3-4 你的出发和返回选项

3. 选择第一个单选按钮，向下滚动，单击“Show Fares”（显示费用）的按钮。终于，出来了！156美元。现在让我再选一个，做点比较。

4. 单击“Return to Train Availability”（回到座位情况）。

现在，回到了“Your Departure and Return Options”（你的出发和返回选项）（见图3-5），这回，我要选择第二趟列车。

AMTRAK

RESERVATIONS HOME | MODIFY TRIP | ACCESSIBILITY | LOGIN | MEMBER PROFILE | ITINERARY AND FARES

Your Fare Information

Please note this is not a ticket.

From Boston South Station on 06/04/04 to New York-Penn. Station on 06/04/04
 From New York-Penn. Station on 06/06/04 to Boston South Station on 06/07/04

Service	From	To	Departs	Arrives	Accommodations
2191 Acela Express	Boston-South Station	New York-Penn. Station	06/04/04 8:45pm	06/04/04 10:15pm	Acela Express Business Class Seat
136	New York-Penn. Station	Boston-South Station	06/06/04 9:07pm	06/07/04 1:00am	Unreserved Coach

Rail Fare: \$ 156.00
 Accommodations Price: \$ 0.00
TOTAL FARE: \$ 156.00

[Click here for important identification requirements for Amtrak travel.](#)

[Book Selection](#) [Return to Train Availability](#) 

图3-5 你的费用信息

5. 再次单击“显示费用”的按钮。

结果是128美元，还不错。

6. 再次单击“回到座位情况”。

现在选择第三趟列车。

7. 再次单击“显示费用”的按钮。

这趟列车也是128美元。

七个步骤、七次上下文切换——载入了八个页面——而这些工作我原本以为在一个步骤中就可以完成。真是惊人呀！

图3-6显示了这个用例用UML序列图来表示导航的样子。每个竖线代表一个页面，每个箭头代表一次跳转，每个长方形显示我当时所在的页面。如果你沿着页面上的箭头往下走，这个序列图显示了这次交互在时间上的进程。

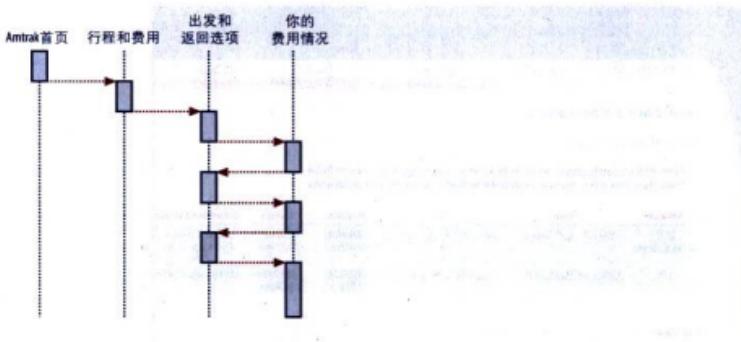


图3-6 Amtrak的顺序图情景：四个页面，七次跳转

怎样才能把这个过程减少到一个步骤呢？

首先，我们可以把一开始多余的“日程和费用”页面去掉吗？它只是简单地确认我们已经输入过的信息。嗯，我们已经知道这个页面要求用户填写一项更关键的信息：实际的出发和到达城市。我们可以用两个客户端下拉列表框来解决这个问题：一旦输入了城市，就会自动填写默认的火车站（不管怎样，默认值常常是用户想要的那个火车站）。另外，我们也可以在另一个页面再完成它。随着这个页面的告别，我们又少了一次跳转，现在只有五次了。

接下来的一个改进——一个大改进——就是简单地在火车车次信息同一个页面显示费用。我们可以在表格中增加“费用”一栏，这就去掉了其他的五个步骤！图3-7显示了这个得到大幅度简化的导航。

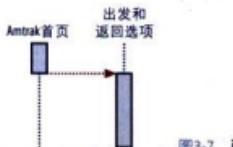


图3-7 两个页面，一次跳转

模式

THE PATTERNS

总的来说，本章讲述了导航的几个方面：总体结构，知道你的当前位置，找出你的目的地，然后快速到达那里。这些模式在不同程度上讲述了这些问题。

一个应用或网站的结构和它的导航密切相关。而实际上，不可能把这两个方面完全区分开来。也许有人可以把结构相关的导航模式，例如清楚的入口点（Clear Entry Points）、中心和辐条（Hub and Spoke）及金字塔（Pyramid）模式放到第2章中。不过第2章主要讲的是如何组织内容，而这些模式讲的是多个页面如何相互关联。在基本导航结构设计好之后，你可以再加上一些其他模式——全局导航、颜色编码的栏目、序列地图、面包屑层级结构、逃生舱——到一些单独的页面上，从而可能把设计进程往下细化到页面布局中（第4章的主题）。

还有一件事要注意。清楚的入口点、模态面板及中心和辐条模式限制（而不是扩展）了导航的入口（其他很多模式则相反）。任何地方任何时候进行完全自由的移动并不意味着完美；有时候理解上的简化意味着少了更好。

不管怎样，我们将从描述导航结构的模式开始：UI的页面（或者窗口/对话框）如何内部关联。如果将这些页面之间的连接画出来，你会得到下面这些模式。

21 清楚的入口点
Clear Entry Points

24 金字塔
Pyramid

22 全局导航
Global Navigation

25 模态面板
Moda Panel

23 中心和辐条
Hub and Spoke

26 序列地图
Sequence Map

28 注释滚动条
Annotated Scrollbar

27 面包屑层级结构
Breadcrumbs

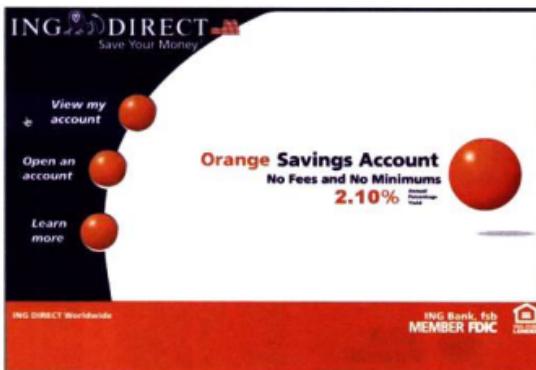
29 颜色编码的栏目
Color-Coded Sections

当用户从一个地方跳转到另一个地方的时候，动画转换有助于让他们保持方向感。这只是一个视觉上的小技巧，没有别的了，但是它非常有效，可以保持用户在哪里和什么事情正在发生的感觉。

30 动画转换
Animated Transition

最后还有逃生舱（Escape Hatch）模式，就是先前提到过的导航“王牌”绝招。页面上有了它，用户就知道他们可以随时回到一个熟悉的地方。

31 逃生舱
Escape Hatch

图3-8 来自<http://ingdirect.com>

它是什么

只在界面上显示几个入口，让它们面向任务，并具有自描述性。

什么时候使用

如果你正在设计一个面向任务的应用，或者其他任何主要供新用户或不常访问的用户使用的应用时，可应用此模式。此模式对一些网站来说也同样适用。不过，若该应用的目的对每个开始使用的人来说都非常清楚，而且如果大部分用户可能会因为多余的导航恼火的话（就像为中级、专家级用户设计的那些），不要使用它。

为什么使用

在打开一些应用和网站的时候，展现在用户面前的界面看起来像一片信息和结构的沼泽：大量平铺的面板，不熟悉的术语和用词，毫不相关的广告或不能用

的工具条。对于犹豫不决的用户来说，这样的界面没有提供清楚的起点。“好，我已经在这里了，现在怎么办？”

为这些用户着想，列出几个可以开始使用的选项。如果这些选项可以满足用户已经有的期望，他可以自信地选择其中一个，开始工作——这就带来了即时满足（**Instant Gratification**）模式的好处（第1章）。如果没有，至少他现在也能知道这个应用实际上可以做什么。因为你已经在眼前定义了几个重要的任务或类别。这样做，已经让应用变得更具自我解释特性了。

如何使用

当人们访问网站的时候，或者应用启动的时候，就把这些起点当做进入网站或应用主要内容的“大门”。从这些起点，逐步引导用户清楚地到达应用的内部，直到他有了足够的上下文知识可以自己继续为止。

总的来说，这些入口点应该涵盖那些“人们为什么来这里”的主要理由。可以只有一个或两个入口点，也可能有很多：这取决于哪种方式更适合你的设计。但是你应该使用第一次访问的用户能理解的短语——这不是展示应用独特工具名称的地方。

在视觉效果上，你应该根据它们的重要程度来显示这些入口点。例如，在图3-8中，我们可以清楚地看到，ING Direct希望把人们指向他们当前的特别储蓄账户，他们把它放在前面中间的位置，采用了加粗的文字和鲜明的颜色。周围是空白。其他三个任务（可能大多数客户经常使用它们）则放到了一个分组中，每一个有着同样的字体大小、而最常用的入口点“View my account（查看我的账户）”在最前面。

示例

小屏幕设备也可以从这个模式中得到好处。绝大多数的PDA和手机都会设计成类似中心和辐条（Hub and Spoke）模式的样式。那样的话可以从一个开始屏幕上选择应用工具。但是当应用工具太多了之后，你是把它们都放在一起，就像PalmOS那样，还是反过来，把最常用的那些分离出来？显示太多可能的情况会让

在一个这样的页面上，绝大多数的网站都会列出它们其他的导航连接——关于我们、联系我们、我们的隐私策略等——这些内容的字体会更小，也只有那些真正寻找它们的人才会看到。它们有专门的位置，并且它们也不会把你带到网站的核心位置，就像家里的车库门不会把你领到客厅一样。

（顺便要提一下，启动画面并不是清楚的入口点模式的应用方式。因为它们不会为用户提供一个选择的地方，而只是把当前已经就范的用户从一个屏幕传递到另一个屏幕。除了显示某个东西正在载入的进度，或者显示设计者的骄人事迹之外，没什么别的作用了。）



图3-9 Google最有名之处就是把一件事做到极致。它的首页设计把用户的注意力集中在这一件事上：你不可能看不到这个搜索框。其他的内容排在第二级（例如网页或图片）和第三级。加上一些实用工具导航（例如高级搜索）。像ING Direct的网站一样，这种设计的简洁让用户有信心在几个主要选项中决定其中一个。



图3-10 本图是一个摩托罗拉手机为六个特性提供单键访问的设计：四个从中间的四向按钮开始，两个从屏幕底端的软键开始。手机中的其他应用都要在菜单上再按两次键。显然，电话簿很容易到达——可以通过某个人的左手大拇指下面的按钮直接选中（我怀疑这里更重要的是方便，而不是信息过载，不过它在这两个方面都表现很好）。

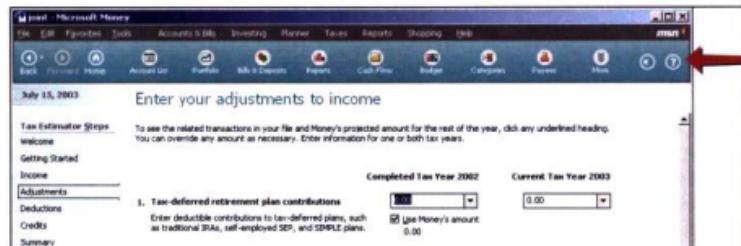


图3-11 来自微软的Money

它是什么

在每个页面上用一个小栏目来显示一组一致的链接或按钮，让用户可以通过它们来访问网站或应用的核心栏目。

什么时候使用

如果你在建立一个大型的网站，或者有几个独立栏目或工具的大型应用，那么可应用此模式。在这两种情况下，用户很可能直接从一个栏目向另一个栏目移动。你拥有足够的空间，也不介意在每个页面上显示一些额外的元素。

中心和辐条（Hub and Spoke）模式是这个模式的另一个可选方案。它更适用于独立的工具、带有高度限制的工作流，或者在屏幕受到限制的情况下使用。

为什么使用

在Web上，全局导航条是众所周知的习惯用法，所以用户期望它出现在应该出现的位置。然而更重要的是，一组链接或按钮反映了界面上最高层次的结构，从而让用户在每个页面都能看到该结构。这样，人们可以

看到用户界面的总体视图，并有助于在需要的时候找到它们（无论如何，如果命名机制有意义的话）。它也有助于探索网站，只要一个单击就很容易地从任何地方到达每一个栏目。

你也可以在全局导航面板上加上一个标记，来显示当前用户的位置，就像Tab那样。于是它就成了一个“你在这里”的路标，也是另一个导航元素。

如何使用

首先，设计一份有意义的组织结构。把栏目的数量减少到你能在合适的地方放得下为止。然后好好对这些栏目命名——让他们对用户有意义，不要使用过多的词语，并且遵循任何可以适用的习惯用法（例如“关于我们”或“产品”）。

对全局导航面板来说，设计一个看起来一模一样的简单区域，在每个页面它出现的地方，都显示在同样的位置。对于Web来说，应该是所有页面上（在使用中心和辐条结构的应用中有一些例外情况）。桌面应用UI在这种情况下的习惯用法少得多。但它仍然应该在每个主要的应用窗口中保持一致（不是说每个对话

框）。一个好的全局导航面板应该是一个良好定义的视觉框架的一个组件（参见第4章）。

为了显示用户现在在什么位置，简单地让当前栏目的链接看起来和其他栏目不一样就可以了。可以使用一种对比颜色，或者一个像箭头那样的简单图形。

你可能会遇到的一个设计问题，特别是在网页上，那就是怎样把这种导航设备和其他许多链接一起显示。它们应该彼此有所区别。用户也许会在页面的顶端寻找全局导航，这就把左边和右边的区域留给了其他链接。或者你可能把它们放在内容区域。你也可能使用两组完全不同的提示方式——例如使用简单的可单击文本作为全局导航，而使用Tab来表达更“局部”的内容。

和中央舞台（Center Stage）（第4章）模式一起使用的时候，记得首页和主窗口可能需要采用与其他页面不同的页面布局。如果首页或当前打开窗口的目的是到达不同的栏目，那么这里的全局导航需要比别的任何地方更突出。你也可能想要使用详细信息或子链接让它们变得更丰富。

最后要知道，不是每个用户都会用到或者注意到这样的一种导航设备。工程师和设计师都普遍错误地认为，用户在逻辑上会先看总体情况，然后决定去哪里。其实不是这样的。用户通常不会注意网站或UI是怎么组织的，而只是简单地沿着最近的，或者最明显的路标往前走，直到他们找到想要的内容为止（这种现象叫满意即可（Satisficing），参见第1章）。就像某个人在机场寻找洗手间一样——如果有一些标志或建筑上的线索，他们很可能不愿意去地图上找。

示例

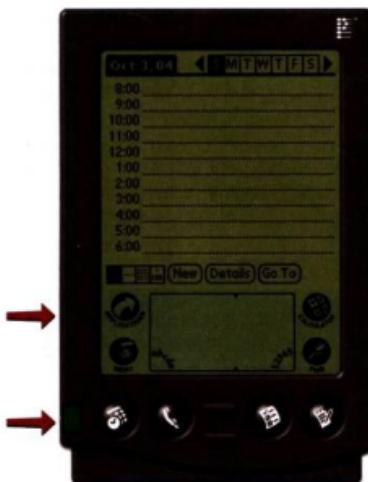


图3-12 Palm和类似的PDA使用一种硬件形式的全局导航。PDA底部的四个圆按钮会把用户直接带到日历、电话清单、TODO清单和记事簿应用。和它们类似的是，上面的丝印软键将把用户带到Applications hub、计算器、两个帮助工具即“菜单”和“寻找”上。

注意，日历应用没有明确的显示关闭或退出按钮。我猜想设计师期望人们使用全局导航来操作，这一点暗示着当用户在应用之间切换的时候，应用将保持它们的状态，因此不会有什麼变化会不经意地丢失。这一点跟中心和辐条模式（下一个模式）有很大的不同。



图3-13 Nokia 60系列的手机界面

它是什么

把应用的各个栏目分割成几个迷你的小应用，每个应用都有一个入口（从主页面），并且可以回到主页面。

什么时候使用

你的UI包含几个相互之间关系不大的任务、子应用或内容元素（例如表单、演示、游戏、文章、事务或一些独立的工具）。你可以从一个中央页面或中央屏幕来访问所有这些部分。不过，因为以下几个可能的理由，你不希望把它们相互关联起来：

- 为了加强子应用之间的独立性
- 对工作流进行限制，来保证某个任务的完成（或显式地取消）
- 减少视觉和认知上的混乱
- 受到物理空间的限制

如果用户有一些充分的理由要在这些组成辐条的工具

之间直接移动，那么你可能不会使用中心和辐条模式。在那种情况下，可以使用全局导航模式。这两个模式是对一个相似问题的两种不同解决方案——对用户的典型移动路径进行组织（它们可以共存。但往往在日常使用中，一个模式会终结另一个模式的统治地位）。

中心和辐条模式特别适用于小屏幕的移动设备，并且常常和单窗口深入（One-Window Drilldown）架构一起使用。

为什么使用

在应用中心和辐条模式的时候，你用导航把用户体验组织成某种与Web上自由浏览或桌面应用中从窗口到窗口跳转等不一样的体验。你希望用户一次只关注一个部分，然后回到中心，再到达另一个部分。这样肯定会减少辐条页面的内容混杂程度——用户不需要看太多，也不需要想太多。

但是把可选的导航路径通过交互式页面进行限制，你也可以防止错误的发生——这样，用户就不会那么容易碰到自己的脚。没有导航按钮，他们不能留下一个填了一半的表单而跳转到另一个页面（“编辑丢弃”问题）。使用桌面应用上的模态窗口，他们不可能丢失屏幕上的对话框。总的来说，他们不太可能把UI弄到一种不一致或混乱的状态。

还有，受到限制的导航意味着你对界面要处理的情况有了更强的控制力——这样常常会产生更简单（因而也代价更低）的实现。这种结构也有着良好的可扩展性。当有了更多功能时，你很容易在Hub页面增加更多的辐条链接。

最后，你可能只是想要把用户对辐条的体验设计成简单、独立的UI。这并不总是可行的——事实上，它可能对那些习惯了自由导航的用户来说完全是一种恼火的体验。但是，要怎么做由你来决定。它的好处之一是，用户会对“可以完成哪些任务”非常清楚，因为中心和辐条的结构在许多上下文中都很直观明显。单击一个辐条，进入要完成的任务，然后当要完成其他任务的时候，回到中心页面就可以了。

如何使用

以任务或工具的方式把内容切分成独立的迷你应用。以合适的方式来设计每个迷你应用。这些迷你应用就是辐条。

在中心页面，为那些辐条设置链接。在每个辐条页面（每个辐条也可能有好几个页面，例如，如果使用了向导模式的话），去掉所有分散主任务流注意力的所有导航链接。

只留下相关的动作，例如后退按钮和下一步按钮，取消、退出或者可能有一些其他不会干扰主任务流的动作（例如帮助功能等）。

当用户到达某个辐条任务的结尾时，为用户提供某种表明他已经完成的方式，例如清楚标识的“完成”或“取消”按钮。这两个按钮都应该把用户带回中心页面。

Bob Baxley的书*Making the Web Work: Designing Effective Web Applications*（《让Web运作起来——设计有效的Web应用》，New Riders出版社出版）首次提出了这样的概念：基于中心的（hub-based）体系结构非常有用，而且有效。

示例

在小设备的世界里，中心和辐条模式非常普遍，但在普通应用和Web应用中很少见到纯粹的中心和辐条模式。我想这是因为对于Web应用和桌面应用来说，这种模式过于局限了。那种每个工具都采用一个模态对话框的日子已经终结一段时间了——很高兴和它们说再见！

偶尔也确实会看到某个网站以这种方式组织。例如，提供免费在线游戏的网站常常在游戏界面中去掉大部分导航。TurboTax的网站也采用了这种形式。TurboTax的首页有着跟平常一样大量的链接——全局导航、二级导航、产品链接等——但是三个“产品”的分组采用了中央舞台模式。



图3-14 TurboTax首页的这个部分为用户提供了三个产品，单击其中任何一个产品的“Get Started”都会把你带到下面那些页面中的一个。

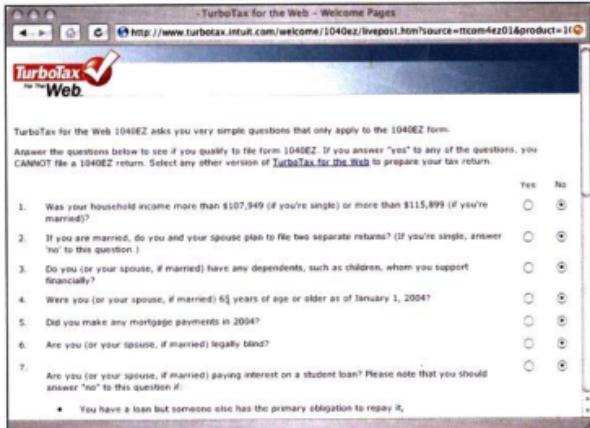


图3-15 除了“Continue”（或“Sign in”，或其他任何把你带到第二步的标志）和几个帮助链接之外，这些页面几乎没有导航元素。这个画面显示了1040EZ和Basic的起始页面。注意一下那个光秃秃的标题和侧导航条，用户不会有任何疑问就会被自动引导到税务表单那里去。

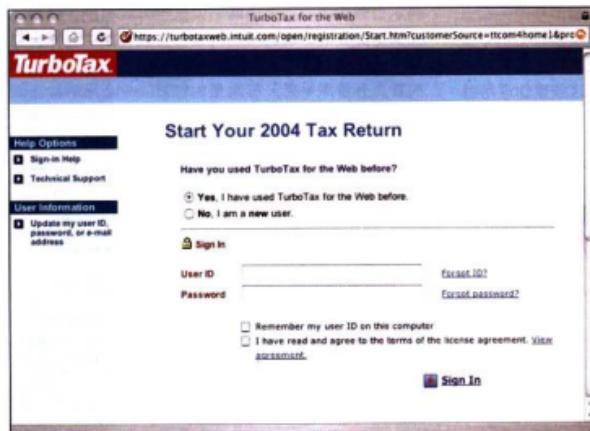


图3-16 这个图形显示的是第一个页面，不过这次是基本税务表单

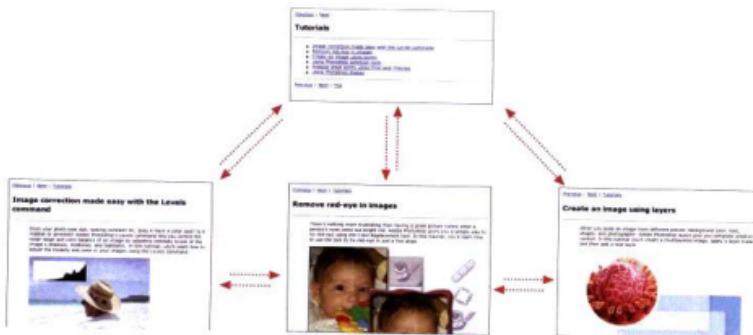


图3-17 Photoshop的帮助手册

它是什么

使用后退/下一个 (Back/Next) 的链接来关联一系列页面。把这种序列化展现方式和一个主页面结合起来，由该主页面链接到（并链接回）这个序列中所有的页面。

什么时候使用

网站或应用有一系列的页面，用户可能看完一个后去看另一个，例如幻灯片、向导、书里的章节或者一组产品。用户从一个“父”页面到达第一个页面，然后可能再去访问序列中其他的页面，父页面在这里成了一个跳转点。

如果有一个页面链接到许多类似但语义上并不关联的页面时，你可能也会考虑使用这个模式。

和设备及网页相比，金字塔模式似乎经常能在单窗口应用中找到。它常常和单窗口深入（One-Window Drilldown）模式一起使用。

为什么使用

这个模式减少了到处遍历需要的单击次数。它提高了导航效率，同时也在页面之间表达了一种更加顺序性的关系。

使用后退/下一个（或上一个/下一个）的链接或按钮都可以。人们知道怎样使用它们。但不是很有必要把用户锁定在一个页面顺序中难以脱身：已经访问了七个页面，然后他需要单击后退按钮七次才能回到开始的地方？这样可一点都不好玩！

通过在每个序列页面放置一个回到父页面的链接，用户得到了更多的选择。现在，你有了三个导航选择而不是两个——后退、下一个、向上（up）。也没有让它变得复杂很多，但一个随意浏览的用户（或者一个中途改变主意的用户）将会在到达他想去的地方上，减少很多次单击。这样对用户更方便。

与此类似，把一组并不关联的页面链接在一起，对确实想要阅读那些页面的用户来说很友好。如果没有后退/下一个的链接，他们可能需要一直从父页面跳过来跳回去。也许他们会就此放弃，拂袖而去。

如何使用

这个模式很简单：把后退、下一个及向上的按钮或链接放到每个序列页面上，并且把每个序列页面的链接放在父页面上。它组成了一个图3-18所示的拓扑结构——因而也有了“金字塔”这个名字。

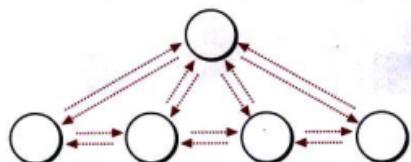


图3-18 典型的金字塔链接拓扑结构

你不需要把按钮在字面上标记为“后退”、“下一步”或“向上”——例如，你可能使用相邻页面的实际名称作为后退和下一步的标签，尽管这样多余的表达方式可能并不值得占用它所需要的屏幕空间。如果用户没有寻找这些特别术语的时候，还是得由你自己来决定。通常最好把向上按钮标记为“回到<页面标题>”或其他的形式。

还有，把这三个链接放在一个小小的页面空间通常会很方便，因为它最小化了所需的鼠标动作，也可以建立空间记忆。甚至，可以做得更好的是，把下一步的链接或按钮放在每一页相同的位置——这样用户在从一个页面移动到下一个页面时，完全不必移动鼠标。

绝大部分金字塔模式都是在静态线性序列的情况下应用的，就像幻灯片那样。设计师只要简单地把一个页面放在另一个页面后面就可以了。完全没有理由说明为什么你不能在有分支的序列上使用这个模式。在那种情况下用户会选择一条路径。可以考虑一下。

该模式的另一个变种是把一个静态线性序列转换成一个循环，这是通过把最后一个页面链接到第一个页面而不必回到父页面来完成的。有时候可以这样做，不过用户知道她一直在来回循环吗？她还记得这个序列里面的第一个页面吗？这可不一定——因此，链接回父页面可能更友好，因为父页面可以告诉她她已经看过所有可以看到的页面了。

你可能也会想到这个模式在列表对象导航中的应用。一个单窗口深入（One-Window Drilldown）的应用可能有一个对象列表来通过导航进入（例如一个地址簿里的条目）。一旦你深入到了其中一个对象，可能再直接进到下一个对象会是很好的选择，而不必回到父页面再来一遍。从拓扑结构上来看，这个问题和网站上的链接是一模一样的，虽然应用的上下文完全不同¹。

1. 金字塔模式的这个方面在Larry Constantine和Lucy Lockwood的一篇论文中有很好的阐述。这篇论文的名字叫“Usage-Centered Patterns: A Collection of Abstract Design Patterns for Presentation and Interaction Design”（以使用为中心的模式——一组展示和交互设计的抽象设计模式），他们供列表用的金字塔模式版本叫做“Detail View Direct Navigation”（详细视图直接导航）。参见<http://foruse.com/patterns/detailnavigation.pdf>。



图3-19 The Museum of Modern Art (现代艺术博物馆) 的网站拥有一个好玩的摩天大楼脚本程序。它描述了世界上24个最高的建筑物（包括规划中的高楼）。用户可以从一个排序的建筑物列表中选择，深入到这个建筑物的详细描述，然后还可以回到那个建筑物列表，也可以单击“上一个”或“下一个”直接进入另一个建筑物的详细描述页面。

这个屏幕上可以看到一个根据高度排序的建筑物列表。它可以水平滚动，当鼠标滑过的时候，每个建筑物的轮廓会转变成整体外观图，就像“金茂”（Jin Mao）工具提示上显示的那样。

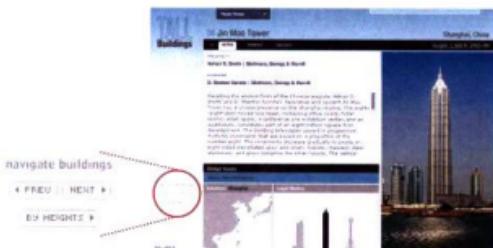


图3-20 如果你单击金茂大厦，就会来到这个页面。注意那些小小的“上一个”“下一个”和“按高度排序”的按钮，都排放在一个固定的位置，参见 <http://moma.org/exhibitions/2004/tallbuildings>。



模态面板 Modal Panel

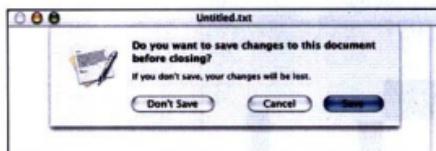


图3-21TextEdit的保存对话框

它是什么

只显示一个页面，在用户解决好当前的问题之前没有其他任何导航选择。

什么时候使用

应用处在一种没有用户协助就无法继续前进的状态里，例如在一个文档为中心的应用中没有已经给定的文件名。“保存”的动作需要用户提供文件名称才能继续保存。

为什么使用

模态对话框（或其他模态机制）排除了用户其他所有的导航选择。他不能扔下当前的对话框不管，不能到达应用的其他位置：必须立即处理当前的问题。当这个问题处理完毕以后，才可以回到之前的位置去做别的事。

这是一个很容易理解的模型——也容易实现——尽管这个模型在过去一些年里被滥用了。模态面板具有破坏性。如果用户还没有准备好回答这个模态面板要求的问题，它就打断了用户的工作流，可能会迫使他做一个当前还不太清楚或不太在意的决定。应用得当的时候，模态面板会把用户的注意力引导到下一个他确实需要的决策点上。因为没有其他的导航可能性来分散他的注意力了。

如何使用

在屏幕上用户当前注意力所在的地方放置一个请求，提供所需信息的面板、对话框或页面。它应该能阻止用户打开应用中其他的页面。要使这个面板非常整洁，保证用户的注意力集中在这个新任务上，而不会被吸引到别的东西上去。

记住这是一个和导航有关的模式。你要小心标记那些出口，而且出口也不能有很多，可以是一个、两个，也可能有三个。在绝大多数情况下，它们是一些简短标注的按钮，例如“保存”、“不保存”、或“取消”。通过单击这些按钮，用户应该能回到她来之前的页面上。

在这个问题上，Web应用的设计师们问题不大。不管设计师是不是想要去掉其他的导航选择，浏览器的后退和前进按钮（还有它的浏览历史机制、书签、URL输入框等）都会提供和往常一样的功能。这可能是一个有趣的地方，来观察这几年Web标准的演化。目前，我认为，除了图3-23中所示的一个来自Google的例子之外，对于Web应用没有什么好的模态面板解决方案。

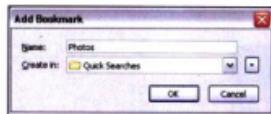


图3-22 这是一个典型的Windows应用模态对话框。它来自Firefox，所以它的外观看起来和大部分Windows应用有一点点不同，但是我们熟悉的模态对话框元素都在这里：OK按钮和Cancel按钮，有限的几个标题栏选择（例如关闭按钮），当前的内容也紧紧集中在一个任务上。不过，Firefox可以执行它的日常功能——作为一个Web浏览器——而不需要现在就回答这个问题。因此一个不使用模态对话框的解决方案可能更适用（虽然更难设计）。

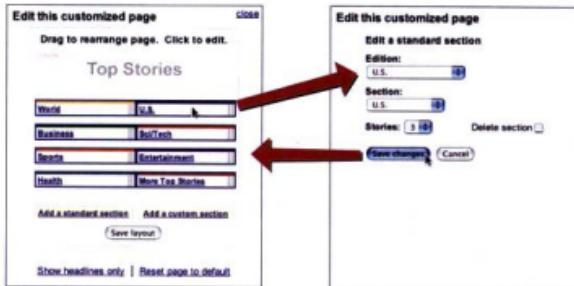


图3-23 这是Web应用里使用模态对话框的一个可能选择。Google新闻页面包括一个交互式的用户定制特性，就像Firefox那样。紧紧集中在一个小任务上。不过，它使用了Google新闻页面内部的一个小面板。当你用鼠标双击一个页面图标（例如“World”或“U.S.”）时，这个小面板就把它本身的内容改变为一个像对话框那样的表单。当你按下“Save changes”或“Cancel”（面板上这两个仅有的导航选择）时，它就会再次显示原来的内容。

本质上，它的行为看起来像单窗口深入模式在网页上的迷你应用。你仍然可以单击浏览器的前进或后退按钮，或Google新闻页面的其他链接，但是可以很清楚地从上下文中看到，那些部分都在当前这个定制迷你小应用的范围之外。

序列地图 Sequence Map

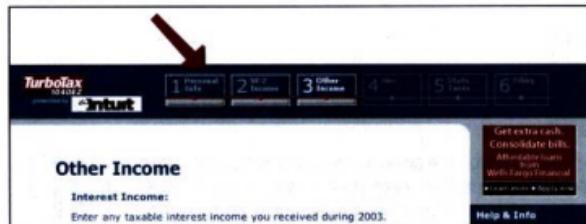


图3-24 TurboTax 2004版

它是什么

在一系列页面的每一页上，显示一幅地图，把所有的页面按顺序排序在该地图上，包括一个“你在这里”的位置指示器。

什么时候使用

你已经设计了一个用户可以一页接一页访问的文档、幻灯片、向导或其他什么东西。用户的访问路径主要是线性的。

如果导航的拓扑结构非常庞大，而且拥有层级关系（和线性结构相反），你可能向使用面包屑层级结构（**Breadcrumbs**）模式，因为你没有空间来显示序列地图。

为什么使用

一名电脑游戏玩家曾经在某个公共论坛上说过：“迷路并不好玩”。而这种游戏的部分吸引力就是要去经历一些陌生的地方完成任务！因此，如果本来就是严肃的应用，人们不想浪费时间找来找去，那么迷路就更不好玩了。

实际上，绝大部分时间里，在某个应用中“迷了路”并不是真正的问题——真正的问题是，你不知道到底还要走多远才能到达目的地。一个好的页面序列地图，显示在每个页面上，就可以在以下几个方面帮助用户：他可以看到已经完成的步骤（可能还会看到跳过的步骤或分支），他当前的位置（有一个“你在这里”的路标），接下来还有多少步骤要进行。

序列地图本身也应该作为一种导航设备，它应该允许用户通过在地图上单击跳过某些步骤。

如何使用

在页面的边缘，放置一幅小的序列化页面地图。如果可以的话，把它们放在一条线上，或者一个表格的一列上，在视觉上不要和页面的实际内容竞争。给当前页面指示器一些特别的对待，例如用浅一些或深一些的颜色来标记它，然后对已经访问过的页面进行另一种处理。

为了用户的方便，你可能想把这个地图放在靠近主导航控件的地方，或者主导航控件的后面，通常使用后退和下一个的按钮。

应该怎样标记地图上的每个页面呢？如果这些页面或步骤上有数字编号，那么明确使用这些数字编号——

它们简短而又容易理解。如果没有数字编号，那么把页面的标题放到地图上。如果标题很长，可能会让地图非常冗长拖沓！尽量缩短这些标题，或者如果实在做不到，那么可以试试把它们隐藏在工具提示或小的鼠标滑过窗口（rollover Window）上。

示例

After making the discovery and completing the requirements for his Ph.D., Dr. Crick plunged into the problems now made accessible by the new structure. How does the sequence of bases in DNA determine the sequence of amino acids in the ribbon-like structure of each protein molecule? How is the information copied from DNA and transferred to the cell's protein-synthesizing centers?

Continued

<Previous | 1 | 2 | 3 | 4 | 5 | Next >

图3-25 新闻网站常常把长的文章切分成多个页面。《纽约时报》的网站就在每个文章片断的右下角放置了一个序列地图。上一页和下一页的链接放在两端，非常方便，之前已经阅读的页面链接会变成和访问过的链接相同的颜色——它利用了用户已经知道（也可以由CSS控制）的链接行为。

注意，那些小的数字链接可能对某些用户来说太小了，不方便单击。所有《纽约时报》的读者都跟年轻而且身手敏捷吗？

面包屑层级结构

Breadcrumbs



图3-26 来自`http://java.sun.com`

它是什么

在层级结构的每个页面，显示所有父级页面的链接，向上追溯到主页为止。

什么时候使用

你的应用和网站完全是按树型结构组织的，在树的内部没有多少内部关联。用户通过直接导航或搜索在这棵树中上下移动。采用单窗口深入（One-Window Drilldown）架构的应用常常使用面包屑导航机制。事实上，我还从没见过多窗口桌面应用采用面包屑层级菜单的，尽管平铺的面板可能会用到它们。参见这个模式中iTunes的屏幕截图，来看看主面板上应用面包屑的例子。

面包屑层级结构和**序列地图**是两种可以互换的导航方案。你可能想采用序列地图来帮助用户一直保持方向感，但该序列地图可能规模太大，难以显示在一小小块屏幕空间上，所以需要另一种方案。

为什么使用

面包屑层级结构显示了到达当前页面的每一层链接，从应用的最顶端一直向下。从某种意义上说，它显示的是整个网站或应用的一个线性切片，从而避免了显示整个地图的复杂性。

因此，就像序列地图一样，面包屑层级结构帮助用户

得知他当前的位置。如果他是突然从外面跳到这棵树的某个节点上，那么这一点尤其方便，因为他也许是沿着搜索结果进来的（或者一个书签——如果是网站的话）。不过，与序列地图不一样，面包屑不会告诉他接下来要去哪里，也不会告诉他刚才是从哪里来的，它只关心现在。

你会从很多文档中得知，面包屑层级结构——这个名字来源于Hansel和Gretel的故事，在那个故事里，孩子在森林里撒下面包屑来记住他回家的路——对告诉用户他是如何从应用或网站的顶端到达当前位置来说，帮助最大。但这只在用户直接从顶往下深入阅历的时候才有用。如果用户转向别的路线，或沿着别的分支前进，或到达一个末端，或通过搜索，或从别的页面直接链过来……那就很难说了。

相反，面包屑在告知你当前的位置和网站或应用其他部分的相对关系上，是最有用的——它和上下文相关，而不是和历史相关。参见上面的图3-26。我在一些漫不经心的浏览之后来到Sun公司Java网站的这个页面，它告诉我，我到了“Products & Technologies”（产品和技术）这一级，属于“Reference”（参考）这个栏目。现在我知道，如果要了解别的产品或找到别的参考资料该怎么走了。因此面包屑导航机制展现了网站或应用信息架构的有用线索。

最后，面包屑通常是可以单击的链接或按钮（和序列地图一样）。这些链接或按钮把它变成了一种有自己风格的导航机制。

在页面的顶部，放置一行文本或图标来表示页面在当前层级结构中的位置。从最顶级开始，在最顶级的右边，放置下一级，然后一直往下直到当前页面。在这些层级之间，放一个图标或文本字符——通常是向右的箭头——来表示从一级往下一级移动的方向。

当前页面的标题应该是这个页面的指示器标志。如果已经访问过那些页面的话，用户应该能认出它们；如果没有，那么这些标题至少应该是自解释的，来告诉用户这些页面是关于什么的。如果标题太长，尽量缩短它们，或把它们隐藏到工具提示或一个小的提示窗口当中。

示例



图3-27 典型的iTunes音乐商店屏幕。我进行了一次关于某首歌的关键字搜索，然后得到了搜索结果页面（我很犹豫把它叫做页面，因为这不是一个传统意义上的网站，它是一个联网的桌面应用）。不管怎样，面包屑层级菜单告诉了我两个级别：首页（以一个图标显示），以及搜索结果。现在我单击我在“Top Albums”下看到的第一个专辑，然后……



图3-28 我得到了这个页面。你可以看到面包屑的路径有所变化。现在，它显示我在iTunes标准三级音乐结构中：流派（声乐）、艺术家（Etta James），还有专辑。这让我有了一个机会一键到达流派和艺术家的页面，而我可能对这两者都感兴趣。我也可以单击后退按钮——就在当前面包屑菜单位置的左边——回到搜索结构。这个页面再次显示，面包屑是关于上下文的，而不是访问历史。

因此面包屑层级菜单在这里做了两件事：在iTunes世界里建立了一种空间感，并为这个空间提供了丰富多样的导航选择。

注释滚动条 Annotated Scrollbar

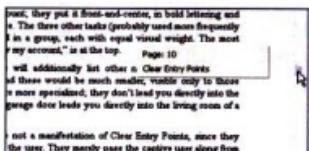


图3-29 微软Word的滚动条工具提示

它是什么

让滚动条在滚动的同时，还可以作为一种内容的映射机制。或者说，作为一个“你在这里”的位置指示器。

什么时候使用

你建立了一个文档为中心的应用，或任何其他包括一个大型滚动空间的系统，例如文档、列表或图像。用户会扫过这个应用来得到一些注释，例如具体的页码或位置标记，而且他们在滚动的时候也可能难以记住他们当前的位置，或者接下来去哪里。

为什么使用

尽管用户在内容中滚动的时候还呆在同一个UI空间里，但这时候的路标仍然很有用。当页面快速滚动时，很难阅读飞驰而过的文字内容（如果屏幕不能及时刷新的话，就不可能读到），因此有必要使用一些其他的位置指示器。甚至就算她暂时停下来，她能看到的文档页面可能也不包含任何能给她位置感的东西，例如标题。

为什么是在滚动条上？因为它是用户当前的注意力所在。如果你把路标放在滚动条上，用户将看到它们。

并把它们当滚动条使用，而不是勉强去同时查看屏幕的两个地方。你可以把路标紧靠滚动条放置，也可以得到同样的效果，越近效果越好。

当滚动条自己显示滚动踪迹的时候，你可以得到某种类似一维总览加细节（Overview Plus Detail）模式（参见第6章）实现的东西。滚动的窗口就是细节部分。

如何使用

把位置指示器放在滚动条上，或靠近滚动条的位置。不管动态的指示器还是静态的指示器都可以——静态指示器指的是不会随时间变化的指示器，例如在滚动条轨迹上的一个颜色块（参见图3-30tkdiff的屏幕截图）。不过，要让它们的意图清楚：这样的小东西可能会让不习惯在滚动条轨迹上看见图形的用户感到困惑！

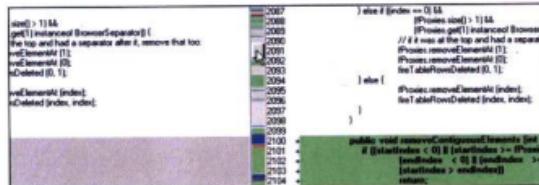
动态指示器在用户滚动的时候改变内容，它们通常以工具提示的形式来实现。当滚动位置发生变化的时候，滚动块旁边的工具提示显示和内容有关的信息。每个应用都有不同的显示内容，例如我们在上面的图3-29中看到的那样，Word把页码和标题放在这些工具提示里。

不管是动态指示器还是静态指示器，你都需要弄清楚用户最可能在寻找什么，从而找出你需要把什么放到注释里。内容结构是一个很好的起点，如果内容是代码，你可能需要显示当前功能或方法的名字；如果是一份电子数据表，你可能需要显示行号、等等。同样也要考虑用户当前是否在一个专门的模态里，例如查

找某个字符串——注释应该显示和它相关的内容，如果那就是用户正在文档中寻找的。

该模式的描述部分来自*About Face: The Essentials of Interaction Design*（“Wiley”出版）。

示例



The screenshot shows a window titled "tkdiff" comparing two versions of a C++ file. The left pane contains the original code, and the right pane contains the modified code. The code is color-coded to highlight differences: green for new additions, blue for changes, and red for deletions. A vertical scroll bar is visible on the right side of the comparison area.

```
size() > 1) &&
get() instanceof BrowserSeparator) {
the top and last separator after it, remove that too.
removeAt(1);
removeAt(0);
removeAt(0, 1);
removeAt(index);
removeAt(index, index);
```

```
) size() > 0) &&
(Promises.size() > 1) &&
(Promises.get() instanceof BrowserSeparator) {
// If there are still more than had a separate
Promises.removeElement(1);
Promises.removeElement(0);
fireTableRowsDeleted(0, 1);
}
else {
Promises.removeElement(index);
fireTableRowsDeleted(index, index);
}

public void removeComponent(LayoutManager layout) {
if (startIndex < 0 || endIndex > PANE_SIZE ||
startIndex < 0 || endIndex >=
(endIndex - startIndex)) {
return;
}
```

图3-30 注释滚动条并不常见。tkdiff是当前很少几个在滚动条中灵活显示这些非标准项目的应用之一。这个程序可视化地高亮了一个文本文件两个不同版本之间的区别，新增加的地方标记成了绿色，修改的地方是蓝色，删除的地方用红色显示。一个注释滚动条的作用相当于一份全局地图，让大文件更容易理解。



图3-31 来自<http://johncoltrane.com>

它是什么

使用颜色来标记一个应用或网站中某个网页所属的栏目。

什么时候使用

大型应用的UI，有很多页面或窗口，这些页面和窗口可以组织成一些栏目（或章节、部分、子应用等）。你可能会使用一个视觉框架来对它们在视觉上进行标准化，但你还希望每个部分有自己独特的外观。可能每个部分都需要单独的品牌，或每个都有不同目的或目标用户的话。

为什么使用

这是一个路标的例子——提供线索让用户知道他当前在哪里的某种机制。在这里有一些微妙之处，颜色是在视觉上而不是文字上起作用的。它甚至也不是某种用户能一下子捕捉得到的东西（不过在图3-31中，Vivid Coltrane的例子，你没法不注意到这些颜色）。一旦用户了解了某种配色方案，他们就会使用这些方

案。甚至在此之前，他们也会在离开一个栏目到另一个栏目时发现这一点，只要他们发现颜色发生了变化的话。

因此颜色编码的作用是把不同的栏目区分开来，它们让各个栏目边界分明。用户很容易从心智上辨识导航空间的各个部分。例如一个栏目，而不是整个空间——无论如何，你应该在设计大型UI时注意这一点，不管你是不是采用颜色编码。

创造性地使用不同的颜色也可以让你的界面更丰富多彩，不那么枯燥。这样甚至也会为UI的品牌做出贡献——参见图3-32中苹果公司的例子。

如何使用

为每个栏目选择不同的颜色。通常，如果改变背景颜色，那么变化太大了——Coltrane的方案只在可视化框架太强大并且与众不同的时候才好用。在绝大多数设计中，改变一种用来修饰的颜色，例如边框或一小片文本的背景颜色会更好。

要记住，色盲用户将看不到这些栏目颜色上的区别，因此颜色绝不应该成为唯一的栏目区分方式。你还需要

要有其他的路标——例如导航条上的指示器及页面或窗口的标题。

示例



图3-32 苹果的网站为颜色编码提供了一个典型的例子。看每个屏幕截图的顶部，为了和内容匹配，Tab页和它下面的条的颜色改变了（还有文本！）——例如，蓝色的是Quicktime，金属色的是OS X。在页面图形中，可以看到，这些主题颜色在标题表和LOGO那里得到了呼应。

它的效果不是特别明显，但仍然是可以觉察到的。它对可用性和品牌都有帮助，同时也不会破坏网站的整体一致性（要注意的是，Tab页是这个网站的全局导航机制，而二级导航链接就在那些颜色编码的条上）。

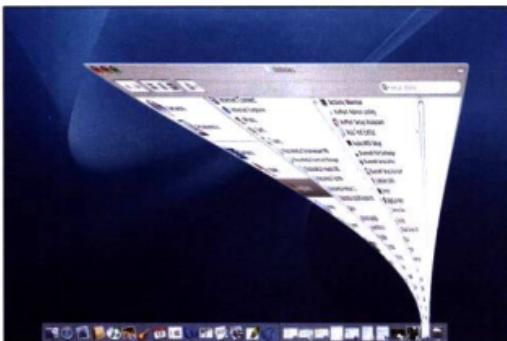


图3-33 Mac OS X的“精灵效果”，一个窗口正在最小化的动画

它是什么

把一个突然出现或位置移动的转换用动画来显示，让它变得更自然。

什么时候使用

用户在一个很大的虚拟空间里移动，例如一个大图像、电子表格、大文档。它们也许能缩放到不同的级别，可以平移或滚动，或者整个翻转。这些动作对信息图形来说特别有用，例如地图和图表（更多关于信息图形的内容见第6章）。

另一方面，这样的界面可能有一些部分可以关闭，然后再次打开，不管是通过系统还是用户——例如一棵有可关闭父节点的树，能够打开和关闭的单个窗口，或者一个由可关闭的面板（第4章）构成的界面。动画转换可能也用在用户从一个独立的页面向另一个独立页面跳转的过程中。

为什么使用

所有这些类型的转换都有可能破坏用户当前在虚拟空

间中的位置感。例如，完成得太快的放大或缩小会让用户失去这种感觉，翻转也是一样：一些区间的关闭会改变屏幕的布局，甚至沿着一个长的页面或文档滚动的时候，如果是跳跃前进的，也会让读者的认知反应慢下来。

但是如果从一个状态到另一个状态是连续转换的，情况就会好一些。换句话说，你可以把这种状态之间的变换用动画来变得平滑和连贯。这样可以帮助用户保持方向感。我们猜测它之所以有用是因为它更接近于我们这个物理的现实世界——上次你从地面直接跳到20英尺高是什么时候？动画化的转换能让用户的眼睛有机会跟踪到这些视图变化时的位置，而不是费力去再次寻找变化之后的位置。

如果做得好，动画转换将给你的应用带来“酷的感觉”。它们很好玩。

如何使用

为你界面上的每种转换设计一个简短的动画，让它把转换的开始状态和结束状态连接起来。对缩放和翻转

来说，你可能会显示缩放中或翻转中的状态。对一个正在关闭的面板，你可以显示它正在缩小而其他面板正在扩展到它原来的空间的过程。如果可能，尽量让它看起来像某种正在发生的物理变化。

如果说其他模式都有利有弊，那么这个模式也不例外。小心会让用户觉得发晕！动画应该快速而准确，在用户开始动作的时候，动画就应该马上开始，最好不要有延迟。动画只限于屏幕上受到影响的部分，不要让整个窗口都动起来。而且动画时间要短。我更喜欢少于1秒的动画，研究表明，300微妙的动画在平滑滚动的时候最理想¹。做些用户测试看看大家的容忍时间大约是多少。

如果用户连续操作，引发了多个动作，例如多次按下箭头按钮来滚动窗口，把它们结合到一次动画中去。否则，用户可能不得不坐在那看好几秒钟的动画作为他10次按键的惩罚。再强调一次：动画要快，反应要敏捷。

示例

这个叫做“The Secret Lives of Numbers（数字的秘密生活）”的Web应用能让用户在相当大的数据范围内探

索——它显示了可缩放的数字柱状图。从0到10万。它使用了动画转换来把用户从高层次的总览视图传递到非常小，非常详细的具体视图。

图3-34显示了当你在某个数字上按住鼠标按钮（这里是2000）所发生的情形：柱状图慢慢地缩小，并在它缩小的时候对图上每个东西都进行了平滑的动画转换。

几点要注意的提示：

- 缩小的时候柱状图的竖条变宽了，这样保持了尺寸和比例的感觉。它们看起来就像真正的对象那样。当它们变宽的时候，你知道它们正在缩小。
- 左边空白处的数字标签慢慢地分开，在白色标签之间出现了颜色很浅的小标签。白色标签之间的间隔也在变化，从25到10再到5，同时它们之间的像素距离相当稳定（这样在查找数字的时候很重要。你不会希望一个有意思的柱状条离它的标签太远）。
- 在水平刻度压缩的同时，竖直的隔线也变窄了。当那些小隔线彼此靠得太近的时候，它们淡入了黑色的背景当中，相应的隔线标签也不见了。因而整个图形保持了整洁。

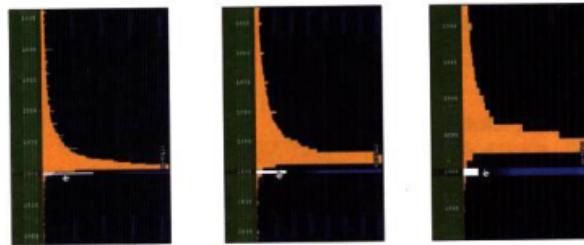


图3-34 在“The Secret Lives of Numbers”上进行放大。参见<http://www.turbulence.org/Works/nums/applet.html>

1. 参见“Benefits of Animated Scrolling”一文，网址为<http://hcil.cs.umd.edu/trs/2004-14/2004-14.html>。

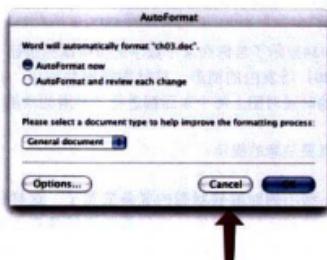


图3-35 一个危险的对话框，来自Word，不过因为Cancel按钮的存在，它没那么危险了

它是什么

在每个限制了导航选择的页面上，放置一个按钮或链接，让用户能明白无误地离开这个页面，回到熟悉的地方。

什么时候使用

你已经有了一些页面，它们包含某些连续的过程，例如向导，或某些页面把用户锁定在一个导航受限的情形下，就像中心和辐条模式或模态面板模式那样。用户可能会突然闯入那些页面，例如通过搜索结果。

（有时候，如果页面上已经有了序列地图或面包屑层级结构，就没必要再有逃生舱的存在了。知道那两种导航机制的用户可以利用它们回到熟悉的地方。）

为什么使用

受到限制的导航是一方面，而没有出去的道路是另一

方面。如果你为用户提供一种简单的、明显的方式逃离一个页面，没有任何附带条件，那么他也不会有那种上当受骗的感觉。

这就是那种可以帮助人们觉得他们可以安全撤离一个应用或网站的功能。有点像撤销功能——它鼓励人们继续向下探索，而不必觉得把自己完全托付给了网站。参见第1章的安全探索模式。

如果用户可以通过搜索结果到达这些页面，例如一个步骤过程中的某个页面，那么在每个页面上放置逃生舱尤其重要。访问者可以单击它们来到一个更常见的页面，告诉他们自己实际上在什么地方。

如何使用

把一个按钮或链接放在页面上，用它把用户带回安全区域。例如，你可以使用首页、中心和辐条模式中的中心页，或者任何有着完全导航和自我解释的页面。具体链接到哪里取决于具体应用的设计。

网页常常使用可以单击的站点图标作为回到首页的链接，通常显示在页面的左上角。这些页面在某个熟悉的位置提供了逃生舱功能，同时有助于加强网站的品牌。

在某些对话框中，取消按钮或其他起相同作用的按钮可以用于逃生。他们也让用户表达这样的意思：“我不想继续了，忘了我前面所做的一切吧！”

你有没有曾经打电话给一个公司——例如你的银行——然后不得不在一堆电话菜单中找到出口？这些菜单可能很长，含义不清，而且耗费时间。如果发现自己在一个错误的菜单下，你可能只想挂断然后重新拨打。不过，很多电话菜单系统都有一个隐藏的逃生舱，它们没有告诉你：如果你在任何地方拨0，可能会转接到人工服务。

04

组织页面：页面元素的布局

**ORGANIZING THE PAGE:
LAYOUT OF PAGE ELEMENTS**

页面布局是一种艺术，它通过操纵用户在页面上的注意力来完成对含义、顺序和交互点的传达。

如果“操纵”（manipulating）这个词让你觉得不太舒服，那么这样来想想看。电影和电视导演谋生的方式就是操纵你对银屏或电视屏幕的注意力，在这个过程中，你多半还是一个自动的参与者呢。为报纸组织文章、头条新闻和广告的编辑也差不多。如果这些内容是灰土土的，没有什么图形重点来抓住并转移你的注意力，你可能会发现这样才难以明白其中的含义呢——到底哪些重要，哪些不重要？

就算这终归是一种艺术，好的页面布局也应该比你想像的要更为理性一点。本章的介绍里会先提到一些重要的图形设计理念，每个理念都可以在页面、屏幕和对话框布局上为你提供指导。我们会讲到视觉层次结构、视觉流及注意力焦点、分组和对齐——这些都是页面设计中一些可预测的、合理的方法。本章的模式讲述了一些具体的方法，来把这些高层的概念应用到界面设计上。

但是计算机显示在本质上是灵活可变的和交互式的，它们让页面布局在某些方面变得更容易了，而另一些方面则更难了。我们也会讲到造成这种现象的原因。其中一些模式在适用于屏幕设计的同时，也适用于平面媒体设计，但其中绝大多数都对平面媒体没有什么用，因为它们假定用户会和页面进行交互。

页面布局基础知识 THE BASICS OF PAGE LAYOUT

这一部分讨论页面布局的五个主要方面：视觉层次、视觉流、分组和对齐，怎样把这三个元素综合到一起，以及怎样使用动态显示技术。

视觉层次：重要程度区分 VISUAL HIERARCHY: WHAT'S IMPORTANT

视觉层次的概念在各种形式的图形设计中都有一席之地。简单地说，最重要的内容应该最突出，而最不重要的内容则应该最不突出。标题应该看起来像标题，二级内容应该看起来像二级内容——换句话说，读者应该能从页面的布局推导出它的信息结构。

这个概念用例子来解释最清楚了。在图4-1显示的文本中，完全没有任何格式来表示视觉层次。

You're invited to Zelda's 30th Birthday Party! Please come dressed as your favorite Gilbert and Sullivan character. Children are welcome. Dinner will be served; if you'd like to bring food, call Stacy at 555-1212. When: October 20th, at 7:00 PM Where: Zelda's house. If you need directions, feel free to call Zelda and ask. Please RSVP to Stacy by October 10th. See you there!

图4-1 没有视觉层次

还过得去，但算不上很好。这个段落里哪些内容是最重要的？你可以猜测第一句话就是最重要的，但是反过来，也很难讲，因为整段文本都没什么变化。一旦你读完它，意识到这是一份邀请，就可以从上下文中得出重点——但是你得先读一遍。

现在让我们来做点改进。空白是组织视觉层次时最好的工具之一。它不需要什么代价，而且也是一种优美的方式，把这些没有变化的文本分隔开来。

You're invited to
Zelda's 30th Birthday Party!

Please come dressed as your favorite Gilbert and Sullivan character. Children are welcome. Dinner will be served; if you'd like to bring food, call Stacy at 555-1212.

When: October 20th, at 7:00 PM
Where: Zelda's house. If you need directions, feel free to call Zelda and ask.

Please RSVP to Stacy by October 10th. See you there!

图4-2 带空白的文本

在图4-2中，至少你可以看到几组不同的信息。而且顶部的大标题——“Zelda's 30th Birthday Party!”——比其他部分更突出，因为它周围留有空白。底下不是很重要的RSVP消息也是这样。但是你的眼睛最先扫到的文字可能是“You're invited to”，它出现在左上角。这是一个所有有从左到右语言习惯的人都会先注意到的地方。这一点就给了它不容质疑的重要性。

在页面排列上也会有排版和位置的问题，图4-3将会演示这些问题。

You're invited to

Zelda's 30th Birthday Party!

Please come dressed as your favorite Gilbert and Sullivan character. Children are welcome. Dinner will be served; if you'd like to bring food, call Stacy at 555-1212.

When: October 20th, at 7:00 PM

Where: Zelda's house

If you need directions, feel free to call Zelda and ask.

Please RSVP to Stacy by **October 10th**. See you there!

图4-3 加上了排版和对齐方式

当然，大的加粗字体表示了它的重要性。我们的眼睛被吸引到厚重而又对比强烈的图形，它们“视觉重量”很大。这份邀请最重要的一行用了一种超大号字体。第二重要的行用了一种大号字体，但比上面的要小一点，正文部分用的是正常字体。同样，指示和注释部分小字号的字体表示：“你可能想要阅读这些内容，但是如果没看到，也没多大关系”。

空间位置的作用在这里稍微复杂一点。这里再次用空白把其中一些文字隔离开来了。它也增强了“*When*”和“*Where*”在外观上的重要性——这很重要，是通过把它们放在几乎空白的左边，和大标题对齐来达到的。

一些页面元素的外形也能给你一些线索。在这个例子当中，关于位置指示的注释在“*Where*”的下面缩进了一点点。可以猜到这个注释和上面的文字有关，但是没那么重要。同样的逻辑也适用于树型视图、链接下面的辅助文本、文字输入框下面的输入提示等。通过这些及其他熟悉的结构（例如表格），它们的视觉外观甚至在用户开始阅读内容之前，就立即表露了它们的含义。

在本章中，中央舞台（**Center Stage**）模式通过鼓励你建立一大片为主要任务服务的UI区域来直接面对视觉层次。同样，使用带标题的栏目（**Tiled Sections**）模式也有助于定义视觉层次。还有，如果你开发了一份视觉框架（**Visual Framework**，这是另一个模式，它在整个UI范围内处理视觉层次的问题），要确定它适合你所需要的不同视觉层次，例如标题、大字标题、副标题、列表、导航条、动作按钮等。

下面这些方面可以帮助你处理视觉层次。

- 左上角优先
- 空白
- 字体对比：字体越大、越粗，就表示这部分内容越重要
- 前景颜色和背景颜色的对比：例如把白色的文字放黑色的背景上，会得到一份在白色的页面上特别强调的声明
- 位置、对齐、缩进：缩进的文字从属于任何它上面的内容
- 图形（例如线条、方框、颜色条）：在一个方框或一个分组中的东西属于一个整体

你马上就会发现，很多界面和印刷图形使用了其中一些机制。网页经常使用颜色和字体来区分标题与正文。很多界面同时使用分组框和空白来建立视觉上的分组。这样很好。有了这些可以选择的“变量”，你就多了很多设计上的自由，特别是它们每个都有双重角色：显示界面的组织方式，传达品牌、情感和其他非理性的特征（在第9章我会再回来讲述这些有意思的主题）。

现在，让我们再来深入了解一些视觉组织方式。

视觉流：接下来我该看到什么

VISUAL FLOW: WHAT SHOULD I LOOK AT NEXT

视觉流的作用是在读者扫描页面的时候跟踪读者的视线。当然，它和视觉层次密切相关——设计良好的视觉层次应该在页面上建立了许多焦点，就是那些把注意力吸引到最重要元素的地方，而视觉流将把视线从这些焦点引导到那些重要程度次之的信息上。作为设计师，你应该可以控制页面上的视觉流，让人们能够按照正确的次序沿着它向前流动。

当你建立视觉流的时候，有几个因素可能会互相牵制。其中一个是我们从上到下、从左到右的阅读习惯，当面对一份没有什么格式的文本时。那就是我们自然而然的做法，但是页面上任意一个焦点都会把我们的注意力从这种阅读过程引开，这一点有时候会带来好处，也有时候会带来坏处。

焦点（focal points）就是页面上那些你的眼睛没法抵制的地方。一般会根据从最重要到一般的次序去查看它们。设计良好的页面上只会有几个焦点——焦点太多会互相竞争从而降低了彼此的重要性。你可以通过很多方式来建立它们，例如使用空白、高度对比、大而粗的字体、吸引人的颜色、汇聚的线条、轮廓鲜明的边线、表情和动作（是的，这个列表就像一个上面那样的视觉层次，标题、LOGO、重要的文本或图片栏目使用这些属性来成为焦点）。

下次你拿起一本杂志的时候，看看其中一些精心设计的广告，然后看看你的视线被引到什么地方。最好的商业图形艺术家都是些大师，他们擅长建立焦点来控制你最开始看到什么。

不过，如果你曾经研究过一个充满广告的网页，并且很快就跳过了那些色彩明亮闪来闪去的广告（这样你才能阅读那些想要阅读的大段格式简单的正文），你就知道，我们并不只是这些硬邦邦的可视化系统的奴隶！我们可以选择跳过我们认为没必要看的部分，把注意力集中在页面内我们认为最重要的那一部分内容上。因此，这也意味着上下文在视觉流里也担任着很重要的角色。

如果你建立了一个界面，而这个界面会受到某种前后顺序影响的话——像向导那样，或者一个对话框，它开始时的选择会影响它之后的选择——那么多考虑视觉流（甚至就算你的界面没有什么前后次序的关系，你也应该考虑它，因为一份设计良好的视觉流在用户看来会更容易）。要建立一份和视觉流相符合的页面布局并不难，但是要小心违反它的那些布局方式。把控件和按钮按照一种直接的视觉路径摆放。在这个路径的最后，放上用来结束这个任务的链接或按钮（“提交”、“回到主页”或“确定”），然后把用户带到别的地方去。

图4-4显示了一个对话框，它的视觉流设计相当不错。留意一下，你的视线在页面顶端开始，直接往下移动到文本栏（可能在横线那里会减速一下），并且在那四个图标处着陆。在这里停了一下，看看它们之后，可能你的视线会跳到Help或OK或Cancel按钮。

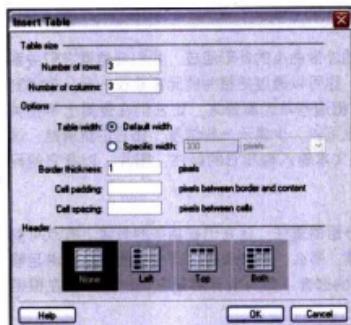


图4-4 Macromedia的插入表格对话框

现在，或许你会有时间来看看这些标签，但是它们的字体比较小，不太可能会让你第一眼就看到。那些文本输入框可能是你第一眼看到的，因为它们是一些焦点——它们是白色的（和灰色相对），刻意进行了对齐，而且在语义上很重要。

总的来说：

- 从上到下，从左到右是默认的视觉流方向
- 强烈的焦点会首先吸引眼球，然后才是那些次之的
- 对页面内容的理解会随着用户选择看哪一部分而改变

分组和对齐：谁跟谁一起

GROUPING AND ALIGNMENT: WHAT GOES WITH WHAT

你早就知道它们了，不过我还是要在这里说一遍，因为在视觉上把元素分组非常重要，分组意味着它们之间互相关联。反过来说，如果两个元素在位置上相隔很远，例如一个表单的最后一个输入字段和这个表单的提交按钮隔得很远，那么这意味着它们之间没有关系。如果视觉流不是按步骤进行，不是通过其他元素把用户的视线引导到这个按钮的话，用户可能就看不到按钮了。

和焦点一样，分组和对齐对于形成一份清晰的视觉层次结构是非常必要的。通过把用户的视线从一组元素引导到另一组，它们也有助于形成视觉流。

人类的视觉系统期望看到先后顺序。如果看到大一些的外形由小的外形组成，我们会觉得它们关系紧密，例如一些字母组成单词，一些小网格组成表格。你可以通过把相关的元素放在一起，来得到这种对顺序的需要带来的好处，然后通过足够的空白把这些单元隔开来，让它们在逻辑上不相关（又是空白！可以使用空白，哪怕是在很小的空间里也可以。少显示一些信息，让它们更清楚，这比把一个页面撑得满满登登要好）。这样你就可以把文本输入框和它的标注、图片，以及它的标题、图表和控制它的滑块联系起来了。

可以使用分组框，但也别期望它们能担任起整个视觉分组的重任，让它们在拥挤的页面上有点可以“喘息的空间”。如果你用斜视的方法看不到分组框边缘，那么，用来获得这些分组的空白留得足够吗？如果够了，那就好。还有，小心两级和更深层次的嵌套，因为要弄清楚它们的关系实在很困难。试试用别的方法来表达这种视觉层次。

对齐是另一种更巧妙的页面元素关联方式。例如，如果你有两组按钮，它们彼此之间在页面上相隔很远，但是它们都被用来完成差不多的任务，此时可以把一组放在另一组的下面，并让它们宽度相同来突出这种相似性。或者你在页面上有两个表单，有一些文字把它们分开了，就可以让这两个表单左对齐，而让文本沿着另一根竖线（不可见的）对齐。

分组和对齐背后的理论早在20世纪已经由格式塔（Gestalt）心理学家们建立起来了。他们描述了几条布局属性，可以用在我们的视觉系统之中。下面就是这些属性。

相邻性

如果把物体相邻摆放在一起，用户会把它们互相关联起来。这就是要对界面上的内容和空间进行分组的基本原因。

相似性

如果两个元素有着相同的大小、形状、颜色或方向，用户也会把它们相互关联起来。

连续性

我们的眼睛想要看到由对齐或更小元素组成的连续线条和曲线。

封闭性

我们也希望也看到简单封闭的区域，例如矩形和大块空白（没有明显画出来的那些区域）。对界面元素进行分组看上去常常组成了封闭的区域。

图4-5描述每个属性，以及如何把它们结合起来创建有效整体设计。

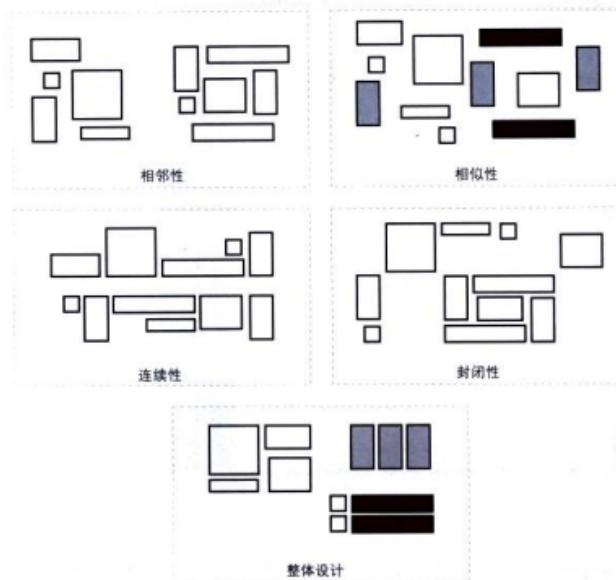


图4-5 四种格式塔原则

虽然每一个原则都很重要，但其实把它们结合起来使用效果更好。我们又一次看到，冗余并不是不好。第五个分组看起来更像一个真正的页面而不是某种怀旧风格的马赛克。

于是，连续性和封闭性解释了对齐的作用。当我们把页面元素对齐的时候，在它们的边缘组成了一条连续的线，用户可以沿着这条线（可能是下意识地）假定它们之间的关系。如果对齐的元素非常一致，组成了某个形状——或者周围都是空白，或者周围都是实体——那么封闭性就开始起作用了，加强了整个效果。

把它们整合起来

PUTTING IT ALL TOGETHER

在图4-6所示的网页上，我把文本变模糊了，因此很难阅读它们。但是我敢打赌你能看明白大体的页面结构。

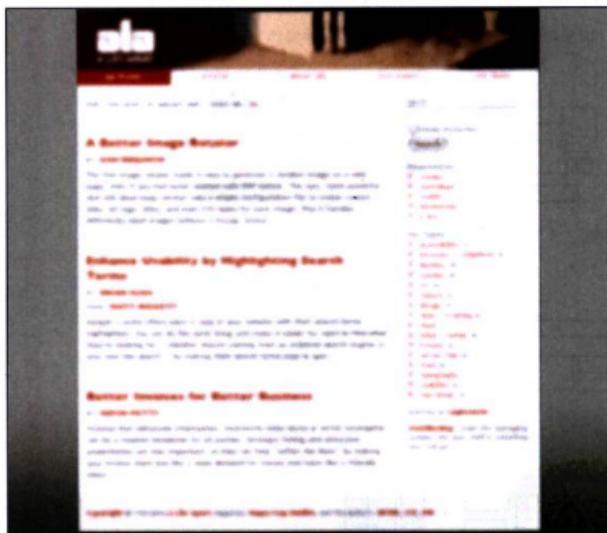


图4-6 来自 <http://www.alistapart.com>

你第一眼就会感知到我们刚才讲到的很多内容。让我们来分解一下第一眼的印象。首先，你的视线是如何移动的？可能是图4-7上的一条路线，或者它的一些变体情形。

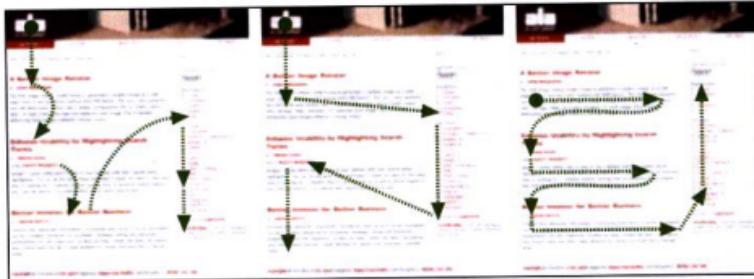


图4-7 在alistapart.com上可能的视觉流

站点图标（LOGO）显然是一个焦点——它处在网页的页首部分，有高度黑白对照的大字体。标题行用它们的颜色、字体大小和周围的空白来吸引注意力。右边的栏目看起来是橘黄色的，内容很多，而且有趣，所以你也许想看看它。

要注意，第三个视觉流并不是从站点图标开始的。一个只对页面内容感兴趣的访问者可能会简单地跳过页首部分，在左边白色部分的上面开始阅读——因为看那些装饰部分没什么认知上的好处。

现在，我们刚刚讲过的格式塔原则怎样了呢？这个页面展现了三个大的分组。每一组都有大的红色字体，小一些的红色字体，正文部分。相似性和邻近性组成了这些分组；而空白把它们分开了。右边的竖栏用一个较宽的边框分开了——实际上是两个，注意它左边的子边栏——它的密度和颜色让这个部分在视觉上非常一致。页首部分的黑色背景组成了另一个视觉区域。灰色的页脚部分和它唯一的文本行是另一个视觉区域。图4-8显示了这个页面的分组情况。



图4-8 ALA页面上的分组

所有这些格式都对视觉层次有帮助。当你运用你对网站已有的知识，开始分解看到的内容时，上下文处理和高级的心智处理就启动了。

你可以很容易把页首部分、页脚部分、主要内容和辅助信息区别开来。在主要内容区域里，有三个兴趣点——可能是链接的文章——每篇文章都有它的标题、副标题（署名）及正文部分。在辅助信息中，有一个小小的表单（搜索），两个表格（可能包含链接，因为它们不是黑色的），还有一些其他的文本。页首部分包括五块文字，它们互相对齐而且看起来很相似，可能是顶级导航。就像在大部分网页上一样，你可以照常忽略页脚部分。

使用动态显示

USING DYNAMIC DISPLAYS

上面我说到的所有内容都可以应用到界面、网站、海报、广告牌还有杂志页面上。它们针对的是页面布局的静态部分。啊，不过你还要看看动态的计算机显示——然后突然间，时间变成了另一个设计维度！重要的是，计算机允许用户和页面布局在某种印刷媒体没法达到的程度上进行交互。

有很多很多方式让你可以利用计算机显示的动态本质。例如，可以集中使用空间——最大的个人计算机屏幕可以使用的屏幕空间比海报或报纸都要小。这是事实。如果你在设计PDA或手机，那么能使用的屏幕空间就更小了。当然，有很多技术可以使这样的小屏幕空间展现很多内容，这些内容可能不是一次就能全部看到的。

你的意思是，更小的空间？

没错。你的21英寸显示器物理上比一本杂志翻开来要大。一个宽银幕的显示空间比一张报纸更大。真正的问题在于分辨率。绝大多数界面包括很多文字。而且就在写这本书的时候，大部分屏幕没法像印刷媒体那样把文字弄成很小的字体而且保持可读性。因此，你没法把屏幕上的那些栏目尺寸（例如）和同样大小的纸质媒体对应起来。非常抱歉！

滚动条是一种常用的展示某个大元素的一部分内容的方式，例如文本、图片、表格等。滚动条让你随意移动，可以在一个方向上滚动，也可以是两个方向（但是不要在文本显示时使用水平滚动条，拜托）。

或者，你可以把内容分成几个连贯的部分，然后有几种模式可供选择——卡片堆、可关闭的面板、可移动的面板——不像静态的带标题的栏目（**Titled Sections**），它们把所有的布局控件都放到了用户的手里（你还可以把内容分成一些虚拟的分页，让用户在这些分页上前后移动，参见第3章“导航”）。它们根据用户的选择，在不同时间让他们看到不同的内容。

如果你想通过一些有顺序的步骤来引导用户，那么响应式允许和响应式展开是两个有名的方法，可以做到这一点。

模式

THE PATTERNS

本章的模式讲述了一些具体的方式来把这些页面布局概念应用起来。

前两个模式，视觉框架和中央舞台，讲的是整个页面、屏幕、窗口的视觉层次问题，而不管你在这个页面上放置什么内容。你应该在项目的早期考虑视觉框架，因为它会影响到界面中所有的主要页面和窗口。



视觉框架
Visual Framework



中央舞台
Center Stage

接下来的一组模式表述了四种把大块内容放到页面或窗口上的方法。如果你一次要在页面上放置大量内容，它们将非常有用，一旦你决定要把内容划分成几个主题部分时，就需要选择如何表现它们。它们应该一次都全部能看到，还是（或者应该是）分开来看？可以让用户在页面上操作这几个部分吗？这些模式和视觉层次有关，但它们也涉及了交互性，还会对你从UI工具箱里选择具体的适用机制有所帮助。



带标题的栏目
Titled Sections



可关闭的面板
Closable Panels



卡片堆
Card Stack



可移动的面板
Movable Panels

右对齐/左对齐和对角平衡模式对视觉流、对齐和前面的介绍中讲到的各种概念进行了具体化。它们处理的是小的静态元素之间的空间关系，例如页面上的文本和控件。



右对齐/左对齐
Right/Left Alignment



对角平衡
Diagonal Balance

属性表模式有点不一样。它也讲到了页面上小元素之间的空间关系，不过它在讲述布局的同时，也讲述了内容和交互。把这个模式放在这里是因为，当一个熟练的用户想起这个页面上有一个属性表的时候，他们的期望就非常明确了。它的布局可以精确地告诉用户他们应该如何与页面交互。



属性表
Property Sheet

最后三个模式讲述的是内容布局的动态方面。响应式展开和响应式允许是两种指引用户通过一系列步骤或一组选择的方式，他们提示了在任何时候用户可以做的事，同时也阻止用户不小心走到可能会引起麻烦的地方。流式布局是一种可以按照用户的爱好改变大小和形状的页面组织技术。



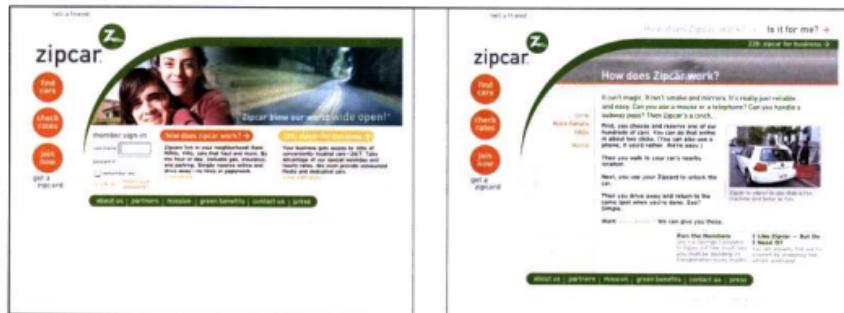
响应式展开
Responsive Disclosure



流式布局
Liquid Layout



响应式允许
Responsive Enabling

图4-9 来自`http://zipcar.com`

它是什么

使用相同的基本布局、颜色、格式方案来设计页面，但是会让设计足够灵活来处理不同的页面内容。

什么时候使用

你已经建立了一个有着很多页面的网站，或者有着很多窗口的UI。换句话说，几乎任何复杂的软件都可以应用这个模式。你想把这个软件整合在一起，看起来是一个整体，经过设计的整体：它也应该容易使用，容易导航。

为什么使用

当一个用户界面使用了一致的颜色、字体和布局，当标题和辅助导航的元素——路标——每次都出现在同样的地方时，用户就会知道他们自己当前在哪里，也会知道下一步要去哪里寻找。他们不需要每次在页面或窗口上下文切换的时候去适应一种新的布局方式。

你看到过哪本书，它的页面和标题在每一页的位置都不同吗？

一个强有力的视觉框架，在每个页面都会重复出现，这样的话，它可以让页面的内容更突出。用户会下意识地忽略那些不变的部分，而注意到那些变化的部分。还有，若为视觉框架的设计增加一些关于网站品牌或产品的特征，页面将更容易识别。

如何使用

设计的时候，为每个页面或窗口建立一种整体的外观和感觉。首页和主窗口比较特别，通常会采用一种与其他页面稍微不同的布局方式，但是它们仍然应该和其他页面在某些特征上保持一致。例如：

- 颜色：背景颜色、文本颜色、着重颜色
- 字体：标题、副标题、普通正文和不太重要的文字

- 书写风格和语法：标题、命名、内容、简要描述、任何大段的文字、任何使用了语言的元素

同样，其他所有的页面或窗口都应该在下列方面保持一致。

- “你在这里”的路标：标题、站点标志、面包屑层级路径、卡片堆索引（例如Tab页或链接栏）
- 导航设备：一组标准链接、确定/取消按钮、后退按钮、“终止”（quit）或“退出”（exit）按钮、导航模式，例如全局导航、序列地图、面包屑层级结构（这三个模式均参见第3章）
- 用来定义带标题的栏目（**Titled Sections**）的技术
- 间隔和对齐：页面边距、行间距、标签和关联控件之间的距离、文字和标签的排版
- 整体布局网格：页面元素的位置、按行或按列摆放

放，还有前面提到的边距和间隔问题

如果对图形设计概念很熟悉，你可能会发现这些技术涉及布局网格（layout grid）。布局网格是一组页面或一组布局的结构模板。每个页面都不同，但是所有页面都会使用特定的页边距，都会把它们的内容跟某些看不见的格线对齐。一份好的视觉框架的确会包含一个布局网格，但它也包括外观和感觉的其他方面，例如颜色、视觉细节、写作风格。

对视觉框架的实现会强迫你把UI的格式部分和它的内容剥离开来。这并不是坏事。如果你只在一个地方定义这个框架——例如在一个CSS样式表里或一个Java类里——这样你可以完全独立地改变这个框架，而不必考虑内容。这意味着你可以更容易改变它，也更容易得到更好的设计（这也是一项良好的软件工程实践）。

示例

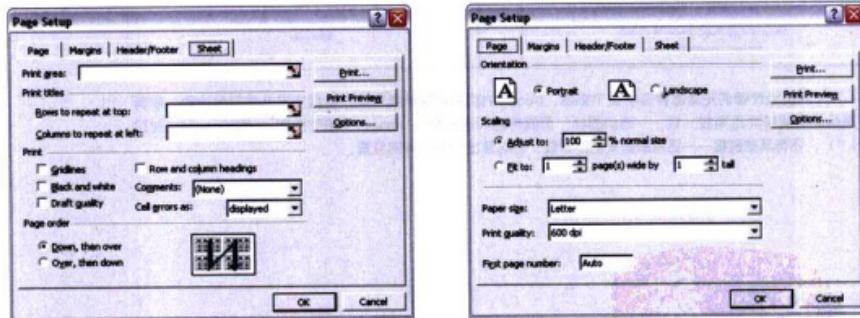


图4-10 Windows和Mac操作系统的外观和感觉（look-and-feel）有助于实现一个视觉框架，因为它们的颜色、字体和控件都相当标准。但是你还需要增加一些更高层的结构，例如布局网格和语言。这些Excel的屏幕截图都来自同一个对话框——页面设置——它们也都很好地诠释了视觉框架的概念。在页面之间，所有这些特征都是一致的：动作按钮放在上面和右下角；边距大小、缩进、文本输入框之间的竖直距离；文本输入框到右边距的延长；使用标签作为分隔（例如“打印标题”和“打印方向”）来界定栏目，还有大小写和语法。



图4-11 网页设计们知道怎样应用这个概念。Google的页面很简单很干净，但是非常非常容易识别。所有的路标都很清楚而且高度一致——站点图标、页面标题和副标题（“Images（图形搜索）”和“Groups（讨论组）”），还有其他链接——还有最重要的搜索框，它总是出现在同样的位置。

中央舞台 Center Stage

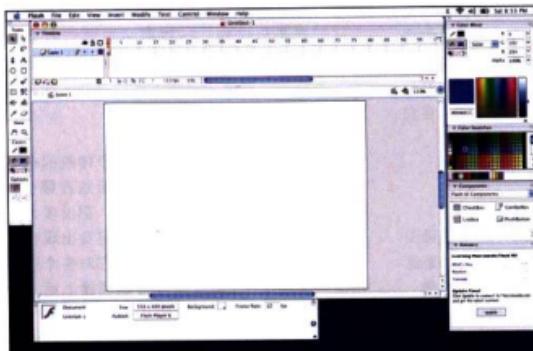


图4-12 Flash MX

它是什么

把最重要的UI部分放到页面或窗口最大的子栏目上，把一些辅助工具和内容放置在它周围的小面板上。

什么时候使用

该页面的主要工作是为用户显示连续的信息，让他编辑一份文档，让他可以执行一项特定的任务。大多数可以使用中央舞台模式的应用——表格和数据表、表单、网页、图形编辑器——都符合这一条件。

为什么使用

你应该把用户视线迅速引导到最重要的信息（或任务）的起始处，而不是让他们一头雾水地在页面上寻找。让一个清晰的中央元素锁定他们的注意力。就像新闻报道的前几句话就确定了文章的主题和目的一样，中央舞台¹中的元素建立了这份界面的目标。

1. 这个模式的名字第一次提出的时候是在1998年，是在P.r. Warren和M. Viljoen的论文“Design Patterns for User Interfaces”（用户界面设计模式）中提出的。

中央舞台

一旦有了中央舞台，用户将会根据它们与中央内容的关系来评估外围的元素。和不断在页面扫描的方式相比，这样会更容易——首先是什么？然后是什么？这两者之间如何关联？等等。

如何使用

建立一份视觉层次结构，让中央舞台来控制其他所有元素（参见本章首介绍部分对视觉层次的讨论）。在设计一个中央舞台的时候，要考虑以下因素，虽然它们都不是绝对必需的。

大小

中央舞台部分至少应该有它左右边距的两倍宽，上下边距的两倍高（用户可能会改变它的大小，但这应该是用户首次看到它时它的默认大小）。

颜色

使用一种可以和周围元素形成对比的颜色。在桌面应用的UI里，相对于窗口的灰色，用白色就很合适，特别是用在表格和树上。碰巧白色也通常用在网页上，因为广告和导航条常常会使用其他颜色作为背景；还有，Web用户已经习惯了在白色的背景上寻找正文。

大标题

大的标题都是焦点。它们可以把用户的视线吸引到中央舞台的顶部。当然，在印刷媒体上也是这样。如果想了解更多关于大标题和焦点的信息，参见本章章首介绍部分和带标题的栏目模式。

上下文

当用户打开页面的时候她想看到什么？一个图形编辑器？一篇很长的文章？一份地图？一个关于文件系统的树型结构？按照这些期望，把它放

到中央舞台上，让它容易识别。用户会在界面上寻找——这是所有关于视觉感知的秘密武器。

（但是并不意味着你可以通过把她想找的东西藏起来来阻挠她！一些网站把它们的主要内容放在页面下端太远的地方，因此它在当前窗口中被盖住了，需要用户往下滚动寻找。简直是在折磨人。）

你可能注意到了，我没有提到一个传统的布局变量：位置。事实上，不管你把中央舞台放在哪个位置——上面、下面、左边、右边、中间，都没多大关系。如果它够大，那么最终或多或少一定会出现在中间。要记住，各种已有的流派已经有了它对各个边距的习惯用法——例如，工具条在图形编辑器上面，导航条在网页左边²。要有创意，也要了解这些习惯。

如果你有些疑虑，那么拿一张布局的屏幕截图，让它变小，变模糊，然后问别人他认为页面上哪里应该是主要内容开始的地方。还有，可以回头参见前面本章的介绍部分。

2. 译注：现在更常见的是导航条在网页的上面。



图4-13 看看这个来自 <http://boingboing.net> 的屏幕截图。它的广告和其他周边内容确实会吸引一部分注意力，不过中央栏目，Blog文章的主体部分相对来说很宽。它也是从屏幕的顶部开始的，页面头部很简短，视线很容易就会被吸引到第一篇文章上。

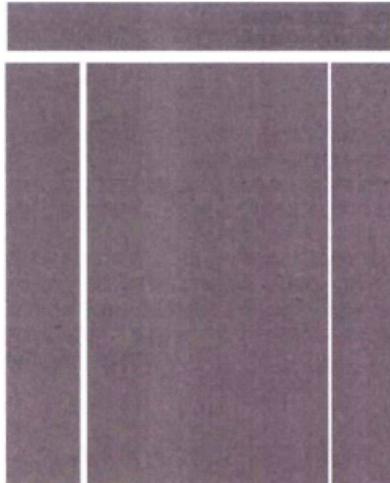


图4-14 要好好领会这一点，就看看主要页面栏目被抽象成简单矩形的时候发生了什么。它可能在截图上看起来不是很宽，不过中间的区域有左边广告区域的三倍宽。需要这么宽，才能和广告那些闪烁的颜色竞争，并形成强烈的对比。



图4-15 Google地图有着非常干净整洁的界面，由于地图的大小和颜色特点，在这里很容易应用中央舞台模式：地图占据了窗口的大部分面积，而且色彩丰富，和上面及右边白色背景的工具形成了鲜明的对照。

当然不能忘记的是——你必须保证，你的产品环境是干净、无干扰的。我们不建议在你的产品中添加任何广告，或者在你的产品中添加任何无关的元素。如果必须添加，那么请确保它们是必要的，并且不会干扰到用户。

图4-16 <http://adobe.com>

它是什么

通过给每个栏目一个醒目的标题来区分不同栏目的内容，然后把它们都排列在页面上。

什么时候使用

页面上有很多很多内容，但你希望让页面变得更容易扫描，容易理解。而且，你可以把这些内容分组成一些对用户有意义的主题栏目或任务栏目。

为什么使用

设计得好并且命名得当的栏目可以把页面内容组织成容易消化的几块，每一块都是在一眼之间就能理解的。它让信息架构（Information Architecture）更明显了。（参见本章首介绍部分关于视觉层次的讨论，该讨论基本上就是关于如何把内容以某种方式表现出来，而这种表现方式要能表达它的实际结构。另外，在第2章中有对信息架构的定义。）

当用户看到一个页面像这样分成了几个整洁的部分时，她的视线可以在这个页面上更舒服地流动。人的视觉系统常常在寻找更大的模式，不管它们是有意还是无意形成的。因此要仔细考虑它们的组织方式。

如何使用

首先，把信息架构组织好——把内容分割成几个大块，给它们一些简短好记的名字（如果可能的话，最好是两三个字）。其次，选择一种表现形式。

- 对标题来说，使用一种突出的字体和其他内容区别开来——加粗、加宽、加大字号、强烈的颜色等（记住，黑色是最强烈的，红色也比不上它）。
- 试用对比的颜色反白标题。黑底白字会让它看起来像Windows的标题风格。
- 使用空白来分离各个栏目。
- 为栏目设置不同的背景颜色在网页上表现很好。在那些想要看起来像网页的界面也不错，不过这种形式在桌面应用界面上很少见。
- 有着刻蚀/浮雕效果的方框在桌面应用界面上很常见。但是，如果方框太大，方框之间彼此太过靠近，或出现深度嵌套的话，它们很容易混失在界面上——只是一些视觉噪音。如果加上标题，会好很多。参见后面的示例。

如果页面还是难以辨认，试试卡片堆模式、双面板选择器模式或者需要时显示模式来管理。也可以把这些模式和带标题的栏目模式结合起来。本章的后面，可关闭的面板模式和可移动的面板模式是表现分块内容的另外两种方式。

如果你在给这些大块内容起标题的时候觉得困难，那也许这意味著分组并不适合这些内容。试试把它们组织成其他一些容易起名字也容易识别的分块。“杂项”(Miscellaneous)类可能也意味著这不是一种十分合适的组织方式，虽然有时候它们确实很有存在必要。

元代

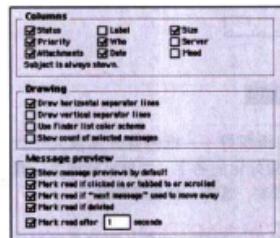


图4-17 桌面应用中带标题栏目模式的典型使用。在Eudora的个人偏好对话框中，这些Checkbox周围的方框看起来不错。深黑的标题很突出。在这些栏目之间也有着足够的空白区域，来给它们一些视觉上“喘息的空间”（实际上，如果去掉这些方框也可以）。

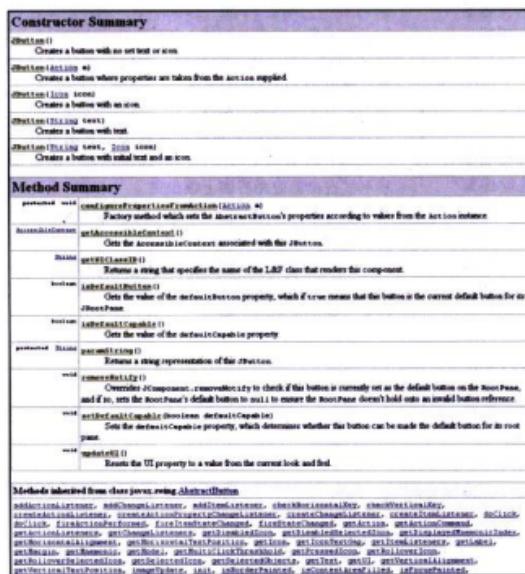


图4-18 这个屏幕截图来自一个很长的Java文档页面。每个栏目都用蓝色的长条来标记，当用户在页面上快速滚动的时候，这些蓝色长条很容易找到，也容易阅读。注意看这里采用的视觉层次：主栏目用更大更深的颜色进行标记，而底下最不重要的栏目使用了小一些的字体，颜色也浅一些。在这个层次上的第二个级别，类和方法的名字比较醒目，因为它们是绿色的，而且向左凸出；最后，正文采用了黑色的小字体，就像常见的正文那样。

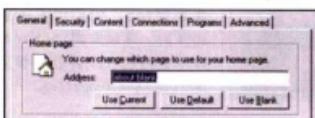


图4-19 IE的属性对话框

它是什么

把不同栏目的内容组织成几个单独的面板或“卡片”(Cards)，并把它们垒成一堆，一次只显示一个栏目：用户可以使用Tab页或其他设备来访问它们。

什么时候使用

页面上的内容太多了。一大堆控件或文本分布在界面上，又不能使用一种很严格的结构（例如属性表）来组织：用户的注意力分散在这些控件或文本上。你可以把内容分组成带标题的栏目，但它们可能太大了，不能一次全显示在页面上。最后——也是最重要的——用户每次不需要看到超过一个的栏目。

为什么使用

这种带标签的卡片把内容组织成容易消化的几块，每一块都变得在一眼之间可以理解了。同样，用户对Tab页这种最常用的卡片堆形式很熟悉。

如何使用

首先，把信息架构组织好。把内容分割成几个大块，给它们一些简短好记的名字作为标题（如果可能的话，最好是两三个字）要记住，如果你把这些内容分块的时候分错了，当用户输入信息或比较信息的时候，就必须在这些卡片之间来回切换。对你的用户温柔一点，并且对你的组织方式进行测试。

其次，选择一种表现形式：

- Tab页非常不错。但它们通常只支持6个或更少的卡片。不要把Tab弄成两行。因为两行的Tab页通常都很难用；如果一行实在显示不下，那么就进行水平滚动。
- 竖直的Tab页会迫使你把卡片堆塞到一个又高又窄的区域里，通常不适合用在正常的Tab页上。
- 在大多数网页和对话框中，左侧竖栏这种方式不错。这样你可以把一大堆卡片放到一个竖向的栏里。它也可以允许你把卡片组织成层级结构，而使用Tab无法做到这一点（在某种意义上，它更像一个双面板选择器。在这两者之间确实没有明显的界限）。
- 有些用户界面在页面的顶部使用了下拉列表框，它占用的空间比侧面的链接区域还要小，但是要在清楚程度上付出代价：下拉列表框的行为常常像控件，用户不会把它当做一个导航元素。如果内容非常明显的话，可以使用它，参见图4-23中的OS X例子。但用户不能一眼就看到所有的卡片标题。

如果你想找到一种卡片堆的替代方式，来让人们一次看到两个或更多的卡片，那么看看可关闭的面板模式。它们没有Tab那么强大的隐喻力量，但对于想要学习的用户来说非常有效。



图4-20 你可以用很多很多种方式来表现Tab，它们也不一定要从左边开始。这是来自Theban地图项目Web应用的一个界面，参见<http://thebanmappingproject.org>。

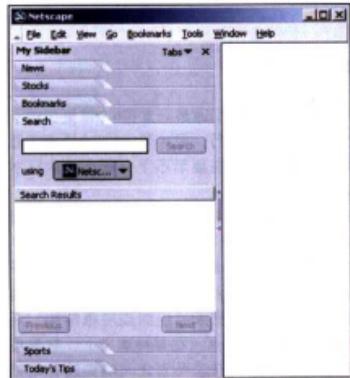


图4-21 Netscape利用了不同的物理限制。这个侧边栏上的Tab按钮竖直排成了一堆，当用户单击的时候它们还会上下移动，所以现在选中的页面总是把它自己的Tab按钮直接放在上面。对于有限的、竖向空间的Tab页来说，这是一个有趣的解决方案（这种形式最初出现在Visio的早期版本中）。

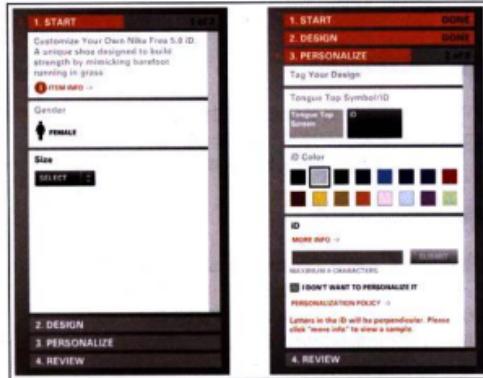


图4-22 这是竖向卡片堆的另一种实现方式，它甚至没有假装使用Tab。Nike的Web应用让你单击橙色的水平条，一次显示一个面板。

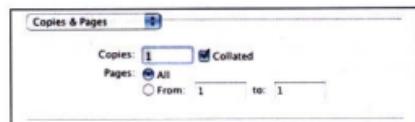


图4-23 有时候一些应用会在应该使用Tab页的地方使用下拉列表框。这是OS X IE浏览器打印对话框的“Copies & Pages”（打印份数和页数）的页面。下拉框非常节约空间，也可以允许又长又多的页面名称，但是如果不下打开下拉列表框，用户就看不到其他可选的页面。请注意，这些控件上面和下面各有一条分隔线。用户需要这种包含关系来理解下拉框究竟是做什么的。否则，它看起来就像一个普通的控件，而不是一个导航设备。下拉框没有Tab那么容易使用。

可关闭的面板 Closable Panels

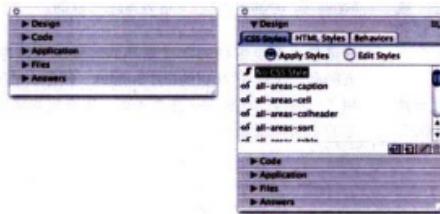


图4-24 Dreamweaver MX

它是什么

把不同栏目的内容组织成几个面板，让用户每次可以打开和关闭单独的面板而不影响其他面板的显示。

什么时候使用

页面上要展示的内容太多了，但你仍然希望只要一次单击就能看到它们。和带标题的栏目及卡片堆一样，我们把内容分成了命名清楚的一些栏目，用户可能想一次看见两个或更多的栏目。

尤其，这是需要时显示模式的一种很有用的具体实现方式。如果已经用需要时显示模式组织了你的内容，你可能想使用某种可关闭的面板来容纳这些额外的内容。

为什么使用

在很多可以使用卡片堆模式的地方都可以使用可关闭的面板模式——同样，卡片堆也可以节约空间，也需要用户单击才能看到内容。不过，可关闭的面板模式让你具有更多的灵活性。

- 它可以包含一些大小完全不同的栏目，如果使用卡片堆来组织大小不同的栏目，小栏目可能会在页面上留出大块空白，那样会很不优雅。

- 用户可以一次打开几个栏目。这一点可能对写作工具及支持各种复杂应用的“高级用户”方面特别有用——根据用户的判断力，他们可以同时打开很多面板，并让它们一直显示在页面上。
- 如果你只有一个可关闭的面板，而且把它放在一个大的页面或对话框里，那么这个面板看起来会没有它的容器重要。如果你把这两个栏目放到两个Tab页里，那么它们看起来就同样重要了。

使用这些可关闭面板的时候也有一些代价。绝大部分人都很容易理解Tab页，但可关闭的面板却不是那么广为接受——它们不会一开始就让每个人都觉得熟悉。最好做一些可用性测试。还有，可以一次打开很多栏目的后果可能会让整个环境变得很杂乱，从而导致不可用。

如何使用

首先，把内容分割成不同的栏目，给它们一些合适的名字，把它们放到面板上。当你单击一个按钮或链接的时候，显示或隐藏它们。在那个按钮或链接上放一个箭头、加号、三角形或双箭头符号（“>>”）——这么做可以暗示用户这个地方有个东西可以打开或关闭。在很多应用里，箭头方向向下表示可以打开的面板已经打开了，向右表示它关闭了；有时候它们也叫做“旋转向下（twist-down）”。

在很多应用里，当你使用这种元素的时候，包含可关闭面板的窗口会随着面板的打开或关闭而变长或缩短。你可以（或特意）把它放在一个滚动面板上，或者把它放到一个大小可变的长方块里，就像这个模式中Dreamweaver那个例子一样。

还有，要注意这里用到了可关闭面板上嵌套的卡片堆。如果你实在有太多太多栏目要处置，而且如果它

们可以放到一个两层的层次结构中的话（例如，“Design”栏目包括三个子栏目，叫做“CSS Styles”、“HTML Styles”和“Behaviors”），这么做是可以的。

这项技术至少在1993年或更早的时候就已经存在了。当时有一种基于Motif的GUI Builder，叫做UIM/x，就使用了可关闭的面板——甚至还有可变方向的箭头——在它的小部件面板上。

示例

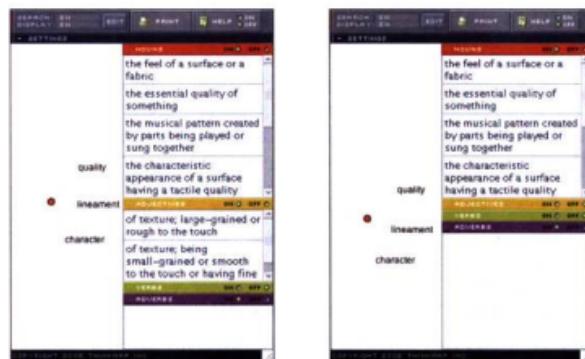


图4-25 Thinkmap的Visual Thesaurus是一个使用了可关闭的面板模式的Web应用。彩色的标题条描绘了四个可以随意打开或关闭的面板，通过on/off的单选按钮来控制。你也可以拖动这些标题条来对栏目重定义大小。用户对他界面上看到的内容有了更多的控制权——如果四个栏目太乱的话，他可以关闭最不相关的那些，参见<http://visualthesaurus.com>。

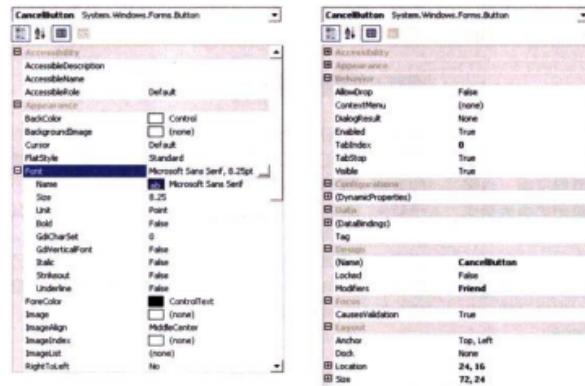


图4-26 Visual Studio的属性表窗口使用了可关闭的面板模式来按照标题对它那些数量众多的属性分类。例如，对访问权限或外观属性不感兴趣用户可以把这两类属性收起来，将注意力集中在其他属性上。Visual Studio在这里没有使用箭头，而是使用了方框里面的加号，就像Windows的树型视图控件一样。尽管这并不是一个树型视图，但还是重复利用了用户已经知道的知识：你可以通过单击加号来打开某个对象，单击减号来关闭它。



图4-27 这是一个网页上应用可关闭面板模式的例子。Flickr为每张照片显示了小巧精致的可关闭面板，用来显示相关照片或系列中的其他照片。你可以单击用“+”或“-”标记的按钮来打开或关闭它们。

This photo also belongs to:

- Murals in San Francisco (387)
- + You are the last photo
- 15 photos
- in this set

+ Murals and Mosaics (Post)

Search: Lord of the rings

Advanced Search

Web Results [Web] Showing 1 - 10 of about 3,430,000 hits

Lord of the Rings
LOTR DVDs, props, toys and more Free Shipping! Official LOTR Store!
www.lotrstore.com

Lord of the Rings shop
adults, teens, children, adults, vips kids & adults, come to collectable
www.hollywoodcostumesparty.com

Latest news for lord of the rings:
■ A chance to win: 'The Lord of the Rings' books
■ See Mercury News (subscription) - 17 hours ago
■ Lord of the Rings star embraces Buddhism + more: 'I'm not a Christian,' says Peter Jackson
■ Lord of the Rings - everyone's favorite read - Sydney Morning Herald (subscription) - Dec 5, 2004

Official Lord of the Rings Return of the King Tickets New Movies ...
the official website for the return of the king tickets. OFFICIAL LOTR New Movies Say... enters instant movie the lord of the rings.
(link) (www.lordoftherings.net/) - 9k
Cannes (link)

Lord of the Rings, Movies, Return of the

Book Results [Books] Showing 1 - 10 of about 103,387 hits

The Lord of the Rings (50th Anniversary Edition)
By J. R. R. Tolkien (21 October, 2004)

J.R.R. Tolkien Boxed Set (The Hobbit and The Lord of the Rings)
By J.R.R. Tolkien (12 September, 1998)

The Lord of the Rings (Leatherette Collector's Edition)
By J. R. R. Tolkien (01 November, 1974)

The Lord of the Rings
By J. R. R. Tolkien and Alan Lee (November, 2002)

The Lord of the Rings Complete Visual Companion
By Jude Fisher (15 November, 2004)

The Lord of the Rings Trilogy (Lord of the Rings)
(September, 2004)

The Fellowship of the Ring (The Lord of the Rings, Book I)
By J. R. R. Tolkien and Ken Sughrue (01 October, 2001)

IMDb.com Movie Results [Movie] [All]

Lord of the Rings: The Two Towers (1999)
The Lord of the Rings: The Fellowship of the Ring (2001)
The Lord of the Rings: The Return of the King (2003)
The Lord of the Rings: The Two Towers (2002)
The Lord of the Rings (1977)

Content provided by Internet Movie Database Inc. Where appropriate, links to detailed content at IMDb.com are provided.

Search Results [Search] [All]

[0] Web
[0] Books
[0] Images
[0] Movies
[0] Reference
[0] History
[0] Bookmarks
[0] Diary

More | Edit this

图4-28 这是来自Amazon的多维搜索工具A9的一个屏幕截图，它显示了九个面板中的三个。每个面板都是一个大小可变的纵向列——它们之间的间隔条是可以拖拽的，所以用户可以在需要的时候把面板加宽或收窄。右边的按钮列是一个排序控制中心，让用户来显示或隐藏不同的面板；不过对于面板本身的最大化和关闭，则在每个面板的标题栏右边有两个像按钮一样的链接，分别是“[Close]”和“[Full]”。因为已经有了多选按钮列，这两个链接有点冗余，不过也没关系。

注意一下，A9面板的标题并不是特别突出，这个页面上有一个视觉层次，但是它并不那么明显——如果标题再加强一些，也许可以更好地定义这三个面板。粗体、标题下面的水平刻度，或者彩色的背景都可以做到这一点。（不过考虑一下这个方面：例如，若这些标题都采用了强对比的颜色，它们可能会看起来像实际的标题栏。用户会不会试着抓住这些标题栏并移动它们呢？如果那样的话，会不会带来什么坏处？）

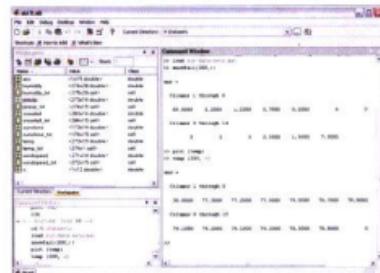


图 4-29 Matlab

它是什么

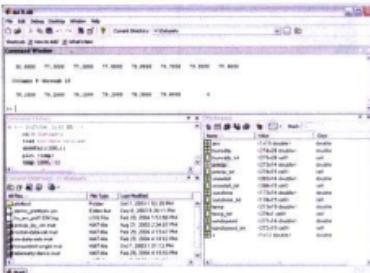
把页面上的工具或栏目组织到几个不同的面板上，让用户可以移动它们，形成自定义的布局。

什么时候使用

页面上有几个互相关联的界面区域，不过它们并不需要放置在一起；而且不管它们放在哪里，其含义对用户来说都很明确。你可能希望用户觉得他自己对这个软件有一些控制权，或者至少在使用的时候有一些乐趣。

为什么使用

如果某个界面已经用了一段时间了，人们希望重新布置他们的环境来适应自己的工作风格。他们可能会把需要的工具挪到当前工作位置的旁边，把那些不需要的隐藏起来，并且使用空间记忆模式（第1章）来记住自己把东西放在哪里了。理性地说，可移动的面板模式能帮助用户更有效率更舒服地完成工作（这是长期影响——一旦他们花时间把界面按自己喜欢的方式重新组织以后）。



但是这种个性化似乎在其他一些方面也很吸引用户。例如，他们可能对不常访问的娱乐网站也进行个性化设置。

如何使用

让用户按自己的喜好在页面上移动UI区块。为用户下次接着使用软件保存界面布局，如果这是他日常生活中很重要的一部分，这样做就特别有必要。

根据你选择的设计，你可能想要给用户一些自由，让他们把那些界面区块放在任何地方，重叠也没有关系。或者你可能需要一个预先定义好的布局网格，用网格来容纳这些区块——这也让页面能保持对齐（在某种意义上来说，也比较优雅）！也要注意，不要让用户在布置窗口上花太多时间。

如果可移动的面板对鼠标单击或鼠标拖动有响应（例如文本选择），那么可以考虑在每个用户可以抓住并移动的区块上放置一个“句柄”（handle，把手）。标题条就很不错。实际上，如果你在这个句柄上放置一个

“X”，有些用户将认为这是完全去掉该区块的方式（如果实现了这个，当然，得提供一种方式来重新创建这个已经消失的区块）。

你的用户可能希望界面上什么地方有个“恢复默认布局”的动作，因为有时候他们把布局弄乱了，而又只想从

头来过。当你对可移动面板进行可用性测试的时候，要特别注意那些意外的面板移动——如果一位用户不小心把一个面板拖到一个奇怪的地方，他可能一下子不知道发生什么事了，或者不知道该怎么撤销。

示例

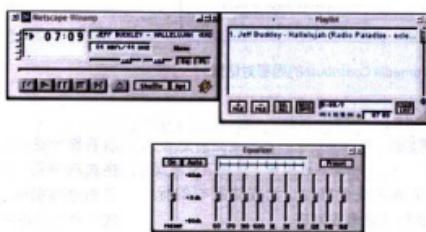
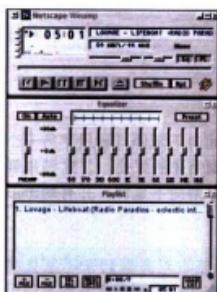


图4-31 在这里，我们分解并重新组织了Winamp的界面

图4-30 组成Winamp的这三个部分可以隐藏、显示、重新摆放位置，甚至彼此完全分开，各自独立移动。用户能通过拖动标题条来拖动它们，而在标题条上同样也有最小化和隐藏按钮。你可以通过弹出菜单项来重新显示隐藏的部分。在这个截图中，你看到的是Winamp的默认设置。

右对齐/左对齐 Right/Left Alignment

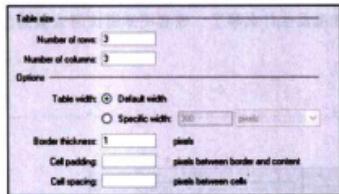


图4-32 Macromedia Contribute的局部对话框

它是什么

在设计一个两竖栏的表单或表格时，让左边竖栏的标签右对齐，而右边竖栏的元素左对齐。

什么时候使用

你在设计一个表单的布局，或者任何其他某组带文本标签元素的布局，这对表格的内部结构或任何其他两栏结构中那些行必须从左到右阅读的情况也完全适用。

为什么使用

当你把文本靠近某个它所标签的界面元素旁边时，这就组成了一个在认知上非常强烈的分组配对——比它们各自分别放在独立的地方时强烈很多。如果让长度不等的文本左对齐，那么那些短标签跟它们相关控件的距离就太远了，这种分组配对的感觉也就没有了（这是格式塔原则中的相邻原则）。简单地说，如果人们看到右对齐/左对齐模式出现的时候，他们更容易把标签和它关联的控件联系起来。

同时，你应该坚持对控件左对齐的原则。如果把左对齐的控件和右对齐的标签及每组标签到控件之间的空白结合起来，就会形成一种良好的双重边界，该边界

沿着整个这一组对象的中间向下延伸（利用了另一个格式塔原则——连续性原则带来的好处）。这种加强了的边界引导读者的眼光平滑向下延伸，并有助于形成一个良好的视觉流。

在一些界面布局中，对标签右对齐看起来不怎么美观。标签之间可能长度差别非常大，在标签的左边可能还有一列图标，或者也许左对齐的标题分隔了表单的栏目——所有这些格式都会破坏一个右对齐/左对齐的布局（虽然它们并非总是很有必要）。采用某种可行的方式。

如何使用

不是对每个文本标签左对齐，而是对它右对齐。把它摆放到相关控件的附近，只用几个像素来隔开它们。可能会产生一种左边参差不齐的效果——通常没事的。如果有些标签太长了，影响了右对齐的效果，可以试试把它们分成几行，或者把标签放到控件上面，那样的话就和这个模式没有什么关系了。

于是，左对齐的控件边缘看起来像是挂靠在一条虚拟的直线上，这条直线离标签的右边缘只有几个像素的距离。让它们精确对齐，对齐到像素级别——否则，这些控件看起来就会很杂乱。（人类视觉系统对细小的对齐误差很敏感！）

同样，控件列的右边缘可能是参差不齐的。如果是文本框、组合框以及其他一些视觉上很重的对象，这样的效果就不大理想，就像图4-32那样。可以试着拉伸它

们让它们的右边缘也对齐，可以拉伸到任何你可以拉伸到的程度。你也可以试着让短的控件互相对齐，而长一些的控件同样互相对齐。

示例

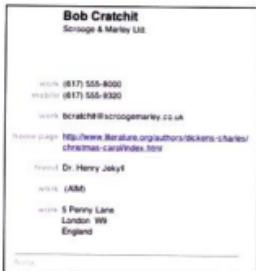


图4-33 这个模式也可以适用于完全没有输入控件的布局。这个Mac OS X地址簿条目的两个竖栏之间空白非常少，但不同的颜色有助于在视觉上把它们区别开来。注意那个“Home Page”的标签比其他标签长出很多，这也许会导致这组标签的左边不那么赏心悦目，也更难阅读。

Method Summary

protected void	<code>configurePropertiesFromAction(Action a)</code>	Factory method which sets the AbstractButton's properties according to values from the Action instance.
final AccessibleContext	<code>getAccessibleContext()</code>	Gets the AccessibleContext associated with this JButton.
final String	<code>getUIClassID()</code>	Returns a string that specifies the name of the L&F class that renders this component.
boolean	<code>isDefaultButton()</code>	Gets the value of the defaultButton property, which if true means that this button is the current default button for its JRootPane.
boolean	<code>isDefaultCapable()</code>	Gets the value of the defaultCapable property.
protected String	<code> paramString()</code>	Returns a string representation of this JButton.
void	<code>removeNotify()</code>	Overrides JComponent.removeNotify to check if this button is currently set as the default button on the RootPane, and if so, sets the RootPane's default button to null to ensure the RootPane doesn't hold onto an invalid button reference.
void	<code>setDefaultCapable(boolean defaultCapable)</code>	Sets the defaultCapable property, which determines whether this button can be made the default button for its root pane.
void	<code>updateUI()</code>	Resets the UI property to a value from the current look and feel.

图4-34 再次，长度参差不齐的标签意味着左对齐的标签会让标签离它们所标明的对象很远，很远。右对齐的话会更容易阅读。

对角平衡 Diagonal Balance

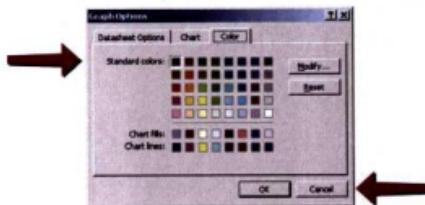


图4-35 Word的图形选项对话框

它是什么

用一种不对称的方式布置页面，但是，通过把视觉重量放在左上角和右下角来使页面保持平衡。

什么时候使用

你在对一个页面或对话框进行页面布置，它的顶上有一个标题或头部，而下面有一些链接或动作按钮——例如确定和取消、提交、后退、前进等。这个页面很短，只需要一个屏幕长度而不需要滚动。

为什么使用

视觉上突出的元素例如标题、Tab，以及按钮，它们都应该有助于让界面成为一个平衡的界面元素组合。它们已经在页面的两端了，当你把它们放在这两个不同的位置时，它们也可以做到彼此平衡（把它们想像成重量——控件越大或对比越强烈，控件就越重，它们越接近页面的边缘，你就越需要加强另一侧的平衡）。

除了页面更美观之外，这种沿着对角线的平衡也有助于让页面准备好。用户的视线可以很容易地从左上角移动到右下角——对有从左到右语言习惯的人来说，这是完美的视觉流（见本章介绍部分对视觉流的讨

论）。页面的其他部分也应该对这种视觉流提供支持。视线最终会来到表示用户可能最后才做的活动上，例如关闭界面或去别的地方。

如何使用

把标题、Tab或其他视觉上更重的元素放在页面的左上角，把按钮等放在右边下面的位置。把任何宽度的内容插在中间。如果内容自己有助于保持页面平衡，那么更好——例如，不要把太多空白放在其中某一边。

考虑一下，如果把上面颜色对话框中的OK和Cancel按钮放在左边而不是右边会怎么样。整个对话框的重心将向左倾斜，非常不稳定。

在Windows里，把标题放在左上角，并习惯性地把按钮放在右下角，这样就自动为你处理了对角平衡的问题。在Mac OS X里面，很多元素例如标题条、Tab、动作按钮等都在中间——因此对角平衡在这里很少见到。

在Web时代之前，Kevin Mullet和Darrell Sano在他们的可视化界面设计经典书籍*Designing Visual Interfaces* (Sun Microsystems出版) 里是这样描述对角平衡的：“对称的布局提供……自动的视觉平衡。不对称的布局一样

也可以做到平衡，但是这样的平衡更紧张、更动态，也取决于仔细的控制来在视觉上补偿主要界面元素的

大小、位置和颜色等。”参见下面的例子来看看如何达到这样的平衡。

示例

Usability vs. A City's Soul
Adam Gopnik | Design
Adam Gopnik rants against a new signage system in TOO MUCH INFORMATION
“Worse than merely unfamiliar, though, the signs are infuriating -- first, because they are there for the convenience of cars, and thus violate the first Law of Civilization, which states that nothing must ever be done for the convenience of cars (the mark of a city is that it is there for people). The second reason is that the signs are ugly, because they eclips, as Oscar, the jaunty, jazz-era synecdoche of the classic New York street-corner sign pair, each sign gesturing toward its own street, but with the two set at an angle, so that they have a happy, semaphoric panache.”
The city's commissioner of transportation argues for the signs by talking usability, but I think Gopnik's rebuttal is sound in both art and aesthetic planning. It's a fine reminder that a system is more than its parts, more than a single homogeneous solution that fits all. It must embrace the soul of a place and the nature of its people.
Posted at 07:26 AM, February 14, 2005
permalink | 2 Comments

图4-36 对角平衡模式也同样适用于纯文本界面。在这个Blog条目页面，视线首先到左上角，阅读内容，然后来到下面右边的链接。这些链接就是用户想对这篇文章做的另外一些动作（该Blog的地址是：<http://eleganthack.com>）。



图4-37 图形很多的用户界面也一样。在这里，我们可以看到这个网站的焦点是站点图标，正在行驶的车，“Let's Motor”的口号，右下角的代理商地址文本框——都在一条对角线上（大体上）。和别的例子相比，这个屏幕把视线往下往右推动得更明显。无疑，网站的设计者想鼓励人们使用那个文本框。如果反过来把它在左下角，页面可能会失去它的冲击力，那个文本框也不会被人注意到了。参见<http://miniusa.com>。

属性表

Property Sheet



图4-38 Visual Studio的属性表

它是什么

使用两栏或表单格式的布局来告诉用户，一个对象的属性可以在这个页面进行编辑。

什么时候使用

该界面为用户展现了一个可以编辑的对象——可能是图形编辑器内部的元素、一条数据库记录或查询记录，或者什么特定领域的元素（例如一辆车或一次股票买入）。用户需要改变这个对象的属性或特性。

为什么使用

可视化建造器（Visual Builder）的绝大部分用户都很熟悉属性表的概念——它是对象属性或设置的列表，按某种有规律的顺序摆放，可以通过和属性类型相对应

的控件进行编辑（例如对应字符串的文本框或有多个选项时的下拉框）。设计良好的属性表总的来说很容易使用，原因很简单，因为它们遵循了广为人知的习惯用法。

属性表同样也可以帮助用户在UI上建立该对象的心智模型。一张属性表会告诉用户，该对象的属性是哪些，对每个属性来说有哪些可选的值。特别是在那些把所见即所得编辑和编程混合在一起的应用里（例如GUI Builder、Web Page Builder、基于时间的脚本和动画工具）。所以属性编辑器能帮助用户如何使用系统。

如果不使用属性表，也可以使用菜单和直接操作。可以通过菜单去访问一些简单的布尔值或一对多的属性。如果用户只需要编辑一个对象的少数几个属性，那么属性表可能是一个太过重量级的方案了。用户需

要把它打开，花一些时间来看它，找到需要编辑的那个属性，编辑属性，然后再关闭该属性表。这样看起来确实有一大堆工作要做，不过好处会随着它的属性的改变而增加。如果用户连续改变了一大堆属性，那么属性表就很适合了。它们也允许同时访问大量属性，不像菜单那样，长度超过八的时候可用程度就大大降低。

直接操作则是一种完全不同的用法。例如，你可以在画布上对图形对象进行拖动、缩放、旋转、分层，完全不需要其他窗口的辅助——这些操作涉及了一大堆属性。任何元素，如果能有一个很好的可视化隐喻，就不需要属性表来描述了。但是如果没有，属性表就很有必要（如果用户希望得到精确的运动、旋转及类似属性的数值，那么它们也可能会很有必要）。

如何使用

属性表的唯一通用之处在于，各种各样的编辑控件都有一个标签，标签上写着这个属性的名字。当用户填写完一个属性的时候，新的属性值就写到了对象里。

在设计属性表的时候通常会遇到下面这样一些问题：

布局

一些系统使用两栏的表格，当用户单击右边一栏的属性值时，编辑控件就会显示出来（Visual Basic所采用的方案似乎是这种方法的事实标准）。其他属性表的外观看起来更像表单或对话框——可能有几行高的文本输入框出现在其他控件或编辑器的旁边。你得自己判断。表格可能更容易看出来是一个属性表，但是表单比表格更灵活，因为它们表现属性编辑控件的时候有很大余地——参见图4-39中Illustrator这个很极端的例子。例如，有了表单形式的布局，你就不会局限于只使用一个

表格行来处理每个属性（要更多的布局变化，可以考虑使用“右对齐/左对齐”模式）。

属性顺序

按字母顺序、分组或一个把最常编辑的属性放在顶部那样容易阅读的顺序来排序？它们各有所长。短的属性表（例如10个或10个以内的属性）通常适合把最常用的属性列在前面。列表越长，就越有必要对它们分组。但是如果用户在通过名称寻找某个特定的属性，他们可能希望有一种字母顺序的排列。和往常一样，最好给用户一些选择。不过，你仍然有责任提供一种合适的默认顺序，因此让用户自己选择并没有免除你的责任。

标签

Visual Studio和其他编程环境经常使用简短而且含义不甚明确的名称来给属性命名。当用户可能会在后面的程序或脚本中用到这些属性名称的时候，可能是没问题的（尽管更明确的命名往往更好），但如果不是这样，那就得想想更好的属性名称。使用用户熟悉的术语，而不是我们常用的特定称谓。如果一个属性很复杂，那么使用更多的词语来命名。把这个整个属性表关联到一个帮助系统，例如我们可以在用户开始编辑这个属性的时候，显示该属性的描述说明。

控件选择

在这一点上我可以写上50页。不过简短一些的描述版本是：让属性现在的取值总是可以看见；尽可能选择那些可以限制输入的控件——例如，在一对多选择的时候使用不可编辑的下拉列表框；使用平台支持的标准编辑器来表示特别的属性类

型。例如颜色、字体、文件名。参见第7章来了解更多关于控件选择的信息。

馈。如果可以，提供一个撤销（Undo）机制。撤销会帮助用户恢复偶然的或不想要的改动。如果是及时更新属性值，那么能不能撤销就更重要。

什么时候保存新的属性值

很多用户界面只是简单地在用户输入完或选择完一个新值的时候马上更新对象的属性。那些更像表单的属性表可能会等到用户有意提交整个设置的时候才更新。例如单击“确定”按钮的时候。不过，如果你的软件可以实现及时刷新，那就及时刷新——这样会给用户刚才的动作提供及时的反馈。

多选对象的混合取值

有些用户界面通过完全不显示属性值来解决这个问题，如果该属性实际上没有值（例如null，或空字符串），这样有可能产生很危险的误导。另外一些界面会用一些代表值来显示它，例如星号（*），或单词“Mixed（已选中多项）”。

示例

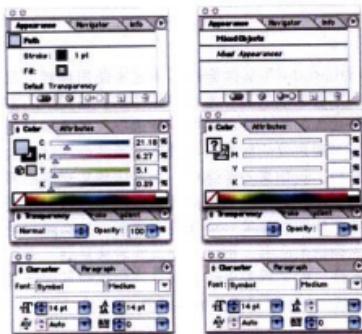


图4-39 Illustrator和Photoshop把“表单格式”的布局方法应用到了极致。这张图显示了Illustrator的属性表（事实上，还显示了两次）。它分成了几个可移动的面板，分别漂浮在工作窗口的顶部，每个面板都有一个或更多的Tab页来根据类别对属性分组。可以隐藏或重新排列这些Tab，也可以把Tab从一个面板拖动到另一个面板。对于高级用户，这种高度密集、灵活、用户可配置的方法很好，但你不会在一个随意的应用中使用这样的设计，也不会为那些还没习惯用这么复杂工具的用户提供这样的设计。

第一栏是一个属性表的截图，当一个浅蓝色的矩形在工作窗口选中时就会显示它。第二列是选中了多个不同的对象——它显示了处理混合取值的一些方法。

对它们——比较一下。注意，第二列中并没有试图显示Stroke和Fill这两个属性，因为它们并不适用于所有选中的对象上。不过，颜色确实是有的，但是显示的是空白的文本区域和问号——表示源对象有着不同的填充（Fill）和边缘（Edge）颜色。空白的组合框也提示了Transparency、Opacity和Text leading（字符间隔字母之间的垂直距离）有多个取值。

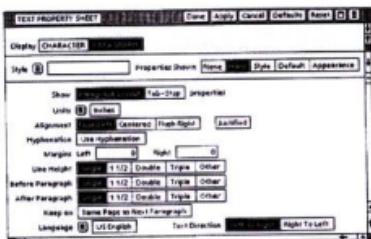


图4-40 这个屏幕截图来自Xerox Star，它显示了一个非常早期的属性表。我们可以看到，属性表的一些特征完全没有变化：注意两栏的布局、右对齐/左对齐、文本区域和单选按钮的使用，甚至还使用了响应式展开模式（虽然从这张图上看不到）。

响应式展开 Responsive Disclosure

Have you used TurboTax for the Web before?

Yes, I have used TurboTax for the Web before.
 No, I am a new user.

Have you used TurboTax for the Web before?

Yes, I have used TurboTax for the Web before.
 No, I am a new user.

Sign In

User ID Forget ID?
Password Forget password?

Remember my user ID on this computer
 I have read and agree to the terms of the license agreement. [\[link to agreement\]](#)

Sign In

图4-41 来自<http://turbotax.com>

它是什么

从一个最小限度的用户界面开始，通过在每个步骤显示更多的界面，引导用户完成一系列步骤。

什么时候使用

用户应该通过多个步骤、一步步地完成一项复杂的任务。也许因为他是计算机初级用户，或者这个任务没有什么先例，又或者这个任务很少需要执行（和向导（Wizard）模式里一样）。但是你不想强迫用户每个步骤一页一页进行——你宁愿把整个界面放在一个页面上。

为什么使用

在这个模式里，界面似乎是在用户面前创建出来的，一个步骤出来一些。首先，用户只看到第一个步骤所必须用到的那一点点界面元素。当用户完成该步骤的时候，下一步需要的界面元素会加在首先出来的那些元素后面，后续的情况依此类推。

当用户看到该任务通过一个动态的界面一点点在面前展开时，他可以按照一个正确的心智模型来更快更容易地完成任务。在这里没有那种分别展示的向导屏幕

上下文之间生硬的切换：用户不会被界面从他们的工作流中生拉硬拽出来，局限在一次只显示一个屏幕那样的硬性规定中。

除此之外，由于所有的界面放在一个页面上，因此用户可以很容易回到前面的步骤改变之前的选择。每个步骤都可以重新来一遍。他很快就能看到前面步骤对后面步骤的影响。这比从一个上下文相关的屏幕跳转到另一个上下文相关的屏幕要好。

对那些偶尔执行的任务，这样的机制比一次就展现一个复杂而互相关联的一组控件更好，因为它的第一步总是一目了然——后面的步骤也是这样。用户永远不必绞尽脑汁思考。

在这个模式和响应式允许模式之间你应该如何选择呢？如果使用响应式允许模式，你将把所有选择需要的控件都显示在界面上——然后禁止那些不相关的控件，直到它们变得相关为止（再次强调，根据用户的选择来进行响应）。有时候这样做会让界面太过杂乱而且看起来很复杂。要根据自己的情况判断：如果你需要把界面放到一个很小的空间里，或者你觉得太多的控件会让界面看起来很糟糕或让用户感到紧张，那么使用响应式展开模式好了。

从只显示第一个步骤所需要的控件和文本开始。当用户完成了这个步骤，则显示第二个步骤所需要的那部分界面。让前面那个步骤的控件保持可见，并且允许用户在必要的时候回到前面。把这些控件放在一个页面或一个对话框里，这样的话用户不会觉得自己突然到了一个另外的UI空间里。

在很多这样一步一步的设计里，用户在某一个步骤所做的选择会对后面的步骤带来影响（例如，任务有分支而不是线性的）。举例而言，一个在线订单询问记账地址是否和送货地址一样。如果用户说是一样，那么界面上就不会多此一举地再次显示记账地址的输入控件。否则，在这个过程里会多一个步骤，界面会在合适的时候要求用户输入这些多出来的信息。

如果用在一个完全改变之前的UI而不是在空白区域增加控件的情况下，这项技术将会大打折扣。在用户面前重新组织界面会让他的空间感上下颠倒——这种影响是破坏性的。但是以一种有序而且可以预测的方式

来组织的话，就以一种简捷而且一致的方式为用户提供了向导。

响应式展开并不是一个新概念。在1981年第一个商业化的界面Xerox Star上已经用到了它。它的设计师考虑的是“Progressive Disclosure”（逐步展开），这是一个更广泛的概念，它包括了响应式展开，是一个主要的设计原则：“逐步展开表示，如果不是被问到或需要看到，界面的细节就会隐藏起来。因此，Star不但提供了默认设置，也把用户不太可能改变的设置隐藏起来，除非用户表示他们真的想要改变这些设置。”³确实如此。

例如在Star的属性表里，空白留给了那些需要时才显示的控件，根据用户的选择显示。例如，当用户从一组取值中选择，而这些取值中包括“其他”的时候，会显示一个额外的文本输入域让用户输入一个数字。Larry Constantine和Lucy Lockwood的“Instructive Interaction: Making Innovative Interfaces Self-teaching”⁴（启发式交互：让创新的界面具有教学功能）中，以“Progressive Disclosure”的名称讨论了这个模式。

3. Johnson, J.A.的“The Xerox Star: a retrospective (Xerox Star的回顾)”一文发表在IEEE Computer杂志，1989年9月刊，从11页到29页，也可以参见：<http://www.digibarn.com/friends/curbony/star/retrospect/>。
 4. 参见<http://foruse.com/articles/instructive.htm>。

示例



图4-42 这个例子来自Northwestern Mutual的一个内部应用。用户首先选择“TSO”作为请求类型，应用就显示了一个新的方框，要求输入TSO信息。当用户选择了“Delete ID”，应用又显示了一个新的方框。用户之前的选择在这个页面上仍然保持可见。

图4-43 现在有这样一个例子，反映了我之前说的不良设计：改变之前存在的界面，而不是在界面上增加控件。不过，从这个例子看来，如果改变的规模很小，还是可行的。它来自Mapblast的界面，用来输入地图上要定义的地点。它显示了两个配置：首先是一个具体的地址，其次是一个地点的名称而没有地址。用单选按钮来选择使用哪个，该设计可能得益于首先显示没有选中任何单选按钮，然后也没有地址或地点名称的控件，当用户单击单选按钮时，相应的控件才显示出来。

响应式允许 Responsive Enabling

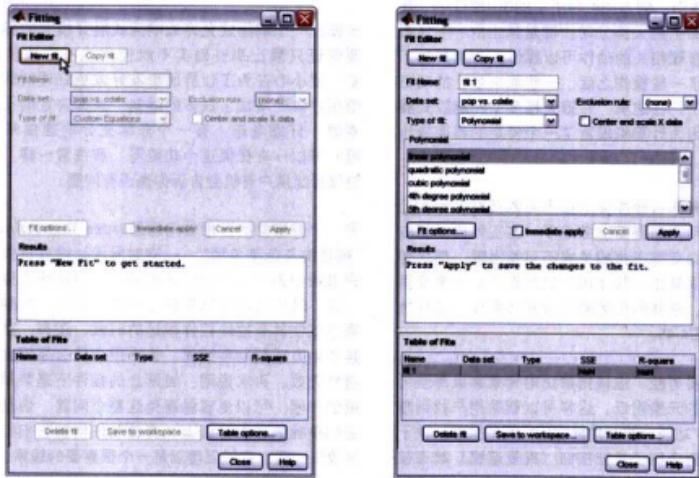


图4-44 Matlab的曲线拟合工具

它是什么

从一个基本上被禁止的UI开始，通过在每个步骤允许更多的用户界面部分有效，来引导用户完成一系列步骤。

什么时候使用

UI引导用户一步一步进行一个复杂的任务，可能因为他是一个电脑初学者，或者因为这个任务很少用到（同向导模式一样）。但你不强迫用户的每个步骤都要从一个页面到另一个页面跳转，你希望在一个页面里保持所有的界面元素。还有，你希望保持界面稳定，宁愿不要在每个步骤动态重组界面，就像你在响应式展开模式里所做的那样。

为什么使用

和响应式展开模式类似，该模式利用了计算机显示的延展性来引导用户交互式地一步一步往前走。这样，用户得到了一个机会形成一个正确的心智模型来认识原因和后果。用户界面本身会告诉他一些选择可能的结果。如果把这个复选框选中，那么我必须填写这四个变得有效的文本框。

还有，用户不能去做那些可能会给他带来麻烦的事，因为界面已经通过禁止它们（让它们变得无效）锁定了一些动作。因而这样也避免了不必要的错误信息。

在有些应用系统中，例如图4-44所示的GUI窗口中，开始的时候，界面上绝大部分动作都是禁止的——只有和用户第一个步骤相关的动作可以操作。当用户做了选择并且执行了一些操作之后，一些原来禁止的项目现在可以操作了。在这里，它很大程度上和响应式展开有点类似，因为计算机规定了一个特定的界面操作顺序。

一个类似但没有那么顺序化的技术可能在桌面界面上用得更多。在用户在界面上完成了一些工作后，相应的动作或设置会变得不再相关或不可能出现，然后那些动作会被界面禁止，除非用户确定无论如何需要重新让它们有效。总体的任务顺序没那么重要。这种技术的例子参见图4-45。

无论何时，只要可能，应该把禁止的元素靠近那些可以使它们有效的元素附近。这样可以帮助用户找到那个神秘的“打开”功能，并且理解它们之间的关系。关于这个模式，有两个例子是把按钮（或复选框）放在被

禁止的元素左上角的，这样也遵循了界面上自然的从上到下、从左到右的视觉流。

当设计一个响应式允许或响应式展开模式的界面时，要保证只禁止那些确实不能使用或不应该使用的元素。要小心在为了让界面更友好或更好理解的诱惑下给用户体验增加了过多的限制。当你决定禁止哪个元素时，仔细考虑：有一个非常充分的理由来禁止它吗？可以一直提供这个功能吗？和往常一样，可用性测试会让用户有机会告诉你是否有问题。

另一个要避免的可用性问题是Bruce Tognazzini所说的“神秘的灰色菜单项”⁵——有时候设计没有提供任何线索来说明为什么一个元素被禁止。因此在这里再强调一次，最小化那些需要禁止的控件集合，特别是它们离可以使其有效的控件很远的时候。还有，在界面上某个地方或帮助系统里，告诉用户什么原因会让一个特性无效。再次说明，当禁止的控件不是菜单或菜单项的时候，可以更容易避免这整个问题，但是不要把它们单独放在主界面上，应该和任何把它们打开的控件放在一起。空间上接近是一个很重要的线索。

⁵ 参见“Ten Most Wanted Design Bugs (10个最严重的设计师Bug)”，<http://www.asktog.com/Bughouse/10MostWantedDesignBugs.html>。

示例

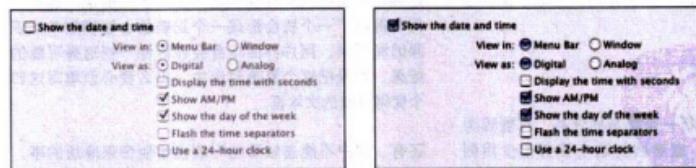


图4-45 Mac OS X系统的参数提供了一个典型的基于二元选择的禁止例子：OS X应该在菜单条上显示日期和时间吗？如果用户选择显示，那么她得到了一整套关于如何显示它们的选择。如果用户选择不要，那么那些选择就完全不相关，因此它们就被禁止了。这样的行为（再加上复选框下缩进的选项）告诉用户，如果复选框选中，这些选项会影响日期/时间的显示方式——如果没有选中，就不会有什么影响。



图4-46 你也可以反转这个模式的感觉。来进行“响应式禁止”。这个Toyota Prius和Lexus汽车里面的导航系统在用户输入一个目的地的时候就使用了这样的技术。已经知道了给定的搜索区域有哪些街道，系统根据用户输入的一半字符来缩减可能的街道名称。这样，它就禁止了那些在当前输入的地地址下不出现的那些街道；用户需要考虑的按钮变少了。再加上系统“知道”他们想要输入什么的保证。地址输入因此变得更容易也更让人愉悦了。（当只留下几个街道匹配时，系统把键盘整个移走，只显示街道的一个选择列表——参见第5章将要提到的自动完成模式）。

参见http://lexus.com/models/rx_hybrid/interior.html——这些屏幕截图的源地址，可以看到一份演示。

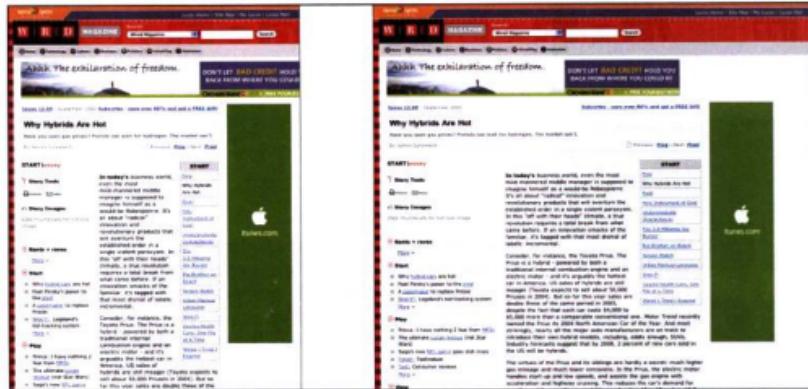


图4-47 来自http://wired.com

它是什么

当用户调整窗口大小的时候，相应地调整页面元素的大小和位置，让页面一直保持填满的状态。

什么时候使用

用户可能需要更多——或者更少的空间来显示窗口、对话框、页面的内容。这有可能发生在任何页面上有很多文本（例如网页）的时候，或者页面是一个高度信息化的控件（比如表格、树、图形编辑器）的时候。它不适用于可视化设计需要特定大小（不多也不少）的屏幕空间的时候。

为什么使用

除非你是在设计一个“封闭的”UI（例如信息亭、手持设备或全屏幕的Video游戏），不然你不能假设用户将在什么样的情况下看到你的用户界面。屏幕尺寸、字体偏好、屏幕上的其他窗口，或者对用户来说任何特定

页面的重要性——这些全都不受你的控制。于是，你将怎样为所有用户决定一个最理想的屏幕大小呢？

给用户一点点页面布局的控制权，这样可以让你的用户界面在变化的情形下具有更大的灵活性。它也可以减少用户对界面的敌对情绪，因为用户可以改变它来适应自己的即时需要和当前上下文。

如果你觉得这样还不那么令人信服，那么考虑一下固定的、非流式的界面在语言或字体大小改变的时候会发生什么情况。那些竖直的栏还整齐排列吗？页面突然变得太宽，或者甚至在边界的地方少了一块吗？如果没有，很好，你已经有了一份简单而健壮的设计。但那些可以很好地适应窗口大小变化的页面通常也可以适应语言或系统大小的变化。

如何使用

在窗口改变大小的时候，让页面内容一直“充满”窗口。多行的文本应该在右边的边界折行，除非它有10到12

个单词那么宽（稍后详细说明）。树、表格、图形和中央舞台模式里的编辑器应该相应变大。而它们的边界仍需要保持紧凑。如果页面上没有任何表单一样的元素，那么水平拉伸应该会让文本框拉长——如果他们需要输入超过文本框正常宽度的内容。用户会感激这种机制。与此类似，任何滚动的元素（例如表格和列表）应该变长，通常也会变宽。

网页及和网页类似的用户界面应该允许页面的正文内容填满新的空间，同时保持导航设备和路标牢牢地固定出现在上面和左边。背景颜色和图案通常会充满新的空间，就算内容本身不能充满的情况下也应该做到这一点。

如果窗口对于内容来说变得太小了会怎么样呢？你可能会在它周围放上滚动条。要不，就相应地根据需要缩减空白。如果窗口变得确实非常小的时候，彻底去掉一些东西也是可能的，不过最重要的内容应该会保持到最后。

6. 参见“Use Reading Performance or User Preference (用户偏好或用户阅读偏好)”一节，<http://usability.gov/guidelines/>。

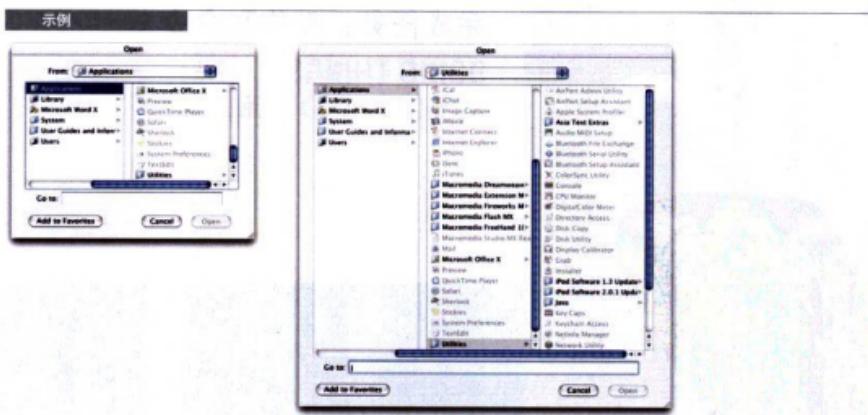


图4-48 Mac OS X允许你重新定义标准“Open”对话框的大小，它使用了流式布局模式。这样很好，因为用户在希望的时候可以看到更多的文件系统层级结构，而不是被限制在一个预先定义的小空间里。另外，这个对话框里也表现了级联列表（Cascading Lists）模式。

如果你在处理文本段落，记住如果太宽的话，它们会变得难以阅读。图形设计师们致力于达到容易阅读时的最佳文本宽度，其中一个标准是每行平均10到12个英文单词。另一个标准是30em到35em宽（em宽即小写字母m的宽度）。当文本比以上数据更宽的时候，用户的眼睛从一行的末尾到下一行的开头需要掠过大长的距离；如果收窄得太厉害，反复换行也会让阅读变得困难。

（据说有证据表明，一行更长的文字，例如每行100个字母比一行更短的文字容易阅读，尽管用户还是更习惯每行少于55个字母⁶。）

一个行为良好的流式布局在复杂的网页上很难实现，特别是如果你想去掉表格而只使用纯粹的CSS来定义样式的情况下。在Visual Basic和Visual C++里，这么做也很困难（或者说，至少在很长一段时间里曾经很困难）。不过，Java提供了好几种页面布局管理器可以用来实现流式布局。

05

完成任务：动作和命令

**DOING THINGS:
ACTIONS AND COMMANDS**

本章要讲的是界面上的“动词”。我们已经花了大量篇幅来介绍用户界面的整体结构和流（flow）、可视化布局，还有页面上一些诸如窗口、文本、链接、静态元素等“名词”。第6章会花更多的精力在名词上，然后第7章将讲述一些传统的（以及少量不那么传统的）控件和小部件：那些让用户提供信息、设置状态，但所进行的动作不太多的界面元素。

现在我们来看按钮和菜单。

听起来让人兴奋，是吗？也许没有人会这么觉得。桌面应用的界面从第一台Macintosh开始就已经使用了菜单条，按钮的历史就更悠久了。我们所说的“按钮”（button）只是某种物理设备的可视化反映，而它们在GUI之前就已经出现了。

按钮和菜单确实有很长的历史了，也有很多可以借鉴的最佳实践。那些标准的平台风格指南，例如Windows、Macintosh和Palm，一般会让你的用户界面相当接近于可用的程度。绝大部分用户依赖习得的习惯来使用菜单、找到按钮，因此遵循这些习惯用法是可以的，即使是在它们有很多限制或者看起来没什么意义的情况下。

诸如剪切、复制和粘贴等常见功能也具有很多历史性的包袱——如果我们现在能重新发明这些元素，它们的形式可能会不一样，但是就算是普通的桌面计算机用户也已经学会了它们“会具有的行为”。对于弹出菜单（上下文相关的菜单）也是一样，一些用户好像希望无论在什么地方都能使用它们，而其他一些用户永远不会去寻找。拖拽并没有受到历史的限制，但它绝对应该按照用户直觉的期望去表现，否则就会破坏了那种直接操作的感觉。

所有这些说明，你可以有一些很酷的特征来让界面不那么枯燥无味，并且让其更可用一点。你的目标应该是让合适的动作处于可用状态，为它们提供适当的标签，让它们容易找到，并支持动作序列。有一些创造性的方法可以做到这些。

首先，我会列出一些常规的动作表现形式。

按钮

按钮直接放在对话框和页面上，而且通常根据相关程度分成不同的组（参见按钮分组（Button Groups）模式）。它们尺寸大、可读性好，显而易见，就算对没有什么经验的用户来说也非常有用。但它们会占据页面上大量空间，而不像菜单条和弹出菜单那样节省空间。

菜单条

菜单条是绝大多数桌面应用的标准配置。它们通常会把一个应用的完整动作集合显示出来，并以一种可预测的方式来组织（例如文件、编辑、视图等）。一些动作作用于整个应用，一些则作用于选中的对象。菜单条通常会重复那些上下文菜单和工具条上的功能，因为它们是可以访问到的——屏幕阅读器可以阅读它们，键

盘加速器可以访问到它们，等等（可访问性本身让菜单条在很多产品里面不可或缺）。菜单条已经开始出现在一些Web应用中了，尽管还不像桌面应用那么普遍。

弹出菜单

也叫上下文菜单，弹出菜单通常出现在鼠标右键单击之后，或者面板/对象上一些类似的动作之后。它们往往列出上下文有关的公共动作，而不是界面上所有可能的动作。记得让它们保持简单。

下拉菜单

用户通过单击一个下拉控件（例如组合框）来得到这些菜单，你可以在很多网页上找到这样的菜单。使用下拉菜单来列出动作通常不是一个好主意，因为下拉控件的典型应用是用来设置状态的，而不是执行动作。

工具条

标准的工具条是一个长长的狭条，上面有很多图标按钮。通常也会有其他形式的按钮或控件，例如文本输入框或下拉选择器（**Dropdown Choosers**，见第7章）。当它表示的动作可以明显地用图标表现出来时，图标工具条效果最好；当那些动作确实还需要文字来描述的时候，试试其他控件，例如组合框或带文本的按钮。含义不清的图标是困惑和不可用性的典型来源。

链接

链接通常用在网页上，但现在也可以在很多应用中找到。它们也可能是蓝色带下划线的文字或一些看起来像链接的短语，而实际上是用按钮实现的。这里的要点是，凡是蓝色带下划线的看起来就是可以单击的东西，人们会找到并且使用它们。也就是说，很多Web应用的设计师现在正在通过坚持这样一个原则来有意识地减少潜在的含义混杂：用按钮调用动作，用链接显示其他内容。

动作面板

你可能知道它们的另一个名字：“任务面板”。动作面板本质上是一些按钮，用户不需要去寻找，它们在主界面上总是可见的。在动作用文字表示比图标表示更好的时候，它们也可以很好地取代工具条。参见动作面板（**Action Panels**）模式。

不过还是有一些不可见的动作，它们不会有什么可见的标签来表示它们是做什么的。除非你在界面上有一些书面的指示说明，否则用户就需要知道（或猜测）它们是不是存在。因此，如果不知道，它们就没什么用。换句话说，用户不能通过扫描来发现可能存在哪些动作——但是如果使用按钮、链接、菜单条，那些动作就是可以看到的。用户可以通过它们进行学习。在可用性测试中，我发现很多用户在看到一个新产品之后，能够老练地打开菜单条，一个项目一个项目地查看，看这个菜单项能做什么。

也就是说，你几乎总是需要使用一个或更多的不可见动作。例如，人们常常期望可以双击某个对象。键盘（或类似的设备）有时候是那些视力较弱的用户和不能使用鼠标的用户唯一可以使用的操作方式。一些操作系统的和应用程序的专家用户宁愿向Shell输入命令，并/或使用它们的键盘动作。

在对象上双击

用户喜欢把双击看做“打开这个对象”或者“在这个对象上进行任何默认操作”。具体处于哪种情况要看当时的上下文。例如在一个图形编辑器里，双击一个对象一般意味着打开一个属性表或一个特别的编辑器。在绝大多数操作系统中，双击某个应用的图标意味着启动这个应用。

键盘动作

键盘快捷方式，如用得很广的Control+S（保存），应该设计到大部分的桌面应用中，因为它们有助于可访问性和用户的高效率使用。主流的UI平台包括Windows、Mac、Gnome，还有Palm OS，都有自己的标准快捷键风格指南——而且它们十分相近。另外，菜单和控件常常会有带下划线的访问键，让用户访问这些控件而无须进行鼠标单击或切换Tab页（输入Alt键再加上带下划线的字母来触发这些动作）。

拖拽

在一个界面上拖拽并放下一些对象通常意味着“把它移到这里”或者“用另一个对象来操作这个对象”。换句话说，某个用户可能拖动一个文件到一个应用图标上，并说“在那个应用里打开这个文件”，或者她可能把这个文件从一个文件Finder拖动到另一个地方，也就是移动或复制了这个文件。拖拽也是上下文相关的，但是几乎总是这两种动作之一。

命令行键入

命令行界面一般会允许自由调用软件系统里的所有动作，不管它是操作系统还是应用软件。我把这些动作看做是不可见的动作，因为很多命令行界面（Command-Line Interfaces, CLI）并不会轻易暴露那些可用的命令。它们很隐蔽，不容易发现。尽管一旦你知道了，它们的威力就很大——只要一个设置好的命令就可以完成很多工作，因此CLI对那些愿意下功夫学习这个软件的人来说是最合适的。

扩展边界

PUSHING THE BOUNDARIES

一些应用的习惯用法让你可以自由设计一些不标准的按钮和控件。那些可视化的编辑器、媒体播放器、为专家用户准备的应用、即时通信软件、游戏，以及任何其他可能好玩又有趣的东西，都有很多用户觉得非常好奇，想弄清楚怎样去使用的一些不太常见但又设计得很好的界面元素。

什么地方可以更有创意？考虑下面这个列表：另外，可视化的按钮和菜单通常比键盘动作要容易使用。从这几条总结下来，有这样一些可能的动作：

- 可以单击的图标
- 可以单击，但看起来不像按钮的文本
- 对鼠标移过产生反应的某个东西
- 看起来可以操作的东西
- 放在屏幕上任何地方的东西

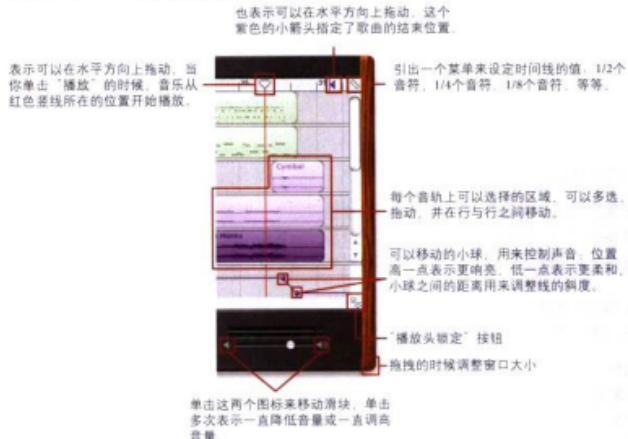
在应用系统变得太难弄懂之前，你可以侥幸用上多少创意？



Garageband

要了解真实的例子，可以看看上图所示的Garageband应用。这个界面上有很多地方都很有意思。一些对象明显是按钮，就像播放器控件——回放、播放、快进等——还有滚动条箭头。在该图中还可找到滑块和旋钮。

但是窗口右边，在那根红色和木纹边界之间的东西，就不那么容易识别了。在你看来，界面上哪些地方看起来可以单击？为什么？如果你希望，可以提前看看图5-2，做个弊（如果你已经知道怎么使用Garageband了，请让我再啰嗦一会儿）。



Garageband的动作

图5-2显示了界面上的哪些对象会执行动作，你肯定不可能知道它们都是做什么的，因为书上不能为你提供工具提示、鼠标滑过的光标或试验性的试用。但你已经知道了其中的哪些对象是可以单击或可以操作的吗？我想你应该已经知道了。

怎么知道的？你可能用过看起来差不多的界面。界面上有一大堆功能，都是可以直接操作的，因此你已经有了良好的背景知识来假定这些好玩的元素提供差不多的功能。可能已经知道那个滑块，例如底下的音量滑块有时候在最后有一个“跳跃按钮”——而你可能是从iTunes知道音量滑块本身的。你可能会猜测那个方形的图标是按钮，通常用来提供和演示相关的动作。它们在Word和Powerpoint里用得较多。你可能在别的地方见过一根竖直线在一个反向三角的上面——有时候可以移动，有时候不行（参见第8章里的指南（Guides）模式）。但是你不觉得这个三角形看起来是可以移动吗？

当一个对象看起来像是让你可以执行某个动作，例如单击或拖拽的话，我们说它具有“启示”（affordance）来执行这个操作。传统凸起按钮的启示是你可以按下这个按钮；滑动的光标意味着你可以推动它。文本框的启示是输入，蓝色带下划线文本的启示是单击（链接）。还有，任何对鼠标光标有反应的对象都意味着可以做某个操作，尽管你可能不一定知道那到底是怎么操作的！

图5-2指出了Garageband界面的各个启示。这是一个重要的概念。在软件界面里，用户不容易得到很多感觉上的线索来知道什么东西可以拧或者可以握住；视觉效果可以提供绝大多数的线索，然后就是鼠标滑过的反应。利用这两点来传递启示。

这里有一些具体的设计建议：

- 在任何可能的情况下遵循习惯用法。重用人们已经知道的界面概念和控件，例如上面的音量滑块。
- 利用拟3D的阴影和向下的影子来让某个东西看起来有向上凸起的效果。
- 当鼠标指针滑过某个可以单击或拖拽的对象时，把鼠标指针变成不同的形状，例如一个手指或一只手。
- 使用工具提示或某种描述性的文本，来告诉用户鼠标指针所指的对象是什么。如果不需要这么做，那最好了——说明你有了自描述的设计——但是不管怎样，很多用户都希望能看到工具提示。

模式

THE PATTERNS

本章最开始的两个模式讲的是两种表现动作的方式——一个很常见，另一个则有点陌生。当你发现自己无意识地把动作放到菜单条或弹出菜单的时候犹豫了一会，那么考虑一下使用按钮分组模式或动作面板模式。

44 按钮分组
Button Groups

45 动作面板
Action Panel

让“完成”（Done）按钮变得突出会改善很多网页和对话框上最重要的按钮效果。智能菜单项是一种改善某些菜单动作的技术：这是一个很笼统的模式，对很多种菜单（或按钮和链接）都很有用。

46 突出的“完成”按钮
Prominent “Done” Button

47 智能菜单项
Smart Menu Items

如果可以很快完成用户发起的所有动作，我们会很开心，但是事实并非如此——有些动作确实需要花费时间。预览（Preview）模式可以在一个动作提交之前预先告诉用户将会发生什么事。大家都了解进度提示（Progress Indicator）模式是让用户知道当一个操作正在进行的过程中情况如何，同时，可取消性（Cancelability）模式也意味着用户界面在用户请求的时候可以停止一个操作的能力。

48 预览
Preview

50 可取消性
Cancelability

49 进度提示
Progress Indicator

最后三个模式——多层撤销、命令历史、宏——都是处理一组顺序动作的。这三个连锁反应的模式在复杂应用中非常有用，特别是那些用户愿意努力学习，并经常使用的软件（这一点也是为什么这几个模式的示例都来自Linux、PhotoShop、Word和MatLab的缘故）。小心这些模式，因为它们实现起来不容易。这需要应用把用户的动作建模成不连续的、可描述的，有时候还需要可以反转的操作。另外，这样的模型很难套用到现有的软件体系结构中去。*Design Patterns*一书中的命令（Command）模式可以为实现这些模式提供很好的参考意见。

51 多级撤销
Multi-Level Undo

53 宏
Macros

52 命令历史
Command History

这些模式已经让本书涉及了很多实现细节，我们现在回到界面设计领域看看。



图5-3 Word自定义对话框中的两组按钮

它是什么

把相关的动作组织成一组按钮，彼此水平对齐或垂直对齐。如果超过三个或四个动作，那么可以多分几组。

什么时候使用

你在展示一小部分彼此相关的动作，例如二到五个——它们可能作用在同一个对象上，或者有着类似或互补的功能。确定、取消、应用和关闭通常组成一组。它们都作用在对话框上形成一个完整的集合。与这些类似的是，对一个列表中的对象进行处理的按钮（例如上移、下移、删除）也应该分组展示。

为什么使用

按钮分组可以让一个界面变得具有自我描述能力。在复杂的布局中，组织良好的各组按钮容易选择，而且因为它们很明显，它们可以很快告诉用户有这些功能可以使用。它们声明：“在这个上下文里，有这些按钮可以使用”。

在第4章中讨论过的格式塔原则也在这里起作用。相邻暗示着彼此相关——几个按钮放在一起，因此它们可能完成类似的功能。视觉相似性也是，例如，若你让这组按钮大小都一样，那么看起来是一个整体。合适的大小和对齐可以帮助一组按钮组成一个大一些的组合图形（闭合原则）。

如何使用

把可能有问题的按钮分成一组。用简短而又含义明确的动词或动词短语作为标签，而且，如果不是用户需要的话，不要使用高深的专用术语。不要把作用在不同对象上或有着不同作用范围的按钮混在一起；参见后面的例子来看怎样做到这些。然后，把它们分到不同的组里。

该组的所有按钮都应该有同样的宽度和高度（除非各个按钮的标签长度差异很大）。可以把它们并排起来

放成一列，或者若所有的按钮都不是太宽的话，则可以摆成一行。超过一行或超过一列都不是很好。

分组按钮的位置

PLACEMENT OF BUTTON GROUPS

如果所有的按钮作用在同样的对象或同样一组对象上，那么把这组按钮放在这些对象的右边。你也可能把它们放在下面，不过通常复杂界面元素的底部是用户的“盲点”，例如多栏列表和树——用户可能根本就看不到按钮。不过，如果你有把按钮放在下面效果更好的设计，那么不妨做些可用性测试看看结果。如果

按钮很多，而且它们都有小图标，你也可以把它们放在工具条或者像工具条那样的狭长区域上。工具条和这样的狭长区域通常在页面的顶部。

当按钮影响的是整个页面或整个对话框的时候，例如关闭或确定，你应该遵从平台的风格指南。通常它们会组成一列放在页面的最右边，或者组成一行放在页面的右下角。用户习惯去这两个位置寻找这些按钮（关于习惯（Habituation），参见第1章）。

不过，单独的关闭按钮不一样。有时候你可以不管它，只要用一个X型图标并把它放在右上角就可以了——对这样的按钮，Windows用户本能地知道怎么做，而其他平台的用户通常也知道这个习惯用法。

示例

图5-3中的Word对话框有两个按钮分组——一组是针对列表的动作，另一组是更普遍的动作。这两组按钮的“范围”相对来说比较合适：针对列表的按钮只会作用在那个Tab上的对象，所以它们放在Tab内部。而其他的按

钮放在Tab外面，组成一组。关闭按钮放在右下角，这是这种对话框关闭按钮的标准位置，在Windows和Mac上都一样。

9. **Things That Make Us Smart: Defending Human Attributes in the Age of the Machine**
by Donald A. Norman
Average Customer Review: ★★★★☆
Publication Date: May 1, 1994
Our Price: \$13.97 Used: \$8.99 from \$5.99

10. **The Sibley Guide to Birds**
by David Allen Sibley
Average Customer Review: ★★★★★
Publication Date: October 3, 2000
Our Price: \$23.10 Used: \$8.99 from \$15.99

图5-4 Amazon在它的推荐页面上使用了两对按钮。那些按钮看起来有点类似（都是黄色圆角），它们紧挨着放在一起，并且，从用户的角度看，它们的功能也彼此类似。要注意，这两对按钮彼此都是水平对齐的。在满是书籍推荐的页面上，只有两组按钮——它们组成了这个页面结构性的元素。



图5-5 iTunes在主窗口的四个角上各放置了一组按钮，再加上标准的标题栏按钮（例如关闭和最小化），这样的按钮共有五组。当用户浏览音乐商店时，看起来像网页的第三个面板上甚至有更多的动作（没有显示出来），它们是一些由链接组成的行为。另外，表格里的每首歌还有一个按钮。

该界面不少于14个按钮。这还没算上四个滚动条按钮，以及三个可以单击的表格头。确实有很多功能，不过因为有了视觉上和语义上的巧妙组织，它并不会让人受不了。

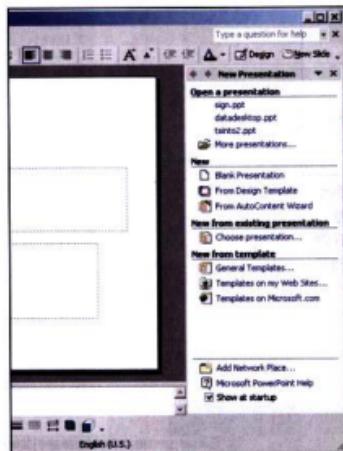


图5-6 Powerpoint的“新演示文档”面板

它是什么

不使用菜单，而是把大量相关动作放在一个UI面板上。该面板可以有丰富的组织方式，并且通常是可见的。

什么时候使用

你需要展现很多动作——用按钮分组都不够用的情况。可以把它们放到菜单里，但是你可能根本就没有菜单，或者你宁可让这些东西的可见性更强一些。弹出菜单也一样，它们的可见性还不够。你的用户可能根本就意识不到弹出菜单的存在。

又或者，你的动作集合可能太复杂了，没法放到菜单里。菜单在显示扁平组织的动作时效果最好（因为有些用户在使用右拉式菜单和级联菜单的时候十分困难），而且要以一种简单而线性的、每行一个菜单的方式来显示。如果你的动作需要分组，而且这些分

组并不适用于标准的顶级菜单名称——例如文件、编辑、视图、工具、格式——那么你可能需要一种完全不同的展现方式。

这个模式可能会占用一大块屏幕空间，因此对于小型设备来说并不是一个好的选择。

为什么使用

“什么时候使用”一节已经暗示了使用动作面板模式的两个主要理由：可见性和自由展示的需要。

通过把动作都放在主界面上而不是藏在一个传统意义上的菜单里，可以让用户更容易看到这些动作。事实上，从更普通的意义上讲，动作面板也是菜单：它只是没有采用菜单条、下拉列表和弹出菜单的形式。用户不用任何其他动作就可以看到动作面板上的内容——就在那里、就在他们面前——因此你的界面更具

有可发现性。这一点对于那些不是很熟悉传统文档模式和文档菜单条的用户来说，是特别友好的。

在界面上组织对象，有数不清的方式可以做到：列表、网格、表格、层级结构，或者任何可以设计出来的其他自定义结构。但是按钮分组和传统的菜单只能给你一个列表（而且还不是很长的列表）。动作面板的格式更自由——它们可以给你足够的自由来组织动词，就像你组织名词一样自由。好好利用这个模式！

这是一个从Web设计习惯用法回到桌面应用的例子之一。由于早期的Web应用不能动态显示菜单，因此它们显然也不可能使用浏览器的菜单条。网页设计师们发现了一些其他的方式来展现动作菜单。一组竖直方向排列的链接似乎不错（它们只是链接而不是按钮的事实。在某种程度上说并不重要，如果它们以动词为标签，或者如果它们是在执行动作，谁还会管它们是什么实现的呢？用户不会管这么多）。

把动作面板放到界面上

PUTTING THE ACTION PANEL ON THE UI

在界面上找出一块空间来放置动作面板。把它放在动作目标对象的下面或侧面。目标对象通常是列表、表格或可以选择的树，但这个对象通常是以中央舞台（**Center Stage**）模式的形式出现的（参见第4章），例如图5-6中Powerpoint那个例子所示。要记住，相邻原则很重要。如果把动作面板放在离目标对象很远的地方，用户可能看不出它们之间的关系。

面板可能是页面上一个简单的矩形，也可能是页面几个平铺的面板之一，可能是一个可移动的面板（**Movable Panel**，见第4章），一个Mac OS X中那样的“drawer”，甚至是一个单独的窗口。采取什么样的形式取决于应用的本质：如果它是可以关闭的，那么要让它很容易地重新打开，特别是如果这些动作只会显

示在该动作面板上而不会在按钮中重复出现的时候尤其应该如此。

如果你需要在不同的时间显示不同的动作，也可以做到。动作面板的内容可能会取决于当前应用的状态（例如，当前是否有任何打开的文档）。取决于某个列表上选中的那些元素，或其他一些因素。让动作面板变成动态的。内容的变化也会吸引用户的注意力，这样很好。

组织动作

STRUCTURING THE ACTIONS

下一步，你需要决定怎样组织这些需要显示的动作。下面有一些方式可以供你参考：

- 简单列表
- 多栏列表
- 分类的列表，就像图5-6中所示的那样
- 表格或者网格
- 树
- 可关闭的面板（**Closable Panels**）
- 上面这些方式的任意组合

如果要对这些动作分类，可以考虑一下以任务为中心的方法。把它们按人们可能做什么的方式来分组。例如，在Powerpoint那个例子里，有一个“打开一个演示文档”的组，再加上其他几个创建新幻灯片的组。

应该尽量线性地展示这些分组。想像一下向某个看不到屏幕的大大声读出这些动作的情形——你可以按照逻辑顺序及明确的起点和终点来读出它们吗？当然，这也就是盲人用户可能会“听”到的界面。

顺便要说的是，你也可以把一些控件放在动作面板里，例如搜索按钮旁边的文本输入框。

为动作加上标签 LABELING THE ACTIONS

对于每个动作，你可以使用文字、图标或文字加上图标作为标签。具体采用什么方式要视怎样能最好地表达动作的本质而定。事实上，如果使用的主要还是图标，那么最终得到的是传统的工具栏（或者调色板工具栏，假设你的界面是一个可视化建造器类型的应用的话）。

动作面板上的文本标签可以比菜单或按钮上的标签更长。例如，你可以使用多行标签——在这里不需要过度吝啬文字。只要记住，长一些的、更具描述性的标签对于新用户和不经常使用的用户更好，他们需要学

习（或记忆）这些动作是做什么的。这些花在长标签上的额外空间对那些密集的高性能运作的界面可能并不合适，因为它们的用户主要是熟练用户。而且如果文字太多，甚至新用户也只会一扫而过。

通常不需要把这些动作做得跟按钮一样，就算它们确实是用按钮实现的也不必如此。蓝色的文本表示它们可以单击，因为它们看起来像网页上的链接；通过在鼠标滑过的时候给它们添加下划线可以加强这种效果。这就是微软在界面上使用动作按钮时采用的方法。尽管去试试。不过，记得做一些可用性测试来确定用户确实把这些动作看做可以单击的对象，而不是普通的文本或图片。



图5-7 这个Windows文件管理器的界面显示了在动作面板上的图片目录。微软把这个特性叫做“任务面板”(Task Pane)。这个面板由几个可以关闭的子面板组成（参见第4章，可关闭的面板模式），每个子面板上有一组相关的动作。

注意前两组动作：图片（Picture）任务和文件及目录（File and Folder）任务，它们完全是面向任务的。这是一组动词短语（View、Order、Print，以及Copy），而且它们涵盖了用户通常想要执行的任务。你也许会回忆起第1章，其中我们讲到过根据对象列表、动作列表、目录列表和工具列表来组织对象。前两组动作是动作列表的好例子。但是相反，这个面板上的第三组是一个对象列表。

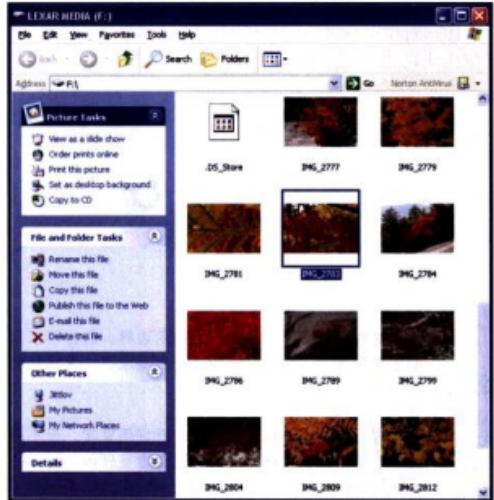


图 5-8 同样是这个面板，但是有一个图片文件被选中了。任务面板根据上下文动态地发生了变化——这次列出的任务跟上次没有文件选中的时候不一样了。

突出的“完成”按钮

Prominent ‘Done’ Button

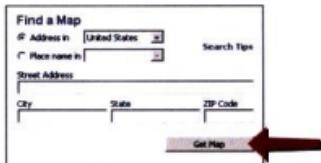


图5-9 来自<http://mapblast.com>

它是什么

把完成一项任务的按钮放在视觉流的末尾，加大它的尺寸并为它提供一个合适的标签。

什么时候使用

无论什么时候，如果你需要在界面上放置一个像“完成”（Done）、“提交”（Submit）或“确定”（OK）那样的按钮，就可以使用这个模式。更通俗地说，在任何事务的最后一个步骤加上一个从视觉上看非常醒目按钮——例如在线购物的事务或提交一组设置的事务。

为什么使用

明显且容易理解的最后一步让你的用户有种完成的感觉。毫无疑问，当这个按钮接下的时候，事务就完成了：别让他们一直在那等待，一边等一边猜测他们的工作是已经做好了，还是白忙一场。

让最后一步非常明显是该模式的本质。好好使用这个模式需要用到第4章中提到的布局概念——视觉层次、视觉流、分组、对齐。

如何使用

让这个按钮看起来确实像一个按钮，而不是一个链接：要么使用平台默认的按钮样式，要么使用一个中

等尺寸的按钮图形，并给它清楚的边界。这样可以帮助按钮在这个页面突出显示，不会淹没在一一大堆其他元素当中。

在为这个按钮添加标签的时候，最好用文字，而不用图标。文字对于这样的动作来说更容易理解，特别是当绝大多数的用户事实上在寻找一些“完成”、“提交”、“确定”这样的按钮时。标签上的文字可以是动词或一个简短的动词短语来描述将来的动作，用用户的话来说——“发送”（Send）、“购买”（Buy）或“改变记录”（Change record），这样的标签比“完成”（Done）更明确，有时候也更有利沟通。

把按钮放在用户最有可能找到的地方。在页面、表单或对话框上，一直跟着任务流往前走，然后把这个按钮放在最后一步的下面。通常情况下这个位置会在页面的底部或右边，或者右下角。你的页面布局也可能为它们设置了一个标准区域（参见第4章的视觉框架（Visual Framework）模式），或者在平台标准中有规定，如果有这样的规定，那就遵守规定。

在任何情况下，确保使这个按钮靠近最后一个文本输入框或者控件。如果隔得太远，用户可能没法在完成工作之后立刻找到它，然后他可能去寻找可以使用的控件/方式来看“下一步我该做什么”。在Web上，他可能会在结束的时候放弃了这个页面（可能是一次购买），而他自己并没有意识到这一点。

图5-10显示的是一个典型的Web表单。你很可能在根本沒有阅读标签的时候就看到了，仅仅因为它有突出的可视化设计的缘故：

- 桔黄色很显眼。它的饱和度高；与白色的背景形成了明显的对照。同时和左上角站点图标上的桔黄色形成了呼应——这两个地方是页面上唯一两处使用了同样桔黄色的地方。（注意此处也通过这两个桔黄色的元素来体现对角平衡模式的使用。）

- 按钮所使用的图片看起来像一个按钮。它是圆角的（或者说“药丸”形状），稍微有点阴影，这样，从背景上看起来有种凸起的感觉。
- 按钮的位置在表单的右边缘，直接放在表单主体的底下。这样，同时从任务流（用户的工作顺序是从上到下）和视觉流把用户的视线带到该按钮上。
- 按钮的左边、右边和底下都留有空白。这让它和表单成为了一个整体，而且周围并没有其他元素会影响到它。

图5-10 来自`http://ofoto.com`

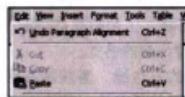


图5-11 Word的编辑菜单

它是什么

动态改变菜单的标题，以便在调用之前明确地知道它的功能。

什么时候使用

你的用户界面有一些菜单项，它们是作用在具体对象上的（例如关闭）。或者它的行为在不同的上下文中有些细微的区别（例如撤销）。

为什么使用

确切地表明它们将会做什么的菜单项会让界面更加容易理解。不言而喻，用户不必去考虑它会影响到哪个对象。也减少了用户不小心误操作的可能性，例如误删除“第8章”而不是“脚注3”。因此，它鼓励了一种安全的试探。

如何使用

每次用户改变了选中对象（或者当前文档最近一次不可撤销的操作等），修改即将作用在它上面的菜单项，让这个菜单项包含动作的细节。显然，如果没有选中的对象，你应该禁止这个菜单项，来加强它和它所作用对象之间的关联。

顺便要说的是，这个模式在按钮标签、链接或其他任何在UI上下文中作为“动词”的元素上也同样适用。

如果有多个选中对象，该怎么办？这里没有一整套的

指导方法——在现有的软件里，这个模式在文档和撤销操作中应用最多——但是你也可能会把它写成复数形式“删除一组选中的对象”（Delete Selected Objects）。

示例



图5-12 这个菜单来自Illustrator的菜单条。在这里，用户最近一次使用的滤镜（Filter）是“Hatch Effect”。菜单记住了这个滤镜，因此它把前两个菜单项修改成：（1）再次使用这个滤镜；（2）在重新应用之前修改滤镜（“Hatch Effects...”会打开一个对话框来修改它的参数）。用户可能已经应用了很多滤镜，因此她发现提醒她最近一次使用的滤镜很有帮助。如果要应用同样的滤镜，还有很方便的快捷键可以使用！

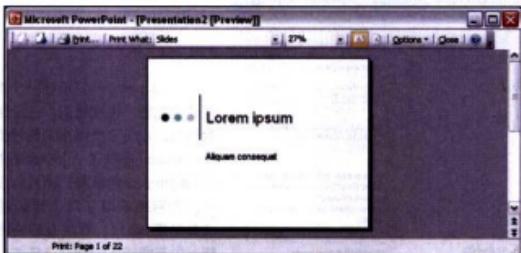


图 5-13 Powerpoint 的打印预览屏幕

它是什么

在用户执行某个动作之前，为其显示执行结果的预览或小结。

什么时候使用

用户即将做一个“重量级”操作的时候——例如打开一个巨大的文件，打印一份10页的文档，提交一份花了不少时间填写的表单，或者在Web上提交一次购买请求。用户希望得到一些他会顺利进行的保证。如果失败了会花费很多时间或要付出很大代价。

另一种适用情况是，用户将要执行一个图形操作，可能这个操作并不是那么重量级的，但是他仍然想提前知道将会发生什么情况。

为什么使用

预览可以帮助用户避免各种错误。用户可能刚做了一次排版，或者可能在到达下一步操作之前已经有了一些误解（例如在线购买的时候买错了东西），通过为他显示一个小结或一份可视化的描述，来告诉他将会发生什么，他就得到了一个可以回到之前的步骤或更正任何发现错误的机会。

预览也可以使应用变得更具自我描述性。如果某个人从来没有使用过这个特定的操作，又或者他不知道在特定环境下这个操作将会有什么结果，那么用预览来解释好过任何文档说明——用户恰好在他需要的时间、需要的地点学到了关于这个操作的知识。

如何使用

就在用户提交一个操作之前，显示任何可以告诉他将会发生什么结果的信息。如果是一份打印预览，显示页面在选定的纸张大小上的打印效果；如果是一个图片操作，显示一份图片将会变成什么样子的效果；如果是一个事务，显示关于这个事务，系统所了解到的所有信息的小结。把那些重要的内容显示出来——不多，不少，正好。

从预览页面为用户提供一种方式直接执行操作。没有必要让用户关闭预览或关闭导航到别的地方去（执行操作）。

同样，让用户有某种回退的方式。如果他可以通过修改前面输入的信息来改正一个事务，那么也要给他这样的机会。在很多向导和其他线性过程里，这只是一个往后走几个步骤的导航问题。

Review the information below, then click "Place your order."

Shipping Details Shipping to: [Change] Jenifer Tidwell 77 Massachusetts Ave Cambridge, MA 02139-4303 United States Shipping Options: <input checked="" type="checkbox"/> Standard Shipping (3-5 business days) <input type="checkbox"/> Two-Day Shipping (2 business days) <input type="checkbox"/> One-Day Shipping (1 business day)	Order Summary Item: \$17.99 Shipping & Handling: \$3.99 Total Before Tax: \$21.98 Estimated Tax: \$0.00 Order Total: \$21.98 Why didn't I qualify for FREE Super Saver Shipping?
The following items will arrive in 3 shipment: Need to <input type="checkbox"/> Change quantity or delete? <input type="checkbox"/> Estimated ship date for this item: July 15, 2005	
⑤ Harry Potter and the Half-Blood Prince (Book 6) - J. K. Rowling \$17.99 · Quantity: 1 · Not yet published Condition: new <input checked="" type="checkbox"/> Gift options None <input type="checkbox"/>	
Payment Method: <input checked="" type="checkbox"/> Check Visa: **** Exp: ****	
Billing Address: <input type="checkbox"/> Change Jenifer Tidwell 77 Massachusetts Ave Cambridge, MA 02139-4303 United States	

Review the information above, then click "Place your order."

Amazon的订单在多个页面的购买过程之后，也就是说，在选定了购买的物品，设定了发货方式和付款细节之后，Amazon提供了一个小结页面。这个页面的格式非常紧凑，而且容易阅读。它的内容组织成了几个带标题的栏目（参见第4章），而且很多信息附近都有一个方便的“Change”（修改）按钮。那几个“Place your order”（下单）的按钮（你可能注意到了，这样的按钮有两个，其中一个是为了那些太着急而不想阅读整个页面的人准备的）很清楚、很大，采用了明亮的颜色——因此很容易找到。

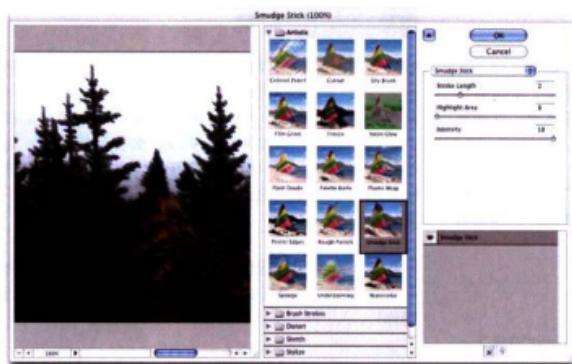


图 1-15 Photoshop 的 Image 滤镜都很多很复杂，也需要花一定的时间才能把滤镜应用到大的数码图片上。用户可能在选择正好适合自己想要效果的滤镜之前，需要试用很多类似的滤镜。而且他想在浏览滤镜库的时候快速对滤镜进行选择，这就是个典型的预览场景。

用户从这个屏幕截图的中央面板上选择滤镜；预览的效果出现在左边，显示的是用户图片的一小部分。一个和原来图片无关的小样本图片代表了每个滤镜，这样在某种程度上，用户实际上看到了两个层次上的预览——一个是选择滤镜，另一个是应用到真实图片上的实际效果（参见第7章来了解图示选项_{Illustrated Choices}模式）。

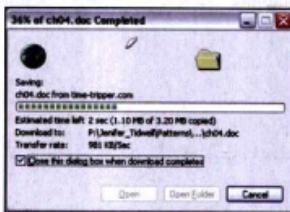


图5-16 IE的进度提示对话框

它是什么

在那些需要很长时间才能完成的操作中为用户显示该操作当前的进度。

什么时候使用

如果有一个长时间操作打断当前界面，或需要在背后运行，就可以应用这个模式了。多少时间算长呢？超过两秒就算。

为什么使用

如果界面一动也不动，用户会失去耐心。哪怕是你已经把鼠标光标改变成一个时钟或沙漏（在任何时候，如果UI的其他部分不能操作，你都应该这么做），你也不希望让用户在等的时候根本不知道要等多久。

试验显示，如果用户看到一个指示器，表示某个动作正在进行，他们会有耐心得多。哪怕这样他们也比没有指示器的情况下需要等待更长时间。可能是因为他们知道“系统正在思考”，而不是挂起了或者在等待他们做点什么后才会继续进行。

如何使用

显示一个动态的指示器，指示当前已经有了多少进展。文字或者图片（或者两者兼而有之）都可以，用这样的指示器来告诉用户：

- 当前正在做什么
- 已经完成了多少比例
- 还需要多少时间才能完成
- 如何停止这项任务

尽管用户会关心时间统计，但有时候错误的时间统计也没有关系，只要你很快回到准确的时间就可以了。但是有时候界面根本就不知道需要多长时间。在那样的情况下，无论如何也应该显示某些指示百分比的动画。想想浏览器的图片环，它在页面载入的时候一直是转动的。

很多GUI工具箱提供了实现这个模式的组件或对话框，就像Java Swing的iProgressbar。不过，要注意潜在的线程问题——进度指示器在刷新的时候必须有一致性。

而操作本身的进展可能是没有规律的。如果可以，让UI的其他部分保持反应，不要在能看到进度指示器的时候锁住界面。

如果有可能取消这个正在进行的操作，那么在进度指

示器上或它的附近提供一个取消按钮或类似设备；那可能是用户寻找取消功能的地方。参见可取消性（**Cancelability**）模式（下一个模式）来获得更多信息。

示例



这个启动屏幕使用了图标来表示KDE在启动的时候做了些什么——当到达每个步骤的时候，相应的图标就会变得清晰。在这里没有使用计时的数字，但它们没有什么必要（在KDE启动结束之前，用户已经被它迷住了），而且，不管怎么说，很难准确计时。



图5-18 Firefox的停止载入按钮

它是什么

提供某种方式快速取消一个耗费时间的操作，而没有什么负面影响。

什么时候使用

当一个耗费时间的操作会打断当前的用户界面，或者只在后台运行，但这个操作需要的时间超过两秒时，就可以应用这个模式。例如打印文件、查询数据库、加载大文件等。还有一种情况也可以应用这种模式，那就是用户正在忙于某个操作，该操作关闭了其他很多交互（例如打开一个模态对话框）的情况。

为什么使用

用户会改变主意。一旦一个耗费时间的操作开始了，用户可能会想要停止它，特别是如果进度提示（Progress Indicator）告诉她还需要一段时间的时候。或者当时用户可能只是不小心开始了它。可取消性当然可以帮助防止错误的发生和从错误中恢复——用户可以取消某个她明知道会失败的操作，例如从她知道已经宕机的某个网站服务器上载入页面。

如果知道任何操作都可以取消，用户可能会更放心地在界面上探索，试试各种功能。因此，它鼓励了安全探索（Safe Exploration）模式（见第1章），而它也反过来让界面更容易使用，更有趣。

如何使用

首先，找出某种方式来让这个耗费时间的操作加快速度，让它看起来更快一些。可能并不一定要真的加快速度，如果用户认为它加快了，就已经够好了。在Web或网络应用上，这可能意味着预先载入数据或代码——在客户端请求之前就发送给它——或者以一种递增的方式发送数据，然后一边接收一边显示给用户。记住，人们阅读也需要时间。你也可以使用载入时间来让用户阅读第一个页面所需要的数据，然后再让用户读下一页，依此类推。

不过，如果你确实需要取消它，那么下面就是取消的方法。把一个取消按钮直接放在界面上，靠近进度提示的旁边（你正在使用它，不是吗？）或任何显示操作结果的地方。用停止（Stop）或取消（Cancel）作为标签，或者只放/同时放一个国际通用的停止图标在上面：一个红色的八角形，一个红色的圆圈加上一个水平方向的横杠，或者一个叉（X）。

当用户单击或按下取消按钮的时候，立刻取消这个操作。如果此时等待的时间太长，长度超过一秒甚至两秒，用户可能会怀疑取消按钮是不是有用（或者你在阻止他取消，因为他可能也在等待操作结束）。告诉用户取消已经起作用了——例如停止进度提示或在页面上显示状态信息。

如果有多个并行的操作同时进行，就有一定的挑战性了。用户怎样才能取消其中一个而不是其他的操作呢？那么，取消按钮的标签或工具提示可以明确地告诉用户，如果它按下的时候取消的是哪个操作（参见

智能菜单项模式来了解类似的概念）。如果这些操作以一个列表或一组面板的方式显示，你可以考虑为每个操作提供一个单独的取消按钮，来减少歧义。

示例

如果你曾经试过取消一次复印作业——也许你不小心请求复印600份文档——通常会很清楚如何操作。在复印机面板上有一个巨大的停止或取消按钮。你按一下它，复印机就立刻在复印完当前这个页面之后停下来。

不过，在Windows里，如果你发送了一份文档到打印机，然后需要取消的时候（也许你不小心打了600

页），你得首先打开打印机的属性窗口。人们并不是很容易就能知道怎样打开它，虽然实际上有好几种方法都可以做到。打开了打印机属性窗口以后，界面上并没有取消按钮。你得使用菜单条或弹出菜单（图5-19有显示）来取消这次打印作业。然后有一个确认对话框会询问你是否真的要取消。因此，这样需要好多次单击，以及一个不太明显的路径来取消一次打印。

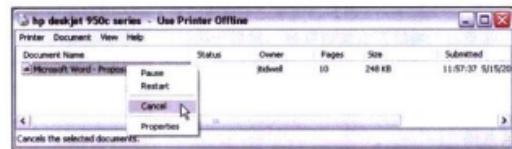


图5-19 Windows XP的打印机窗口，它有一个难以找到的取消操作，别这么做。

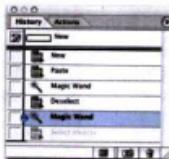


图5-20 Photoshop提供了一种对“撤销堆栈”的可视化展示

它是什么

提供一种方式可以很容易撤销用户执行的一系列操作。

什么时候使用

如果你已经设计了一个高度交互式的用户界面，相比简单的导航和填写表格来说，它要复杂得多。这样的界面包括：邮件阅读器、数据库软件、创作工具、图形软件、编程环境等。

为什么使用

能够撤销一长串操作的能力让用户觉得可以对这样的界面进行安全探索。在他们学习界面的时候，可以在上面进行试验，并且确信在试验的时候不会造成某些不可取消的修改——哪怕只是不小心做了什么不合适的操作。这一点对于各个层次上的用户都适用，而不仅仅是初级用户¹。

一旦用户对界面了解更多了，她就可以带着自信继续操作，而不必担心会造成永久性的错误。如果她的手指滑了一下，单击了错误的菜单项，不需要进行复杂而压力巨大的恢复，她不需要去一点点修复已经保存

的文件，不需要关机再重新启动，或者寻求系统管理员的帮助来恢复一个备份文件。这可以帮助用户免去时间上的浪费和某些精神困苦之扰。

多级撤销也让专家用户更快更容易地探索工作路径。举例而言，一名Photoshop用户可能会在一个图片上执行一系列的滤镜操作，研究那些结果看是不是她喜欢的，然后再——撤销，回到起点。然后她也可能尝试另外一组滤镜，也许会保存图片，然后再次撤销。在没有多级撤销机制的情况下，她也可以这么做，但是需要更多的时间（用来关闭和重新载入图片）。当用户在进行一些创造性的工作时，速度和容易使用是用来保持注意力集中，以及心流体验的两个重要因素。参见第1章阅读更多相关内容，特别是安全探索和递增构建这两个模式。

如何使用

可以撤销的操作

UNDOABLE OPERATIONS

你的软件首先需要对操作进行建模，是什么样的操作——它叫什么，它和什么对象有关，怎样反转它的效果。然后你才能为它建立一个撤销的界面。

¹ Alan Cooper和Robert Reimann在*About Face 2.0: The Essentials of Interaction Design* (“Wiley”出版)一书中花了整整一章的篇幅来描述“撤销”这个概念。

你得决定哪些操作需要能被撤销。任何可能改变文件或任何永久性内容的操作都应该可以撤销，而暂时的操作或与视图相关状态通常不需要撤销。具体而言，在大部分应用里，人们期望能撤销下面这些改变：

- 文档或电子表单的文字输入
- 数据库事务
- 图片或画布的修改
- 布局上的变化——位置、大小、顺序或图形应用中的分组
- 文件操作，例如删除或修改
- 对象的创建、删除和重新组织，例如邮件消息或电子表单的列
- 任何剪切、复制、粘贴操作

下面这些修改基本上不可撤销。即使你想多付出一些努力让它们可以撤销，也要考虑一下。如果让一些无用的撤销操作充斥在撤销堆栈里，用户会不会觉得很难堪。

- 文本或对象选择
- 窗口或页面之间的导航
- 鼠标光标和文本光标的定位
- 滚动条的位置
- 窗口或面板的位置和尺寸
- 在一个未提交的对话框或模态对话框上的改动

有些操作处在两者之间。例如表单填写，有时候可以撤销，有时候不可以。不过，如果用Tab键移出某个区域的时候自动提交了该区域的变化，那么可能有必要使它可以撤销。（有相当一些操作不可能撤销，不过应用软件通常会让用户在尝试之前清楚地了解那一点。不可能实现的撤销包括一次电子商务的购买步

骤，往公共栏上或聊天室里发布消息，或者发出一封邮件——似乎有时候我们确实想撤销这样的操作！）

不管怎么样，确认那些可以撤销的操作对用户来说是有意义的。要确认是从用户的角度而不是从计算机的角度来定义和命名那些撤销操作。例如，你应该以一次一堆单词的形式，而不是一个字母一个字母地撤销输入的文本。

设计撤销堆栈

DESIGN AN UNDO STACK

每个操作按照它执行的顺序放在堆栈的顶部，然后每个反转这个操作的撤销也在顶部，接下来是第二个、第三个。重复操作（Redo）也以类似的方式组织作为这个堆栈的补充。

这个堆栈要可用，至少要有10到12个项目，或者更多的项目——如果你能管理的话。长期的观察或可用性测试也许会告诉你，可以使用的数量限制是多少（Constantine和Lockwood认为，超过12个项目通常是没有必要的，因为“用户几乎不会用到这么多层次的撤销”²。高度密集型应用的专家用户们可能有一些不同的说法。因此，跟以往一样，要了解你的用户）。

展现方式

PRESENTATION

最后，决定以哪种方式把撤销堆栈展现给用户。大部分桌面应用会把撤销/重复的菜单项放在编辑菜单底下。还有，撤销通常关联到Ctrl+Z或差不多的快捷键。行为良好的应用会使用智能菜单项来确切地告诉用户撤销堆栈上的下一个操作是什么。

2. 参见Larry Constantine和Lucy Lockwood的文章“Instructive Interaction: Making Innovative Interfaces Self-Teaching”（指示性的交互——让创新的界面具有自我教授能力），网址为<http://foruse.com/articles/instructive.htm>。

但是从这个模式顶上的屏幕截图（图5-20），你可以看到一种不太一样的、更加可视化的展示方式。Photoshop显示了一个可撤销操作的滚动列表——包括那些已经撤销过的（可以看到其中一个，用灰色显示的）操作。它让用户选择堆栈中的某个点，从这一点

起往上的操作是需要撤销的。一个像这样的可视化命令历史会很有帮助，哪怕只是作为一个你最近动作的提醒也好。参见命令历史（Command History）模式了解更多相关信息。

示例

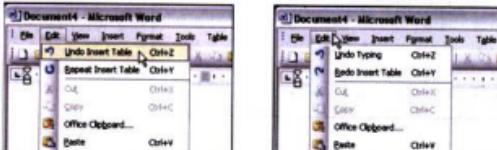


图5-21 这是一个更加典型的多级撤销展示方式。在这个例子中，用户输入了一些文字，然后插入一张表格。第一个撤销删除了表格，一旦这个撤销完成了，撤销堆栈里的下一个动作——是撤销那些输入的文字。再次调用撤销将删除那些文字。同时，用户也有机会利用“Redo（重做）”菜单项“撤销刚才的撤销”。如果我们位于堆栈的顶部（就像第一个屏幕截图上那样），就没有重做的菜单项，那个菜单项的标题是一个重复动作。

有点糊涂了？肯定是。绝大多数用户永远不会对这里使用的算法建立一个清楚的心智模型：很多人不知道“堆栈”是什么，更不用说它怎样用来重置和重做了。那也是为什么这里智能菜单项（Smart Menu Items）对可用性测试来说很关键的缘故。它们清楚地解释了将要发生什么，这样就减少了用户的认知负担。

命令历史

Command History

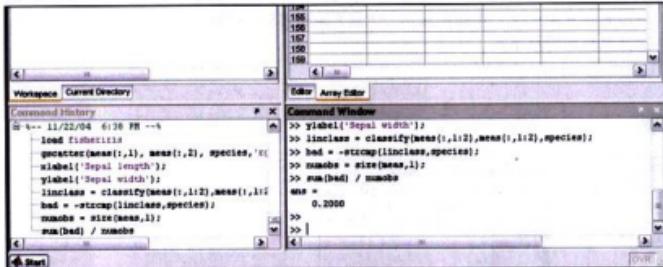


图5-22 Matlab的命令历史，显示在左下角

它是什么

当用户执行某些操作时，保存一份可见的记录，记录做了什么操作、作用在什么对象上，以及执行操作的时间。

什么时候使用

当用户执行一系列冗长而且复杂操作时，不管是通过界面还是命令行的方式，很多用户都有经验，或者如果没有，至少他们也希望有一个效率较高的界面可以支持这种时间很长而且经常重复的工作。图形编辑器和编程环境通常是这个模式的优秀候选者。

为什么使用

有时候，在操作软件的过程中，用户需要记住或回顾他刚才做了什么，例如，他可能想做下面的任何一项操作：

- 重复某个早些时候所做的操作或者命令。他对那

一个命令记得不是特别清楚了。

- 回忆前面某些操作的顺序
 - 重复一系列操作，那些操作原先是作用在一个对象上的，现在需要作用在另一个对象上
 - 对他的操作进行记录，不管是出于法律还是安全的理由
 - 把一系列交互式的命令转换成一份脚本或一个宏命令（参见本章的宏模式）

计算机很擅长把已经进行的操作步骤准确地记录下来。人不擅长这样做。好好利用这一点。

如何使用

为用户所做的操作保持一份连续的列表。如果界面是命令行方式的，就很容易了——只要把任何输入的命令记下来就可以了。参见图 5-22。如果可以的话，应该在多个 Session 之间记录这些历史。这样用户就可以看到一个星期或者更长时间以前自己做了些什么。

如果是一个图形界面，或者图形界面和命令行界面的结合，那么就稍微复杂了一点。找到一种方式来连续、一致地表达这些操作，通常是使用文本（虽然没有理由为什么不能以图形的方式来记录）。确定你已经以合适的粒度来定义这些操作——把一个作用在17个对象上的操作记录成一个操作，而不是17个。

哪些命令应该记录，而哪些不需要呢？参见多级撤销（**Multi-Level Undo**）模式来看这方面的讨论。如果一个命令可以撤销，那么应该记录在历史操作里。

最后，把这份操作历史显示给用户。在大部分软件中，这样的显示应该是可选项，因为它在用户的工作中可能总是扮演一个支持性的角色，而不是主要角色。命令列表——从最老的命令到最新的命令——就很合适。如果你喜欢，也可以给这些历史操作的显示加上时间戳。在图5-22中所示的界面，就显示出无论什么时候重启这个程序，它都会在历史记录里加上日期和时间。

示例

```
swing-admin:/var/log/httpd> history
1 23:55  psud
2 23:55  putenv /var/log/httpd/
3 23:56  (a -l .) >> log
4 23:56  tail -1000 access.log | grep "index"
5 23:56  tail -1000 access.log | grep "index" | wc
6 23:57  tail -1000 access.log | grep "index" | wc
7 23:57  ls -l
8 23:57  ls -l /usr/share/
9 23:57  tail -1000 access.log | grep "index" | wc
10 23:58  tail -1000 access.log | more
11 23:58  tail -1000 access.log | grep "google"
12 23:58  tail -1000 access.log | grep "google"
13 23:58  tail -1000 access.log | grep "google"
14 23:59  tail -1000 access.log | grep "googlebot"
15 23:59  history
swing-admin:/var/log/httpd>
```

图5-23 Unix和它的许多版本使用Shell程序，例如Tcsh和Bash，会把它们自己的命令历史保存到文件里面。用户可以通过“History”命令把它调出来，如上图所示。这份历史记录也可以通过很多命令行结构来访问，例如“!!”（重复调用上一个命令），“!3”（重复调用三个命令之前的那个命令），还有用Ctrl+P，可以用来一次重复显示前面调用的一批命令。



图5-24 Photoshop的撤销堆栈，在多级撤销模式中也看到过，它就是一份有效的命令历史。你可以用它来撤销之前所做的操作，但并不是一定要这样；你也可以只是查看一些，然后上下滚动，来回顾一下自己做过些什么。它使用了图标来识别不同种类的操作，这一点很少见，但是很不错。

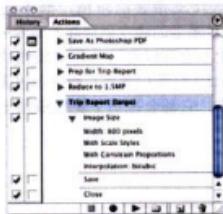


图5-25 Photoshop的动作列表

它是什么

宏是一个操作，但是在该操作里组合了其他一些小的操作。用户可以通过把一系列操作放在一起创建一个宏命令。

什么时候使用

用户想重复一长列操作或命令。他们可能想要把这些操作或命令重复作用在文件列表、图片列表、数据库记录或其他对象上，例如对每个对象进行相同的操作。你可能已经实现了多级撤销或命令历史这两个模式。

为什么使用

没有人想要一次又一次地重复执行同样一批交互式任务。这正是计算机应该擅长的。第1章描述了用户的行为模式，其中有一个叫做简化重复工作（**Streamlined Repetition**），宏就是可以为它提供很好支持的那种机制。

显然，使用宏命令可以帮助用户加快工作。不过，通过减少命令个数和需要进行的动作，他们也减少了可能的手指滑动、疏忽和遗漏，以及类似的错误。

你也许还记得“心流”（flow）的概念，也是在第1章提到过的。当用户可以把一长串操作压缩到一个命令或

一个快捷键的时候，这种心流的体验增强了——用户可以在更短的时间内，花更少的精力来完成更多的工作，而且她也可以一直保持更高层的目标而不会陷入各种细节中。

怎样使用

任何时候，为用户提供某种方式来记录一系列操作，并且可以很容易地“回放它们”。这种回放应该像一个简单命令，一个按钮或对象的一次拖拽那么容易。

宏的定义**DEFINING THE MACRO**

用户应该可以按自己的想法给宏命名。让她可以在某种程度上回顾这个操作序列，因此她可以检查自己所做的工作，或者重新查看一个已经忘记的操作列表来看它到底做了些什么（和命令历史模式里一样）。让宏和宏之间可以彼此引用，这样的话它们可以建立在其他的宏上。

用户会想保存这些宏，以备下次使用，因此确认它们是持久的——把它们保存到文件或数据库。把它们显示在一个可以搜索、排序甚至分类的列表里，具体怎么做按用户的需要而定。

宏的执行

RUNNING THE MACRO

宏本身可以回放，来让操作更简单。或者如果它作用在某个对象上，而这个对象可以从一种调用方式转换到另一种调用方式，那么你可以允许这些操作带参数（例如，使用一个占位符或者变量而不只是一个对象本身）。宏也应该能一次作用在多个对象上。

这些宏的名字（或者调用它们的控件）如何展示基本上取决于该应用的本质，不过需要考虑以内建的操作

来显示它们而不是使其处于次要地位。

记录这些操作顺序的能力，再加上宏与宏彼此引用的能力，可以让用户有机会发明一些全新的语言或可视化的语法——一种很好地协调他们自己工作环境和工作习惯的语法。这是一种非常强大的能力。事实上，这就是编程；但是，如果你的用户并不认为他们自己是程序员，那么不要把它叫做编程，否则你会把他们吓走（“我根本就不会编程，我没法完成这样的工作”）。

示例

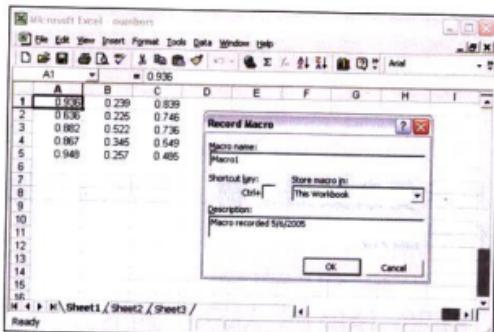


图5-26 Microsoft的Excel允许对宏进行记录、命名，在整个文档里保存，甚至分配一个快捷键。用户也可以选择从工具条上的一个按钮，或者使用文档内部的一个Active X控件来运行（这意味着你可能把它们作为按钮或文本输入框的回调函数）。

```
Sub Test()
    ' Test Macro
    ' Macro recorded 5/3/2008

    Selection.Insert Shift:=xlDown
    Range("A1").Select
    Selection.Sort Key1:=Range("C1"), Order1:=xlDescending, Header:=xlGuess, _
        OrderCustom:=1, MatchCase:=False, Orientation:=xlTopToBottom
End Sub
```

图5-27 这些宏命令是用Visual Basic写出来的。用户可以在想要的时候手工编辑它们。它这就变成了编程的一种情况。因为Visual Basic提供了对那么多常规功能的访问——绝大部分都和电子表单的操作无关——宏对于Office应用来说是一个大的风险。通过严格约束宏能执行的功能，以及限制用户运行宏的方式（例如单击工具条按钮），你可以通过降低功能来换取安全性。

06

显示复杂数据：树、表格及其他信息图形

**SHOWING COMPLEX DATA:
TREES, TABLES, AND OTHER
INFORMATION GRAPHICS**



信息图形——包括地图、表格和图表等——以可视化的方式而不是文字的方式传达知识。如果设计得好，它们可以让人们用眼睛和心灵来得出他们自己的结论，它们“显示”(show)，而不是“告诉”(tell)。

这些图形是我最喜欢的那一类界面。不过，不好的工具或不充分的设计会大大削弱这些界面的能力。而且有很多信息丰富的界面确实没有达到它们应有的水平。

本章的模式可以帮助你在最大程度上使用这些工具，并为你介绍一些可视化设计上有用也好玩的创新。接下来的介绍部分的一些想法可以帮你看出：一个给定的界面中，哪些设计因素对你来说是最重要的。

信息图形基础

THE BASICS OF INFORMATION GRAPHICS

信息图形的意思，简单地说，就是带着给用户传递数据的目标，对数据进行可视化展示。在这个定义里也包括了表格和树型视图，因为它们天生就是可视化的。尽管它们主要是显示文本而不是线段和多边形。其他熟悉的静态信息图形包括地图、流程图、条状图，还有真实对象的图表。

但是我们现在面对的是计算机，而不是纸。因为它所具有的交互性，你可以让几乎所有的静态设计变得更好。交互式的工具让用户按她的需要显示或隐藏数据，而且，当用户选择怎样查看和探索界面的时候，它们会把她放在“司机的位置”。

在交互式的图形里，就算对数据进行简单地控制和重新组织这样的行为也有价值——用户成为了发现过程里的参与者，而不只是一个被动的旁观者。这一点意义重大。用户可能不会创造出世界上设计得最好的图形或表格，但是操作这个图或表格的过程就会让她面对面地接触这些数据，如果是在纸上，她可能根本意识不到这些。

最终，用户使用信息图形的目标是学习一些知识，但是设计师需要理解用户想要学什么。用户可能在寻找什么特定的东西，例如地图上一条具体的街道，在这种情况下她要能找到它——就是说通过直接搜索或根据额外的信息把它筛选出来。她需要能得到的“总体视图”只要详细到能找到那个具体的信息就够了。搜索、过滤和获得细节，这些很关键。

另一方面，她也可能要学习一些不太具体的东西。她可能看一眼地图来获得城市的分布情况，而不是找到一个具体的地点。或者她可能是一个科学家，在可视化一个生物化学过程，试图理解它的工作原理。现在总体视图很重要，她需要了解部分是怎样连接到整体的。她可能想要放大，再缩小，偶尔看看详细信息，在数据的各个视图之间进行比较。

良好的交互式信息图形为用户提供下面这些问题的答案：

- 数据是如何组织的？
- 它们之间的关系如何？
- 我能怎样进一步了解这些数据？
- 我可以重新组织这些数据来换一种方式看待它吗？
- 只要把我想知道的数据显示给我就可以了。
- 具体的数据值是多少？

在本节里，请记住术语“信息图形”（Information graphics）是一个很大的范围。它包括图、图形、地图、表格、树、时间线，以及各种各样的图表。数据可能很多、有很多层次，又或者很少、很集中。这类技术可以很好地支持很多你都想不到的图形类别。

在描述这些模式本身之前，让我们先来讨论一下上面提到的一些问题。

组织模型：数据是如何组织的

ORGANIZATIONAL MODELS: HOW IS THIS DATA ORGANIZED

在信息可视化过程中，用户首先看到的是你为数据选择的外观。理想的情况下，数据本身有一个内在的结构，已经为你建议了它的外观。那么，哪些模型最适合你的数据？

模型	图表	常见的图形
线性模型		列表或单变量图表
表格		电子表格、多栏列表、可排序的表格 (Sortable Table)、多Y值图表 (Multi-Y Plot)，或者其他多变量图表
层级模型		树、级联列表、树表、树形图、有向图
网络模型（或组织模型）		有向图或流程图
地理模型（或空间模型）		地图或示意图
其他模型		各种不同的图表，例如平行坐标图，或者树状地图 (Treemaps)

你可以替你想要显示的数据选择这些模型。如果两个或更多的模型都合适，想想哪一个可以体现数据的外观。如果你的数据可能既有地理特征又有表格特征，例如，如果只用表格来显示，那么可能会削弱它的图形天性——如果没有同时也以地图的形式显示它们，一个用户可能会错过数据里的一些有趣特性或关系。

前摄的变量：谁和谁有关系

PREATTENTIVE VARIABLES: WHAT'S RELATED TO WHAT

你选择的组织模型讲述了很多关于数据外观的信息。这些信息的一部分在潜意识下起作用：人们认识树、表格和地图，他们会在还没有下意识思考之前，立即得出关于背后数据的结论。但并不是数据的外观就做到了这些。每个数据元素的外观也会在潜意识的情况下起作用：看起来很像的东西可能彼此相关联。

如果已经阅读了第4章，那么这一点听起来有点熟悉——你已经了解了格式塔原则（如果是在跳着翻阅这本书，那么也许你应该回到前面去看看第4章的介绍部分）。大部分原则特别是相似性和连续性，也在这里起作用。我会多花一些篇幅来介绍它们是怎样运作的。

一些可视化特性是前摄的。在用户有意识地注意它们之前，它们就已经开始在传达信息了。我们来看看图6-1，找到那些蓝色的对象。

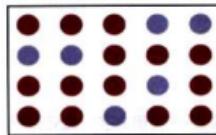


图6-1 找到蓝色的对象

我想你可以很快完成这个任务。现在看看图6-2，同样找到那些蓝色的对象。

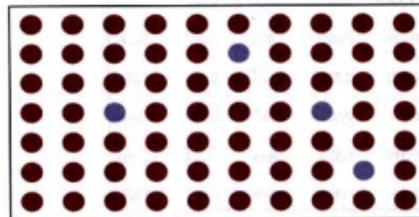


图6-2 再次找到蓝色的对象

你还是很快完成了任务，对吗？事实上，不管有多少个红色的对象都没关系：你花在寻找蓝色对象上的时间是固定的。你可能认为它会随着总数量而线性增长——以算法术语来说，需要 $T(n)$ ——不过不是这样的。颜色作用在初级认知层次上。你的视觉系统为你完成了那些复杂的工作，而且它似乎是以一种“大规模并行”的方式进行处理的。

换句话说，视觉上没有变化的文字会迫使你阅读它们的含义并花费脑力去思考。图6-3就正好显示了这个问题。它用的是数字而不是颜色。要找到比1大的数字，你能有多快？

0.103	0.176	0.387	0.300	0.379	0.276	0.179	0.321	0.192	0.250
0.333	0.384	0.564	0.587	0.857	1.064	0.698	0.621	0.232	0.316
0.421	0.309	0.654	0.729	0.228	0.529	0.832	0.935	0.452	0.426
0.266	0.750	1.056	0.936	0.911	0.820	0.723	1.201	0.935	0.819
0.225	0.326	0.643	0.337	0.721	0.837	0.682	0.987	0.984	0.849
0.187	0.586	0.529	0.340	0.829	0.835	0.873	0.945	1.103	0.710
0.153	0.485	0.560	0.428	0.628	0.335	0.956	0.879	0.699	0.424

图6-3 找到大于1的数字

当处理这样的文字时，你的“搜索时间”确实是随项目数量线性增长的。如果还使用文字，但让目标文字比其他文字大一些，就像图6-4那样，会怎么样？

0.103	0.176	0.387	0.300	0.379	0.276	0.179	0.321	0.192	0.250
0.333	0.384	0.564	0.587	0.857	1.064	0.698	0.621	0.232	0.316
0.421	0.309	0.654	0.729	0.228	0.529	0.832	0.935	0.452	0.426
0.266	0.750	1.056	0.936	0.911	0.820	0.723	1.201	0.935	0.819
0.225	0.326	0.643	0.337	0.721	0.837	0.682	0.987	0.984	0.849
0.187	0.586	0.529	0.340	0.829	0.835	0.873	0.945	1.103	0.710
0.153	0.485	0.560	0.428	0.628	0.335	0.956	0.879	0.699	0.424

图6-4 再来一次

现在我们又回到了固定时间。实际上，大小是另一个我们会前摄的变量。大一些的数字向右边突出，这个事实也有助于你找到它们——对齐是另一个前摄的变量。

图6-5显示了很多已知的前摄变量。

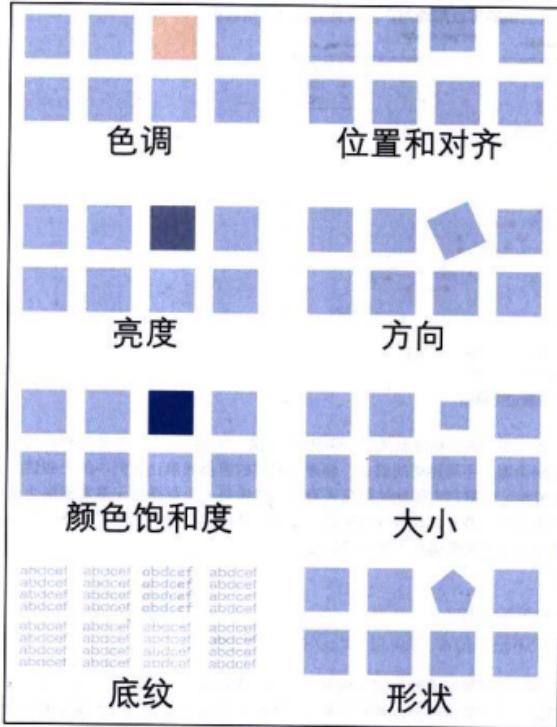


图6-5 八个前摄变量

前摄变量的概念对基于文字的信息图形有着深远的含义，就像图6-3中所示的数字表格一样。如果你希望某些数据比其他数据更突出，就需要让它们看起来和其他数据不同：不同的颜色、大小，或者在一些其他的前摄变量上不一样。更一般地说，你可以使用这些变量来区分任何信息图形上的数据类别或维度。这样的做法有时候也叫做“编码”。

当你必须为多维数据画图的时候，可以使用几个不同的视觉变量来编码所有这些维度，达到单一的静态显示要求。回头想想图6-6中所示的散点图，点的位置是通过X和Y轴来体现的；然后用颜色对第三个变量进行了编码。散点标志的外形还可以对第四个变量进行编码，但是在这个例子里，形状

和颜色互相冗余了。这样的冗余编码可以帮助用户从视觉上区分三组数据。

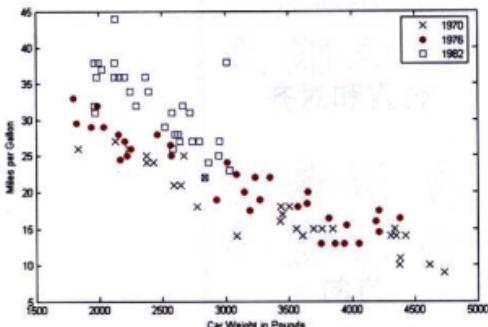


图6-6 在一幅散点图中对三个变量进行编码

用前摄因素编码和一个更普遍的图形设计概念有关系，那就是“分层”（Layering）。当你看到任何设计良好的图形时，就会感觉到页面上不同种类的信息。像颜色那样的前摄因素让它们中的一些信息从页面上“跳出来”了，而相似性让你把它们彼此关联起来看待，仿佛每一个都是位于基本图形之上一些透明的图层那样。这一点在分割数据的时候尤其有效——每个图层都比整个图形要简单，用户可以挨个研究它们，但是整体之间的关系得到了保持和加强。

导航和浏览：怎样对这些数据进行探索

NAVIGATION AND BROWSING: HOW CAN I EXPLORE THIS DATA

用户一开始可能会通过浏览对交互式的数据图形进行研究——只是看看这儿有些什么内容。他也可能继续深入浏览来找到某些正在寻找的东西。过滤和搜索也可以达到同样的目的。但是通过某个数据集上的“虚拟空间”进行导航常常更好一些。空间记忆（**Spatial Memory**, 第1章）在这里起作用了，用户可以看到一些有趣的地方同时和其他数据在一个上下文里。

在信息可视化领域有一个著名的说法：“焦点加上下文”（Focus Plus Context）。一个好的可视化界面应该允许用户把注意力集中在一个兴趣点上，同时在它周围显示足够的素材，来让用户在一个大的背景下得到一种当时身在何处的位置感。

下面是一些常规的导航和浏览技术：

滚动和平移

如果数据不能一次显示在屏幕上，你可以把它放到一个滚动窗口里，让用户可以用某种很容易也很熟悉的方式来访问屏幕以外的部分。滚动条几乎对每个人来说都很熟悉，也容易使用。不

过。有些显示区域实在太大了，或者它们的大小并不确定（这就会让滚动条不准确）。又或者它们有一些数据在窗口之外，你需要重新获取/计算（这样的话滚动条的反应太慢了）。在那些情况下，不要使用滚动条，相反，试试按钮，让用户单击它们来得到下一屏数据：想想Mapquest和Mapblast是怎么处理的。其他一些应用采用了平移的方法，在这种方法里，鼠标光标抓住并一直拖动信息图形，直到找到了当前感兴趣的点为止，就像Google Map那样。

这些技术适用于不同的情形，不过基本概念是一样的：交互式地移动图形的可见部分。有时候总览加细节（Overview Plus Detail）模式可以帮助用户保持方向感。也可以用一个矩形加上整个图形的一小块视图来显示，这个矩形代表整个图形上当前可见的视野。用户可以通过拖动这个矩形来进行平移，作为滚动条或其他方法的附加手段。

缩放

缩放改变了可见部分的范围，而滚动改变的是显示的位置。当你展示一份数据密集的地图或图表时，考虑一下为用户在那些兴趣点上提供缩放的能力。它也意味着你不需要把每个数据细节都放到整体视图里面——如果你有大量标签，或者很多细小的特性（特别是在地图上），那可能根本就没法全显示出来。如果放大以后有了足够的空间，当用户放大的时候，那些特性就可以显示出来了。

很多缩放都是以鼠标单击或按钮按下的方式实现的，同时整个视野也会立刻改变它所显示的范围。但是那并不是缩放的唯一方式。一些应用在用户的鼠标光标移过图形的时候，创建了非线性的信息图形变化：在鼠标光标之下的任何内容都变大了，但是离光标较远的地方不会有变化。参见局部缩放（Local Zooming）模式来得到更多信息。

打开和关闭兴趣点

树形视图通常会让用户任意打开或关闭父节点对象，所以他们可以检查这些对象的内容。一些层级结构的表、图形也为用户提供打开和关闭当前部分内容的机会，而不用打开一个新窗口或去一个新屏幕。有了这些机制，用户可以很容易地对包含关系或父/子关系进行探索，而不用离开当前窗口。级联列表（Cascading Lists）模式也描述了另一种探索层级机构的方式：它通过一个单击打开和关闭对象达到自己的效果。

深入到兴趣点内部

有些信息图形只提供了顶级信息。用户可能会在地图上单击或双击来查看她刚才单击的城市，也可能单击一份图表的关键点来看看更详细的下级图表。这种“深入”可能会重用当前同一个窗口，也可能使用同一窗口的另一个面板，或者打开一个新面板/窗口（参见第2章来查看关于窗口机制的讨论）。除了这些查看发生在当前图形之外（它们并没有集成在一起），这项技术重新组织了兴趣点的打开和关闭。

如果你也为一份交互式的信息图形提供搜索功能，可以考虑把搜索结果和上面提到的技术结合起来。换句话说，当用户在地图上搜索悉尼这个城市的时候，可以使用地图缩放和/或平移来到达目的地。这样，搜索的用户就得到了一些上下文及空间记忆的好处。

排序和重新排列：我可以重新排列这份数据来以一种不同的方式查看它吗

SORTING AND REARRANGEMENT: CAN I REARRANGE THIS DATA TO SEE IT DIFFERENTLY

有时候重新排列一份信息图形可以体现出意想不到的关系。看看下面的图形，这是来自NCI（National Cancer Institute，联邦癌症研究所）的死亡率分布图（见图6-7）。这张图显示了德克萨斯州肺癌的死亡数据，德克萨斯州的大城市以字母顺序排序——如果你在寻找具体的城市，这不能不算一种有意义的排序方式。但是，正如它所展示的那样，数据并没有引出一些有趣的问题。例如，为什么在Abilene、Alice、Amarillo和Austin都似乎有着差不多的死亡率，从图上看并不清楚，可能只是偶然。

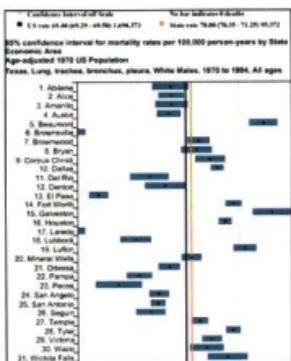


图6-7 来自`http://cancer.gov/atlasplus/`，根据字母顺序排列

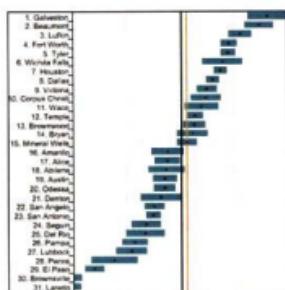


图6-8 同样的图表，根据数字排列

但是这个Web应用可以让你重新对这些数据排序，如图6-8所示，以数字递减的顺序。图形一下子就变得有意思多了。Galveston排在第一位——为什么，它的邻居城市Houston的数据要低很多呢？Galveston有什么特别的地方？（OK，你需要一些关于德克萨斯州的地理知识来提出这些问题。不过，你知道我是什么意思。）与此类似的是，为什么在这两个邻居城市Dallas和Fort Worth之间也不一样呢？还有，很明显，和墨西哥接壤的南部城市El Paso、Brownsville和Laredo比其他城市的肺癌情况要少很多，为什么会这样呢？

人们可以用这种方式和数据图形交互，并且有机会从图形中了解到更多知识。排序和重新排列数据让不同的数据点彼此靠近，从而让用户进行各种不同的比较——对邻近数据进行比较要比对散落的各个点进行比较容易得多。而且用户也容易关注那些分布的极端情况，就像我在这个例子上所做的一样。

你还能怎样应用这个概念呢？可排序表格模式讲述的是一个显而易见的方式：当你有一个多列表格的时候，用户可能想要根据他们选择的列对各行数据进行排序。这个模式很常见。

(很多表格的实现方式也允许通过拖拽对它们自己的数据列进行重新排序。) 树可以对它们的子节点重新排序。图表和连接图可能会允许在它们的元素之间拥有空间关系，同时保持它们的连通性。好好想像一下！

可以考虑下面这些排序和重新排列的方法：

- 字母顺序
- 数字顺序
- 日期或时间顺序
- 物理地点顺序
- 根据类别或标记排序
- 受欢迎程度——用得很多的和用得很少的
- 用户设计的排列方法
- 完全随机排列（你永远不会知道自己将看到什么）

这里有一个微妙的例子，如图6-9所示。在每个条上显示了很多数据值的柱状图（堆叠式柱状图）。可能也可以做到重新排列——最靠近基线的竖条是最容易评估和比较的，所以你可能想要用用户确定哪个变量在基线附近。

这个例子里，浅蓝色的变量可能在每个竖条上的高度都一样。它和别的变量不一样吗？怎么不一样？哪些浅蓝色竖条是最高的？如果不把这些数据系列挪到基线附近，你就很难说出这些问题的答案。挪到基线附近，这个转换把所有蓝色矩形的底部排成了一条直线。现在进行可视化就容易了：第6个浅蓝色条和第12个是最高的，这个不同似乎和整体竖条的高度有某种松散的关系。

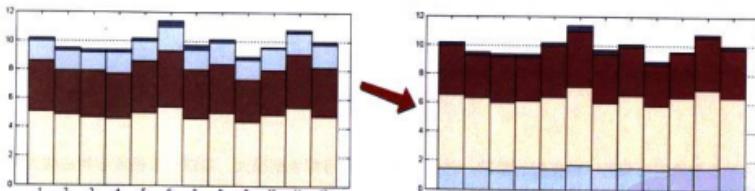


图6-9 对堆叠式柱状图的重新排列

搜索和过滤：只为我显示我需要知道的内容

SEARCHING AND FILTERING: SHOW ME ONLY WHAT I NEED TO KNOW

有时候你不想一次看到整个数据集。你也许是从整个数据集开始的，然后顺着它细化到你需要的数据——过滤。或者，你可能会通过搜索或查询建立该数据的子集。绝大部分用户甚至不会区分过滤和查询之间的区别（虽然从数据库的角度来看，它们之间的区别很大）。不管你使用哪个术语，用户的意图都是一样的：把注意力集中到感兴趣的那部分数据上，然后去掉其他的数据。

最简单的过滤和查询技术为用户提供了一个选择，选择查看数据的哪些方面。复选框和其他一次单击的控件可以把一部分信息图形打开或者关闭。一个表格可能按照用户的选择，只显示其中几列数据，而不显示另外几列；一张地图可能只显示用户选择的兴趣点（例如所有的餐馆）。动态查询模式可以提供更丰富的交互，它是对这类简单过滤控件的扩展。

有时候只是高亮显示一部分数据，而不是隐藏或去掉另外的数据，就已经足够了。如果用这种方式，用户可以看到整个上下文里的这部分数据，同时还可以看到其他数据。你可以使用一些前面提到过的简单数据高亮显示它们。数据刷模式讲述了一种数据高亮的变种，它一次对几个数据图形中那些相同的数据进行高亮显示。

看看图6-10。这张交互式的滑雪路径图可以显示四个种类的路径，通过符号进行编码，再加上其他特性，例如滑雪小车和休息小屋。当所有东西一次全部显示出来的时候，图上十分拥挤，很难看到任何信息。但是用户可以单击这些路径符号，如图所示，打开或关闭那些数据层。第一个屏幕截图显示了没有高亮的路径，而第二个通过一次单击打开了黑钻路径的细节。



图6-10 来自<http://www.sundayriver.com/trailmap.htm>

一种类型的图形和另一种类型的图形，它们的搜索机制可能差别很大。自然，表格或者树应该允许文本搜索；而地图应该提供按地址和其他物理位置的搜索功能；数字图表和图也许应该让用户搜索特定的数值或数值范围。你的用户感兴趣的搜索是什么？

当搜索完成，得到搜索结果的时候，你可能会建立某个界面，在合适的上下文中以图形的方式来查看那些结果——例如，你可以滚动表格或地图，让搜索结果出现在当前视野的中央。在上下文中和其他数据一起查看可以帮助用户更好地理解这个搜索结果。跳转到对象模式常用在一步完成搜索和滚动的时候。

最好的过滤和查询界面是这样的：

高度交互

它们会尽快响应用户的搜索和过滤请求。要实现这一点，对于Web应用和其他需要从网络上获取数据的界面来说，并不是一件容易的事。

迭代

它们允许用户不断细化搜索、查询、或者过滤，直到获得想要的结果为止。它们可能也会结合下面这些操作：用户也许会进行一次搜索，得到一大堆搜索结果，然后把这些结果过滤成自己想要的那些。

上下文相关

它们在整个上下文中显示结果，周围仍然有其他数据，这样可以让用户更容易理解他们在数据空间的具体位置。这一点对于其他类型的搜索来说也是一样的：最好的Web搜索引擎会把搜索关键字和句子一起显示，或者把图片和包含它的网页一起显示。

实际数据：具体的数据值是多少

THE ACTUAL DATA: WHAT ARE THE SPECIFIC DATA VALUES

有几种常规的技术可以帮助用户从信息图形中得到具体的数据值。了解你的目标用户——如果他们只想得到数据的大体结果，那么就没有必要花很多时间和空间去描述每个细节。但是一些实际的数据和文字通常是必需的。

由于这些技术都和文字有关，因此别忘了那些让文字看起来更美观的图形设计原则：可以阅读的字体、合适的字号大小（不要太大，也不要太小），和其他不相关的文本对象进行适当的视觉分离，相关对象对齐，不要使用深色的边框，不要故意让数据难以理解。

标签

很多信息图会直接在图形上放置标签，例如地图上的城镇名称。也可以用标签表示数值，例如散点图上某个符号的值、柱状图上的竖条值，还有其他依赖变量轴和图例的数据值。标签很容易使用，它们可以精确而毫无疑义地表示数据（如果放在合适的地方）。它们经常放在当前数据里面或数据旁边——不用在数据点和图例之间来回切换。它们的缺点是，如果过多地使用标签，可能会让图形变得很杂乱，要小心这一点。

图例

当你在信息图形上使用颜色、文本、线段、符号或尺寸大小来表示数据值（或者数据种类、数值范围）的时候，图例会告诉用户什么元素表示什么数据。你应该把图例和图形本身放在同一个页面上，这样的话用户就不需要在数据和图例之间来回切换了。

无论什么时候，只要位置代表了数据，例如地图（但大部分的图表不是这样的），上面这些技术会告诉用户那些位置的数据值是多少。它们可能是一些参考线条或曲线，并在上面标注了参考数值。用户必须在当前数据和坐标轴之间画一条想像的线，也可能要通过插入来找到合适的数字。这种情况对用户来说，比直接加标签更麻烦。但是如果数据很密集的话，标签会显得很乱，而且很多用户不需要从图形中得到精确的数值；他们只想得到关于数据的总体感觉。对于这些情况，坐标轴就很合适。

数据提示

本章讲述了**数据提示 (Datatips)** 模式。数据提示，也就是当用户鼠标移过某个点的时候，用工具提示来显示数据的值。它有着和标签差不多的好处，而又不会产生重叠和杂乱。不过，它们只能用在交互式的图形中。

数据刷

一种叫做“数据刷”的技术可以让用户选择信息图形上的一部分数据，然后看怎样把这些数据放到另一个上下文中。你通常会在两个或者更多的信息图形上使用它。例如，在一个散点图中选择一些突出的数据，然后在另一个显示同样一批数据的表格中高亮这些选中的项目。要了解更多信息，请参照本章的**数据刷 (Data Brushing)** 模式。

模式

THE PATTERNS

因为本章主要讲述交互式的软件，因此绝大多数模式都是针对怎样和数据交互的——移动、排序、选择、插入或修改、探查具体的数值或数值集合。其中一些模式只针对静态的图形：信息设计师们已经知道使用颜色交替的数据行（**Alternating Row Colors**）、多Y值图表（**Multi-Y Graph**）和大量小对象（**Small Multiples**），不过它们已经在软件世界得到了很好的体现。

还有，不要忘记本书中其他地方的模式。从第2章可以回忆一下可选视图（**Alternative Views**）和需要时显示（**Extras On Demand**）模式，这两个模式都可以帮助你组织交互式图形的结构。第3章里的注释滚动条（**Annotated Scrollbar**）和动画转换（**Animated Transition**）模式可以帮助用户在大型而且复杂的数据空间里保持方向感。

不管数据背后的结构是什么，第一组模式可以应用到各种交互式图形中。（其中一些模式比其他模式更难学习或应用，所以要把它们应用到每个你所创建的数据图形上——特别是数据刷和局部缩放，它们都是“强力工具”，最适合经验丰富的计算机用户。）

54 总览加细节
Overview Plus Detail

57 数据刷
Data Brushing

55 数据提示
Datatips

58 局部缩放
Local Zooming

56 动态查询
Dynamic Queries

接下来是一组关于标签和列表的模式。

59 斑马行
Row Striping

61 跳转到对象
Jump to Item

60 可排序表格
Sortable Table

62 新对象行
New-Item Row

级联列表（**Cascading Lists**）和树状表格（**Tree Table**）对于层级结构组织的数据非常有用。如果你在使用树状视图（也叫做“大纲视图”），那就考虑一下这两个模式。

63 级联列表
Cascading Lists

64 树状表格
Tree Table

剩下的几个模式讲述了怎样为多维数据组织信息图形——多维数据就是那些有很多属性或变量的数据。

65 多Y值图表
Multi-Y Graph

67 树状地图
Treemap

66 大量小对象
Small Multiples



图6-11 来自<http://wildfire.usgs.gov>

它是什么

在放大的详细视图旁边放置一个总览图。当用户在总览图上圈出一个视野的时候，详细视图里将显示该视野范围里的内容。

什么时候使用

在把一个数据集的数据显示成一个大型的信息图形——特别是图片或者地图的时候。你希望用户在拥有总体感觉的时候能保持方向感，但同时还希望能够放大它们，得到一定程度的细节信息。用户对数据进行浏览，查看细小的区域，或者搜索感兴趣的地点。抽象一些的总体视图有助于找到这些兴趣点，但用户不必一次就看到所有数据的全部细节——只对一小片区域放大对于得到特定的细节来说已经足够了。

为什么使用

这种处理复杂性的方法由来已久：显示一份比较抽象的视图来表示当前的情形，然后让用户从那个抽象视图放大来得到他们需要的细节。让这两个层次的视图同时显示在一个页面上，可以很快互相对照。

Edward Tufte使用术语“宏观阅读和微观阅读”(micro and macro readings)来形容印刷地图、图表，以及其他静态信息图形中类似的概念。在用户面前一直有一

份大体结构，而且可以根据自己的需要进入相应的细节：“可视化的步调得到了浓缩、减缓，并且得到了个性化”。与此类似，该模式的用户可以在内容中滚动查看、跳转、比较、对照、快速移动，或者慢慢移动。

最终，总体视图可以充当一个“你在这里”的标记。用户可以看一眼总体视图上的视野，然后说出他们在当前整个数据集中的位置。

如何使用

把数据集的总体视图一直显示在页面上。它可以是一个内嵌的面板，如上面的示例那样。它也可以是详细视图旁边的面板，或者另外一个窗口，就像在Photoshop那样的多窗口应用里一样（参见图6-12的示例）。

在总体视图上，把当前的视野显示出来。出于惯例，它们通常是红色的矩形框，但并不一定如此——它们只是需要一眼就能看到，因此可以考虑使用其他颜色。如果图形本身的颜色比较深，那么让视野框的颜色浅一些；如果图形本身的颜色浅，那么让它的颜色深一些。让视野框可以随鼠标拖动，这样的话用户可以抓住它并且让它在总体视图上移动。

详细视图则对视野框内部区域的内容放大并显示出来。这两者之间应该彼此同步。如果移动了视野框，

那么详细视图也应该相应变化：如果视野框变小了，那么详细视图放大的比例应该加大。与此类似，如果详细视图有滚动条或者其他平移功能，那么视野框也

应该随着详细视图的范围移动位置。它们之间彼此的反应应该很快，最好在十分之一秒以内（标准的直接操作反应时间）。

示例



图6-12 Photoshop把图片画布（详细视图）放在左边，总体视图放在右边。导航窗口显示了整个图片的视图，并且有一个红色的方框显示图片画布窗口的大和滚动位置。

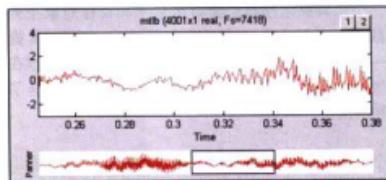


图6-13 在这个信号处理软件里，总体视图面板放在了当前窗口的底部。“你在这里”的标记对那些需要处理又长又复杂信号的用户来说，非常重要。

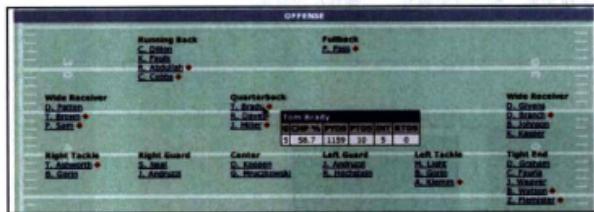


图 6-14 来自 <http://nfl.com/teams/depthcharts/NE>

它是什么

当鼠标滑过图形上一个兴趣点时，把该点的数据显示在一个工具提示或其他某种浮动窗口上。

什么时候使用

你在为一个数据集显示任何形式的总体视图。然后，还有很多数据隐藏在图上各个特定的数据点后面，例如地图上街道的名称、柱状图上圆柱的数值，或者球队深度图上球员的数据。用户可以用鼠标或触摸屏“指着”某个感兴趣的数据点。

为什么使用

查看特定的数据值对于数据图形来说是一种很常见的操作。用户想要总体视图，但他们也可能会需要寻找那些总体视图中没有显示的特定事实。数据提示可以让你为上下文相关的数据显示一小片有针对性的数据。它们把那些数据就显示在用户注意力所在的地方：鼠标指针旁边。如果总体视图的结构组织合理，用户会发现很容易就能找到他们想要的数据——而且你不用把它们都放在图形中。

另外，一些用户可能会很好奇——这里还有什么？我能找到什么？数据提示提供了一种容易上手，并且容

易看到结果的交互性。这种方式很快（不需要载入页面），不需要复杂的技术，为那些可能看不到的内容提供了露个小脸的机会。

不过如果你发现自己在试图使用数据提示来对目标对象进行放大，而不是显示一些额外的数据，那么可以考虑使用局部缩放（Local Zooming）模式。

如何使用

使用一个工具提示那样的窗口来显示和目标对象相关联的数据。在技术上，它没有必要成为一个“工具提示”——重要的是它们需要出现在鼠标光标附近，浮在图形表面，而且它是一个临时对象。用户很快就会了解这种用法。

在那个窗口内部，适当地为数据添加格式。通常显示得密一点更好，因为工具提示窗口一般都很小：不要让这个窗口太大，否则在显示的时候，它会遮住太多的图形内容。而且要把它们放在合适的位置。如果有某种方式规划它们的位置，让它们尽可能遮住更少的内容（再强调一次，参见上面的NFL图形），试试看。

根据不同的情形，你也可能想让数据提示的格式有所不同。例如，一份交互式的地图可能会让用户在地点名称和经纬度坐标值之间切换。如果你有一些数据

集。它们在图表中以几条不同的线段表示，那么每条线的数据提示应该加上各自的标签，或者没有标签，但是数据提示上显示的数据种类不同。

另一种动态显示隐藏数据的方式是在当前图形上或图形旁边保持一个面板，并把这个面板作为静态数据窗口。当用户在图形中不同的数据点上移过时，和这些点相关联的数据就出现在数据窗口里。思路都一样，

不过在这里使用了一个保留的空间，而不是临时出现的数据提示窗口。用户必须把注意力从当前鼠标光标位置移动到那个面板，但是它也有一个好处，就是永远不会有某一块内容受到遮挡。而且，如果该数据窗口可以保持一些数据的话，用户还可以在鼠标进行别的操作时看到那些数据。

示例

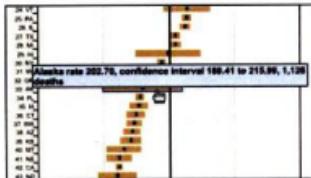


图6-15 这是从一个大规模图形上截取的一小块。每个长条代表一个州，长条X轴方向上中心的位置表示每个州和癌症相关的死亡数目。它的宽度代表这个数字的信心间隔。（甚至采用了一个统计类别？）当鼠标指针移过任何长条的时候，都会显示一个数据提示。该数据提示会显示实际的数字，因此用户不必沿着图形滚动很长的距离来找那些数字了。

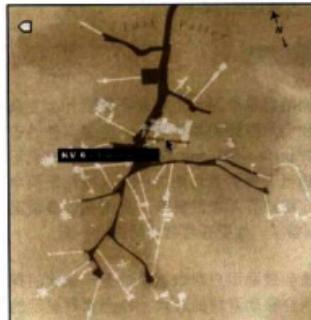


图6-16 就像这个Web应用一样，你可以使用数据提示来有选择性地对地图上任何鼠标滑过的地方加上标签，本图来自<http://thebamappingproject.org>。

动态查询

Dynamic Queries

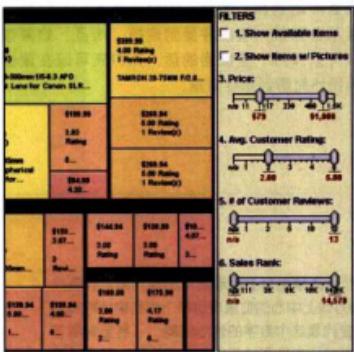


图6-17 来自<http://hivegroup.com>

它是什么

提供一些方式来快速过滤数据，并让这种过滤充满交互性。采用容易使用的标准控件，例如滑块和复选框，来定义数据需要显示的范围。只要用户调整了这些控件的值，显示结果也随之调整。

什么时候使用

你在为用户显示一份大型的、多变量的数据集，一份任何形状、任何展现方式的数据集。用户需要过滤某些数据来完成任意几个目标——去掉不相关的数据，查看哪些数据满足某些特定的条件，理解不同数据属性之间的关系，或者简单地研究数据集，看看有哪些数据存在。

数据集本身有着固定而且可以预期的一组属性（或者参数、变量、维度或别的你喜欢的用词），而用户对这些属性感兴趣。它们通常是数字，并处在某些

范围之内：它们也可能是可以排序的字符串、日期、分类或枚举类型（一组数值，代表非数字的取值）。或者它们可能是显示的信息图形上可视化的一部分数据，并且可以交互式地进行选择。

为什么使用

首先，动态查询很容易学习。用户端不需要什么复杂的查询语言，我们用熟悉的控件表达一些常识性的布尔表达式，例如“`price > $70 and price < $100`”。它们不具备数据查询语言完整的表达能力——只可能是简单的查询，不会让用户界面太过复杂。但是在很多情况下，这样已经足够了。你要在两者之间进行平衡。

其次，即时反馈会鼓励用户对数据集进行开放式的探索。例如，当用户移动滑块的时候，她会看到显示的数据减少了，或者增加了。在增加或去掉了数据集某些子集之后，她会看到它们去了哪里，显示的图形发生了什么改变。因此，通过一会调整这个控件，

然后再调整另一个，她可以逐步控制长而复杂的查询表达。这样，一段持续而且交互式的“问答过程”就在用户和数据之间开始了。即时反馈也缩短了这种反复的时间，因而数据探索变得好玩了，也更可能出现心流的状态（参见第1章，**递增构建（Incremental Construction）**模式）。

第三点——这一点有些微妙了——带标签的动态查询控件可以清楚地显示可供查询的条件，而它们出现在前面最醒目的位置。例如，假设其中一个数据属性是一个数字，并且它的范围是0到100，用户将会通过那个起始位置是0，终止位置是100的滑块知道这一点。

如何使用

想找到一种最佳方式来设计动态查询，要依赖于你的数据显示方式，希望创建的查询种类，以及所拥有的工具集的功能。就像前面提到过的，很多应用会把数据属性映射到数据显示旁边的普通控件上。这样的安排可以允许一次查询多个变量，而不只是图形上已经通过空间属性编码的那些属性。还有，绝大多数的人都知道如何使用滑块和按钮。

其他一些程序直接在图形显示设备上提供交互式的选择功能。通常，用户在一个感兴趣的范围四周画一个方框，然后这个区域里面的数据就会被移除（或只保留这部分，而移除其他的数据）。这种操作是最直接的，但它的坏处是，它被绑定到数据的空间显示方式上了。如果不能在它周围画一个框或选择某些感兴趣的数据点，你就不能对它进行查询。参见**数据刷（Data Brushing）**模式来了解这种类似技术的长处和短处。

因此，回到控件。为动态查询选择控件就像为任何表

单选择控件一样：如何选择取决于数据类型、要进行的查询类型，以及可供选择的控件。一些常见的选择包括：

- 用滑块来表明某个范围内的单个数字。
- 用双滑块或一对滑块来表明某个范围子集：“显示那些大于这个数值，同时又小于另外那个数值的数据”。
- 用单选按钮或下拉框（组合框）在几个可能的取值中选择一个值。你也可能会使用它们在所有的变量或整个数据集中进行挑选。在这种情况下，“所有”通常会用来代表一个附加的元数据。
- 用复选框来选择某些取值或布尔变量。
- 用文本输入框来输入单个取值，可能会用在填空（Fill-in-the-Blank）式的上下文中（参见第7章“从用户获取输入”）。记住，文本输入框比滑块或按钮出错的可能性更大。

示例

本章上面的例子显示了有一组六个过滤器的树状地图（参见本章中的**树状地图（Treemap）**模式）。那两个复选框，过滤器1和过滤器2，使用两个简单的固定查询条件有效地筛选出一部分数据：这个对象可以使用吗？它有图片吗？

剩下的几个过滤器使用了双滑块。每个过滤器都有可以相互独立移动的滑块头，让用户来定义一个范围。价格滑块设置的价格范围是80美金到1000美金，当用户移动这个范围的某一端时，整个树状地图都会发生变化，来反应新的价格范围。其他滑块也是这样的。

示例显示了其他几种实现方式，参见以下示例。

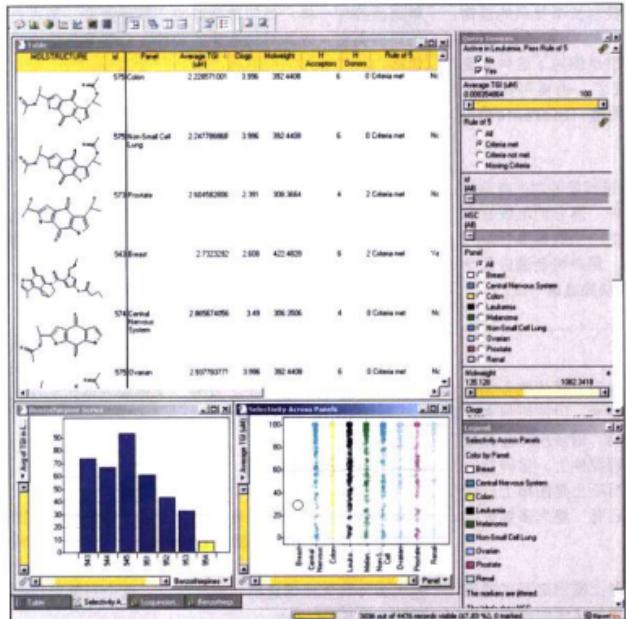


图6-18 Spotfire是一个生物工艺学研究人员经常使用的可视化软件。就像前面的例子一样，“查询设备”的面板放在窗口的右边，它也使用了复选框和双滑块（见中央部分两个黄色的声调滑块），以及单选按钮和单滑块。在这个动态查询面板上，至少显示了八个查询变量。当用户操作这些控件的时候，所有这三个表格和图形上的数据都会进行更新。

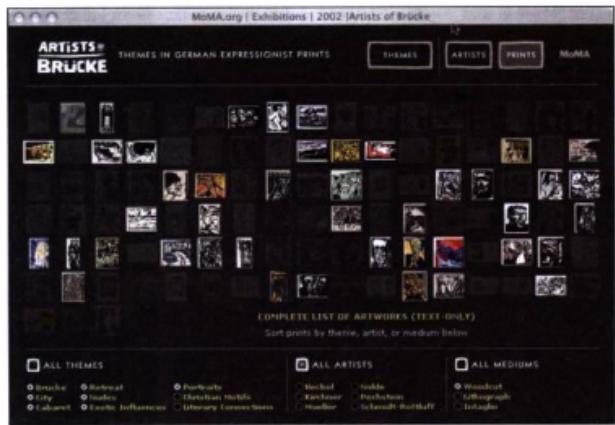


图6-19 这个Web应用允许用户浏览大量的德国表现派印刷物。例如，使用窗口底部的按钮，用户可以对那些印刷品进行过滤（只留下木刻画），或者只显示Heckel和Mueller的肖像作品（方形的复选框上写着“All Themes (所有主题)”，等等。取消选中它们下面的圆形多选按钮——那些按钮看起来像单选按钮，但是其实它们不是）。

这个界面非常强大，可供一名专业用户通过各种方式缩小选择范围来了解这些主题及印刷品之间的关系。参见<http://www.moma.org/exhibitions/2002/brucke>。

数据刷 Data Brushing

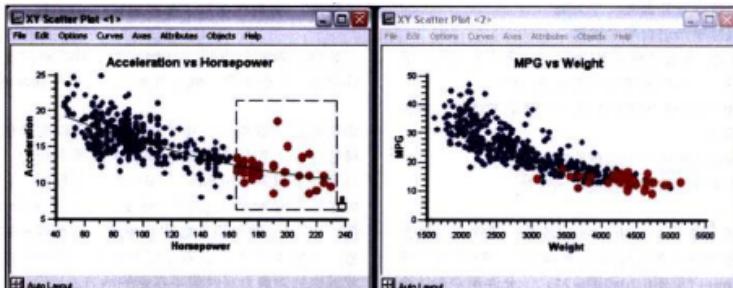


图 6-20 BBN Cornerstone

它是什么

让用户在一个视图里选择数据，然后在另一个视图里同时显示这些选中的数据。

什么时候使用

你一次显示了多个信息图形。可能是两个线形图，再加上一个散点图，或者一个散点图和一个表格，又或者一个图表和一个树形视图，或者一个地图和一个表格，等等等等——只要每个图显示的是同一个数据集就可以了。

为什么使用

数据刷模式提供了一种非常丰富的交互式数据探索方式。首先，用户可以把其中一个信息图形当做选择器，在它上面选择一些数据点。有时候通过可视化的方式，比动态查询那样间接的方式更容易找到数据点——例如，你可以立即看到并操作一个图上的外层数据，而以数字的方式找到它们可能要花几秒钟，甚至更长的时间。“我是需要所有X小于200，同时Y大于5.6的数据吗？我说不准；只要让我在那组点外面画个方框就行了。”

其次，通过同时在多个图形里看到选中的，或者说“刷过的”数据，用户至少可以在另一个图形的上下文中观察这些数据。那种上下文可能有着很大的价值。再次用外层数据的例子来说明，用户可能想知道那些外层数据在另外一个数据空间里处于什么位置，根据其他的变量进行索引——通过学习此模式，他可能很快就能透过产生这些数据的现象得到更深入的了解。

在这里有一个更大的原则，那就是并列的视图或关联的视图。同一数据的多个视图可以关联起来，或者进行同步。这样在一个视图上特定的操作——例如缩放、平移、选择——会在另一个视图上同步看到结果。多个视图之间的协调一致加强了这样一种思想，那就说，视图只是同一数据的不同视角。再次说明，用户在不同的上下文中关注同一批数据，可以得到更深入的了解。

如何使用

首先，用户会怎样选择或怎样“刷”那些数据？这里的问题和你在任何可选择的对象集合中一样：用户可能想要一个或几个对象，连续的或零散的，一次选择全部

或者慢慢递增选择。考虑一下下面这些想法：

- 在一组数据点外面画一个方框
- 通过鼠标单击进行单个选择
- 通过Shift+单击选择某个范围（如果这个范围有意义的话），人们通常在列表里这么做
- 通过Control+单击增加或减少选择的数据库，像列表中那样
- 在数据点外围画一个不规则的“套索”
- 通过菜单项、按钮或按键取消选择

如果你只想使用一个方框，那么考虑一下在一些别的操作之后还把这个方框保留在屏幕上。一些系统，例如Cornerstone（见图6-20和图6-21），允许用户对这个刷取框重新定义大小。实际上，用户可以从任何交互

式的方法来增加或减少“刷过”的数据点中得到好处，因为他可以立刻从另一个视图上看到新刷过的数据点“亮了起来”，那样也增加了更多洞察的可能性。

正如你能看到的那样，其他视图对数据刷的动作响应很快这一点很重要。确认系统可以进行快速响应。

如果刷过的数据点在每个数据视图中以同样的可视化特征显示，包括在数据选择发生的那个视图，那么用户可以更容易找到和认出那些刷过的数据。它们也形成了一个单独的知识层（参见本章介绍部分“预摄变量”一节）。颜色是在数据刷模式中用得最多的预摄变量，可能是因为你很容易就能看到一种明亮的颜色，就算你的注意力当时集中在别的地方。

示例

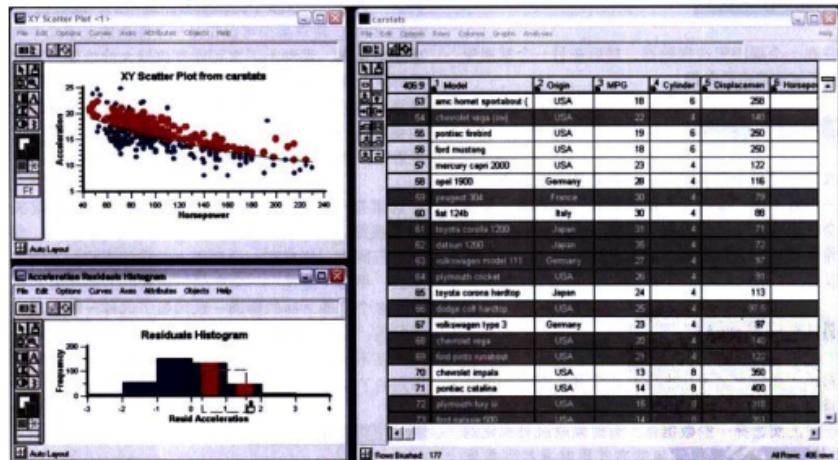


图6-21 这个屏幕截图和本模式开始时的那个图片都来自Cornerstone，一个统计和图形软件包。这里的三个窗口表示了一个散点图、一个柱状图和一个原始数据的表格。所有视图都提供数据刷功能：你可以看到柱状图上两个列外面的数据刷方框。两个图都用红色显示刷过的数据，而在表格里面，它们是灰色的。如果在表格里“刷”了一个小汽车模型，你将会在上面的散点图上发现代表那个模型的点变成了红色，而柱状图上多了一个红色的条纹。

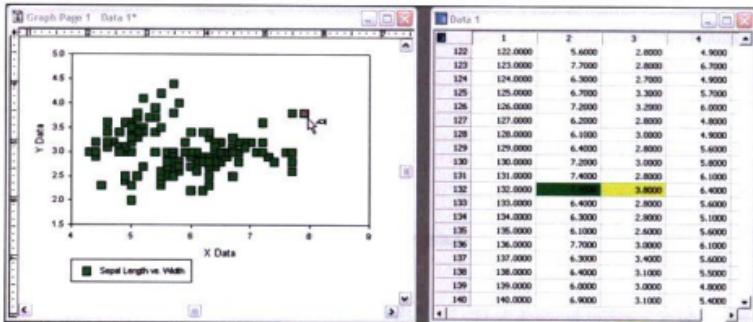


图6-22 Sigmalet允许用户进行单点选择。注意，Sigmalet和Cornerstone都使用了特别的鼠标光标来提示数据刷模态。

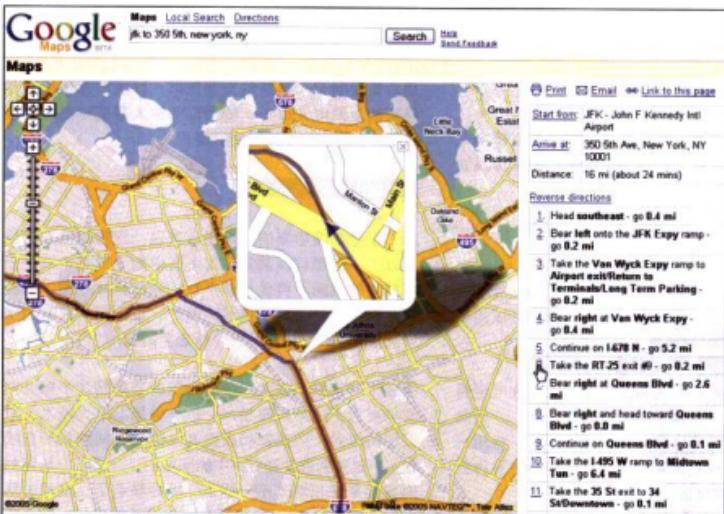


图6-23 Google Map。把这项技术叫做“数据刷”有点过了，但它们的概念是类似的：在一个视图里选择，将会让另一个视图同时展现关于被选数据的一些方面。当用户单击方向箭头上的一个点时，该点的特写图就会放大在地图上（顺便要说的是，这是局部缩放模式的完美使用，因此这个示例将把我们带到下一个模式）。

局部缩放 Local Zooming

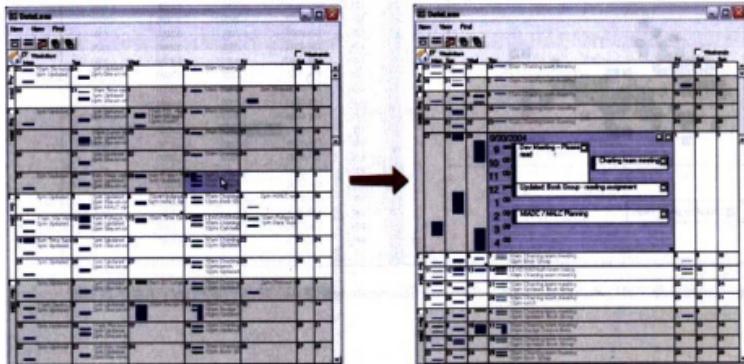


图6-24 日历上的日期镜头

它是什么

在一个密集的页面上显示所有数据，其中有很多小个的数据项目。当鼠标移动时页面会变形，鼠标位置的数据项变大，变得可以阅读。

什么时候使用

你在使用任何组织形式——图、地图、网络，甚至表格来显示大型数据集，屏幕可大可小。用户可以用鼠标光标或触摸屏“指向”那些感兴趣的地方。

用户将会通过那种组织方式浏览数据点或搜索一些感兴趣的点。（例如，找到一个日历上的日期）。因此高层总体视图对于找到这些点来说很有必要，但用户不需要一次看到所有数据点的全部的细节——要得到那些必要的细节，放大就够了。

一些形式的局部放大方式，特别是鱼眼镜头，只有在

用户想要学习一种新的交互技术，以便熟练使用某种特定的应用时才适用。使用局部放大需要有耐心。似乎程序员学起它们来特别容易。

为什么使用

对于那些高度密集的信息图形来说，常规的缩放已经可以了，但那样做常常会失去当前的上下文：一个全屏放大的视图不再显示整个数据集的总体视图。局部放大在集中关注局部细节的同时，也保存了原来的上下文信息。用户还在同一个概念空间里。

不过，对于局部放大，一个可能的开销是这个概念空间的变形。注意鱼眼技术——一种局部放大技术，可以保持放大了的局部和视图其余部分之间的拓扑连通性——是如何改变图6-24中的图形的。一下子，总体视图看起来就和刚才不一样了：地标移动了，空间关系发生了变化（“它以前出现在右边屏幕的中间，但是现在已经不在了”）。

一些其他方式的局部放大不会导致变形。但会把总体视图的一部分隐藏起来。例如，通过一个虚拟的放大镜，用户可以看到放大的区域及总体视图的一部分，但是看不到那些被放大镜边框遮盖住的部分。

总览加细节 (Overview Plus Detail) 模式可以用来作为局部放大模式的替代方案。它也在同一个页面上提供了细节（焦点）及一份总体视图（上下文）。但是把这两个级别的缩放摆在并行的视图上，而不是把它们集成到一个变形的视图里。如果局部缩放很难实现或用户交互起来太难，可以回到总览加细节模式。

数据提示 (Datatips) 模式是它的另一个替代方案。同样，在数据提示模式里，你同时得到了总体视图和细节。但是，如当前数据点所描述的那样，信息并未真正被放大。数据提示是图形表面上一个临时对象，而局部放大是这个图形中的一部分。你可以打印它，也可以在屏幕上捕捉到它。

如何使用

用整个数据集填充所有可用的空间，把它们画得很小。拉伸它来动态地填满整个窗口（参见流式布局 (**Liquid Layout**) 模式）。必要的时候去掉一些细节。如果文字是重要的元素，那么尽量使用小的字体。如果用小字体还不行，就使用抽象的可视化表示方法，例如用实心矩形或线段来表示文字。

提供一种局部缩放模态。当用户切换到这个模态，并四处移动鼠标的时候，对就在鼠标光标下面的小面积区域进行放大。

这种放大看起来像什么样子取决于你使用的是哪种信息图形——它不一定要是字面上的放大，像页面上的放大镜那样。图6-24中的日期透镜，同时使用了水平方

向和垂直方向上的放大和压缩。但是例如图6-25中的表格透镜，只使用了垂直方向上的放大和压缩，因为那些兴趣点的数据是一整行，而不是表格里的一个单元格。不过，一张地图或图片，需要在这两个方向上进行严格控制，来保持它的长宽比例。换句话说，不要拉伸或压缩一张地图，那样会让地图的可读性变得很差。

局部放大透镜可能会被过度使用，因为用户也许会用它去瞄准非常微小的目标。它们看起来并不小——因为它们在透镜下放大了！但是用户实际上是在整个空间里移动，而不是在放大的空间里移动。一个小小的动作变成了一个大型的跳跃。因此，当数据集是不连续的数据点时，例如表格上的单元格或网络节点，你可能会考虑使用**磁性吸附 (Magnetism)** 模式（第8章）来帮助用户从一个焦点移动到另一个焦点。

特别是鱼眼视图，它基本上是可视化领域里的学术研究课题。鱼眼视图承诺了很多，但是我们不太明白怎样才能让它们在产品界面上更有效。例如，我们开始认识到，它们对不太熟练的计算机用户来说，效果并不好。

示例

在这个模式开头显示的日期透镜，是一个同时在桌面和掌上电脑上使用的日历软件。它显示了一份关于日历的总体视图——每行是一个星期——在已经预约的时间上有蓝色的方块。要查看细节，单击一个单元格，这个单元格就会放大。放大的时候使用了**动画转换 (Animated Transition)** 模式（参见第3章），来显示当天的日程安排。在这个设计里，整个图形（除了包含那个单元格的行和列）都压缩了，这样就有空间来显示当前日期的内容（这样实际上为这一周的日程安排及每周四的约会提供了更多信息）。

The screenshot shows a software application window titled "inxight". Inside, there is a table with columns labeled: Price (\$), Status, Bedroom, Bath, Square Foot, Address, City, Zip, Realtor, and MLS#.

Four rows are highlighted in red, indicating they are being viewed in detail:

Price (\$)	Status	Bedroom	Bath	Square Foot	Address	City	Zip	Realtor	MLS#
550	Sale Pending	4	3	1826	1618 WOODST...	San Jose	95118	CENTURY 21...	51219
550	Sale Pending	5	2½	2001	2786 RIOADA...	San Jose	95130	CENTURY 21...	50748
560	Sale Pending	4	2½	1813	EDGEOFRT CT...	San Jose	95123	LAWRENCE W...	101924
561	Sale Pending	4	2½	1893	5872 CHEBRI...	San Jose	95123	COLDWELL B...	100110

Below the table, a note says "Powered by the California Association of Realtors".

图6-25 Inxight的表格透镜允许用户打开任何数目的数据行，并使那个放大窗口在表格中上下移动。这个屏幕截图显示了四个放大的数据行。注意这里只在垂直方向进行了放大。



图6-26 另一个更像透镜的例子来自AT&T。这个网页显示了一份高分辨率地图。当用户移动鼠标的时候，一个文字透镜随着它移动，放大底下的地图来显示这个地区详细的移动电话覆盖情况。参见<http://attwireless.com/media/swf/maps/gsmawonnet-nat.swf>。

斑马行

Row Stripping

Song Name	Time	Artist	Album	Price
3 Silver Thunderbird	4:18	Marc Cohn	○ Marc Cohn	\$0.99
4 Dig Down Deep	5:08	Marc Cohn	○ Marc Cohn	\$0.99
5 Walk on Water	4:01	Marc Cohn	○ Marc Cohn	\$0.99
6 Miles Away	3:23	Marc Cohn	○ Marc Cohn	\$0.99
7 Saving the Best for Last	5:35	Marc Cohn	○ Marc Cohn	\$0.99
8 Strangers in a Car	2:47	Marc Cohn	○ Marc Cohn	\$0.99
9 29 Ways	3:06	Marc Cohn	○ Marc Cohn	\$0.99
10 Perfect Love	4:23	Marc Cohn	○ Marc Cohn	\$0.99
11 True Companion	4:10	Marc Cohn	○ Marc Cohn	\$0.99

图6-27 iTunes

它是什么

在表格行上使用两种样式类似但颜色不同的背景，来间隔显示数据。

什么时候使用

你使用了一个表格，但表格的各行很难在视觉上区分开来，尤其是当表格有很多列数据（或者一行数据中有多行文字或对象）的时候。

为什么使用

温和的颜色块定义并描绘了它们内部包含的信息，就算你没法使用空白来把数据分成不同部分的时候也是如此。制图师和图形设计师很久以前就已经知道这一点了（要记住，不同颜色的背景在定义不同带标题的栏目时也非常有用）。

当某人看到一个单一背景颜色的表格时，因为相邻原则，他们倾向于认为表格的列是内在的数据——表格上每个列上的项目比每个行上的项目更加彼此靠近。但是你希望看到的人可以在按列读取数据的时候同样也能按行读取。通过为相邻的行加上不同的背景颜色，你把行也变成了连贯的可视化对象（这个动作利用了“格式塔原则”（Gestalt Principles）中关于连续性和封闭性的特点，参见第4章）。

具体而言，斑马行可以帮助用户：

- 从左到右浏览一行的数据，然后再回到另一行的开头，不会在行与行之间混淆数据
- 从页面上分别看到表格自己的“脚印”

如何使用

选择两个安静、低饱和，并且相似的两个颜色，但是不能完全相同（换句话说，其中一个颜色要比另一个深一点点）。一些良好的候选方案是浅蓝色和白色，米白色和白色，或者两个不同程度的灰色——无论如何，假设这些背景上的文字是深色的。总的来说，其中一个颜色就是页面上的背景颜色。

数据行上间隔使用不同的颜色。如果行不是很高，你也可以试试对行进行分组：前三行是白色，接下来三行是浅蓝色，等等。

这个模式在视觉上减少了行与行之间的分隔线（虽然你还可以使用格线，如果它们很细而且不显眼的话）。如果你的列与列之间是对齐的，那么不需要使用竖线条来分隔，或者表格周围的边框很宽——查看表格的人关于视觉封闭的感觉会起作用，而行的背景颜色为你定义了表格的边界。

示例

Here are the top women finishers in the 2004 Boston Marathon.

1 Catherine Neterika	31 Kenya	2:24:27
2 Elisaah Alcina	28 Ethiopia	2:24:43
3 Olga Jevtic	26 Serbia and Montenegro	2:27:34
4 Jana Prochazka	27 Latvia	2:30:16
5 Nata Oclu	33 Romania	2:30:44
6 Lucie Demetsky	32 Russian Federation	2:31:17
7 Małgorzata Sobolewska	34 Poland	2:32:23
8 Victoria Klims	29 Russian Federation	2:33:20
9 Ramona Buranadova	42 Russian Federation	2:34:08
10 Ai Yamamoto	25 Japan	2:34:32
11 Rika Tabeshi	22 Japan	2:41:41
12 Jessica Rodriguez Galvan	27 Mexico	2:50:57
13 Andrea Higgenmeyer	34 Germany	2:50:59
14 Greta Varchi	31 Italy	2:54:15
15 Yuriko Ueda	32 France	2:54:49
16 Julie S. Spencer	27 Baraboo, WI	2:56:39
17 Angie M. Battaglia	23 Canada	2:57:06
18 Mary Ann Pritz	47 St. Petersburg, FL	2:57:58
19 Kim A. Donaldson	42 St. Petersburg, FL	2:58:15
20 Lee DiPietro	48 Ruston, MD	2:58:59
21 Tracy Fischer	36 Jamul, CA	2:59:36
22 Stephane Hodge	38 Canada	3:00:00
23 Sononetta Piergentili	39 Woburn, MA	3:01:00

图6-28 2004年波士顿马拉松比赛女子排名，来自<http://boston.com>

Here are the top women finishers in the 2004 Boston Marathon.

1 Catherine Neterika	31 Kenya	2:24:27
2 Elisaah Alcina	28 Ethiopia	2:24:43
3 Olga Jevtic	26 Serbia and Montenegro	2:27:34
4 Jana Prochazka	27 Latvia	2:30:16
5 Nata Oclu	33 Romania	2:30:44
6 Lucie Demetsky	32 Russian Federation	2:31:17
7 Małgorzata Sobolewska	34 Poland	2:32:23
8 Victoria Klims	29 Russian Federation	2:33:20
9 Ramona Buranadova	42 Russian Federation	2:34:08
10 Ai Yamamoto	25 Japan	2:34:32
11 Rika Tabeshi	22 Japan	2:41:41
12 Jessica Rodriguez Galvan	27 Mexico	2:50:57
13 Andrea Higgenmeyer	34 Germany	2:50:59
14 Greta Varchi	31 Italy	2:54:15
15 Yuriko Ueda	32 France	2:54:49
16 Julie S. Spencer	27 Baraboo, WI	2:56:39
17 Angie M. Battaglia	23 Canada	2:57:06
18 Mary Ann Pritz	47 St. Petersburg, FL	2:57:58
19 Kim A. Donaldson	42 St. Petersburg, FL	2:58:15
20 Lee DiPietro	48 Ruston, MD	2:58:59
21 Tracy Fischer	36 Jamul, CA	2:59:36
22 Stephane Hodge	38 Canada	3:00:00
23 Sononetta Piergentili	39 Woburn, MA	3:01:00

图6-29 看看如果去掉那些灰色的数据行背景会怎么样？这些列在视觉上立刻变得突出了，而每一行从左到右更难阅读了。不过，一些设计师发现这样的设计更干净，也更让人心情愉快。在使用或不使用斑马行的时候并没有什么绝对的对和错。

可排序的表格

Sortable Table

Name	Size	Type	Modified
demo		File Folder	8/12/2001 8:36 PM
doc		File Folder	8/12/2001 8:36 PM
frameworks		File Folder	8/12/2001 8:36 PM
javavoc		File Folder	8/12/2001 8:36 PM
lib		File Folder	8/12/2001 8:36 PM
index.html	1 KB	HTML Document	5/1/2001 12:03 PM
index.htm	14KB	HTML Document	5/1/2001 12:04 PM
notes.html	1 KB	HTML Document	5/1/2001 12:03 PM
n_connect.html	1 KB	HTML Document	5/1/2001 12:03 PM
n_dev.html	25KB	HTML Document	5/2/2001 4:51 PM
n_sync.html	1 KB	HTML Document	5/1/2001 12:03 PM

图 6-30 Windows 文件管理器

它是什么

在一个表格里显示数据，并让用户根据表格列的内容对表格进行排序。

什么时候使用

界面上显示了包含多个变量的信息，用户想要探索、重新排序、定制这些信息，搜索其中某个特定的数据项，或者只需要对这些不同的变量有些初步理解。

为什么使用

让用户有能力改变表格的排序方式可以带来很多好处。首先，它有助于数据探索。用户现在可以从数据中得到知识，而这些知识也许在不能排序的情况下永远都看不到——这一类数据有多少？这两个种类之间的比例如何？只有一个这样的数据吗？排在第一位的是什么？最后一位呢？一下子，找到某些特定的数据项也更容易了：用户只要记住该数据项的一个属性就可以了（例如它最后一次编辑的日期），对那个属性排序，然后查找那些他记得的数值。

还有，如果排序顺序从一个软件到另一个软件之间有一致性，那么用户现在可以有效地应用他喜欢使用的模式对界面进行定制。一些用户喜欢让表格从前往后排序，一些用户喜欢从后往前排序；还有一些用户会用一个其他人都不感兴趣的变量进行排序。为用户提供那样的控制权是有意义的。

最后，现在很多用户都很熟悉“可以单击的表头”（The

Clickable-header）这个概念，就算你不提供，他们可能也会想要拥有这种能力。

如何使用

仔细选择数据列（例如，那些变量）。哪些列是用户可能想要排序或想要搜索的？相反，哪些列或变量是不需要在这个表格中显示的？哪些列或变量是可以隐藏起来，直到用户请求某个特定对象更多细节的时候才显示的？

表头上应该有一些可视化的提示，表示它是可以单击的。大多数表格上有凸出的、看起来像按钮一样的边框，或者蓝色带下划线的文字。你应该使用上下方向的箭头来显示排序的方向：升序的还是降序的（箭头的存在可以显示最后一次是在哪个列上进行了排序——一种不错的副作用）。可以考虑在表头上使用鼠标移过效应，例如用高亮或光标变化来加强这种可以单击的感觉。

使用一种稳定的排序算法。意思是说，如果用户首先用名称排序，然后用日期排序，结果列表应该根据日期的不同分成多个组，而每个组中都是用名称排序的。换句话说，如果有可能，表格在下一次排序的时候保持了原来的顺序。这种特点不容易捉摸，但是很有用。

如果界面技术允许，用户可以通过拖拽来重新对数据列排序。Java Swing 就有这个功能。

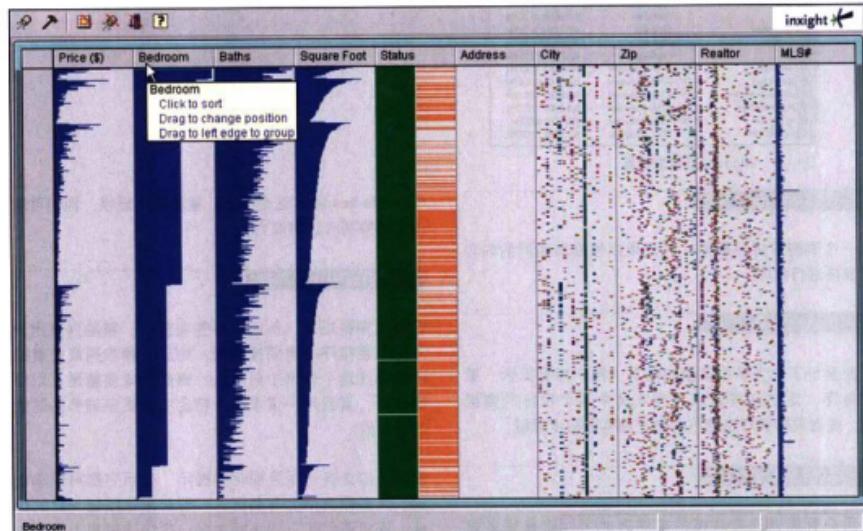


图6-31 Inxight的表格透镜，这是一个表格，它的数据行被压缩成了很多小横条，每个横条的长度代表了这个单元格的值（用户可以单击具体的行来看这些数据行平时是怎么显示的，但那不是我在这里要说的）。这种可视化界面的一个精妙之处在于，它可以在任意一个数据列上排序——当数据高度相关的话，就像这个例子里一样，用户可以很容易看到这种相关性。

这里显示的数据集是加利福尼亚州Santa Clara地区待出售的房屋列表。在这个屏幕截图上，用户已经单击了卧室这一列的表头，因此根据这个变量进行了排序：卧室的数量越多，那个蓝色的横条越长。之前，已经排序的表格是根据平方英尺排序的（也就是房屋的面积大小），因此你可以在这里看到一个次要的“锯齿”模式；例如，它对所有拥有四个卧室的房子根据面积大小排序。洗手间这个变量几乎反映了平方英尺变量，价格变量也是这样，这几个变量展示了一种大致的相关情况。它让人们产生了这样的直觉——房屋的卧室越多，也就越有可能带更多的洗手间，面积也更大。

你可以想像有哪些其他问题可以用这种交互式的图形来解决吗？邮政编码和价格相关吗？价格和面积之间的关系有多强？一些特定的房产经纪人只在特定的城市提供服务吗？这儿有多少房地产经纪人？

跳转到对象

Jump to Item

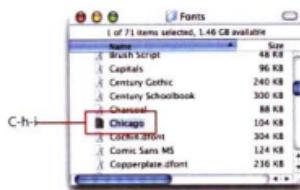


图 6-32 来自 Mac OS X 的 Finder

它是什么

当用户输入某个对象的名称时，直接跳转到该对象并选中它。

什么时候使用

一个界面在使用滚动的列表、表格、下拉框、组合框，或者树来代表一大组对象。这些对象通常是排序的，要么根据字母顺序，要么根据数值。用户想要快速准确地选择其中一个特定的对象，可能更希望通过键盘来选择。

这个模式通常用在文件管理器里，长长的名字列表里，用于国家或城市选择的下拉框里。你也可以把它用在数字上——例如年份或金钱的数量——甚至日历的时间上，例如月份或一周的每一天。

为什么使用

人们并不擅长沿着一个长长的文字或数字列表进行扫描。但是计算机很擅长。让他们各自做自己擅长的事。

1. 参见 <http://www.welie.com/patterns/gui/index.html>。

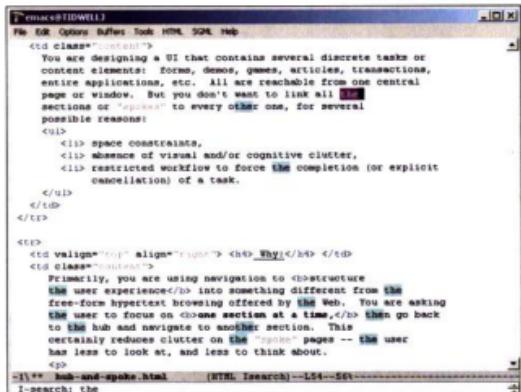
这项技术的另一个好处在于，它允许用户把手一直放在键盘上。当用户在一个表单或对话框里移动时，她可能会发现只要简单地输入几个字母就能从一个列表中选择对象——系统会把这个对象选出来，然后她可以继续做下面的工作。不需要滚动，也不需要单击：用户的手永远不必非得从键盘转移到鼠标上。

如何使用

当用户输入目标对象的第一个字母或数字的时候，跳到匹配用户输入的第一个对象。自动滚动列表让这个对象显示在当前视野内，并选中它。

当用户接着输入更多字母的时候，一直改变选中的对象，让它始终是和用户输入的所有字符一致的第一个对象。如果没有匹配的对象，就停在最近的一次匹配上，不要滚回列表的第一行。你可能想要发出响声来告诉用户没有找到合适的匹配——有些系统就是这么做的，有些不是。

Martijn van Welie在他的模式集里定义了一个和这个模式非常相似的模式，叫做“持续过滤器”¹。



The screenshot shows a GNU Emacs window with the title "emacs@TIDWELL". The buffer contains an HTML document with the following content:

```

<td class="content">
  You are designing a UI that contains several discrete tasks or
  content elements: forms, dropdowns, buttons, transactions,
  entities, applications, etc. All are reachable from one central
  section or window. But you don't want to link all the
  sections or "spoke" to every other one, for several
  possible reasons:
  <ul>
    <li> space constraints,
    <li> absence of visual and/or cognitive clutter,
    <li> restricted workflow to force the completion (or explicit
        cancellation) of a task.
  </ul>
</td>

<td valign="top" align="right"> <h3>Why</h3> </td>
<td class="content">
  Possibly, you are using navigation to restructure
  the user experience</b> into something different from the
  free-form hyper-text browsing offered by the Web. You are asking
  the user to focus on choose section at a time,</b> then go back
  to the hub and navigate to another section. This
  certainly reduces clutter on the "spoke" pages -- the user
  has less to look at, and less to think about.
</td>

```

The status bar at the bottom of the window shows the file name "hub-and-spoke.html" and the line numbers "1-search: the".

图6-13 在GNU Emacs的递增搜索功能里，使用了一个变种的跳转模式。在用户通过Control+S进入了i-Search模式以后，每个输入的字符都会把用户带到文档中字符串出现的第一个位置，不管原来的文档有没有排序。

一旦找到了一个字符串，用户可以通过重复键入Control+S找到其他的匹配目标。在某种程度上，这样的递增搜索会更方便——肯定也更快——和常见的桌面“查找”对话框相比，后者不会随着你输入搜索的字符串持续更新。

还有，Emacs最近的版本在滚动到第一个匹配目标之外，还高亮显示了这个字符串在文档里其他所有的匹配。这样可以为用户提供很多额外的上下文信息，来协助完成他们正在进行的搜索：这是一个普通字符串，又或者不是？它们都聚集在一起，还是分散在各个地方？



图6-34 Outlook里面一个可以编辑的表格

它是什么

在表格的最后一行创建新的对象。

什么时候使用

界面上有一个表格列表、树形视图或其他任何竖直展现的一组对象（每行一个）。在某些时候，用户需要在这个列表中新增一个对象。但是在界面上没有多少可用的空间来显示额外的按钮或选项，而且你希望让新对象的创建效率很高，并让它容易使用。

为什么使用

通过让用户直接在表格的最后一行输入，你把创建新对象的动作放到了这个对象最终显示的同一个位置。在概念上，这样比在界面上其他某个地方进行创建更紧凑。而且，它也让界面更简洁，不需要为创建对象建立一份完全不同的界面。它使用的屏幕空间更少，减少了需要的导航动作（从而不必跳转到另一个窗口），减少了用户的操作。

如何使用

为用户提供一种简单而又显而易见的方式从一个空白的表格行初始化一个新对象。例如，在这个数据行上用鼠标单键就可以开始编辑，或者这个行上可能有一

个“新对象”的按钮，或者它有一个如图6-34所示的初始对象。

在这种情况下，界面应该创建一个新对象，并把它放在那一行上。这个表格的每一列（如果它是一个多列的表格）应该都可以编辑，从而可以让用户为这个新对象输入数值。单元格里可能有文本输入框、下拉框或其他任何能快速而又准确建立对象值的控件。和任何表单格式的用户输入一样，**良好的默认值（Good Defaults）模式**（第7章）可以通过预先填写那些数据节约用户的工作。这样的话用户不必编辑每个数据列。

不过，还有一些别的收尾工作要做。如果用户在结束之前丢弃了新对象，怎么办？你可以从一开始就建立一个有效的对象——如果用户在任何时候丢弃了他的改动，这个对象会依然存在，直到用户回来删除它为止。再说一次，如果这个对象有多个字段，良好的默认值模式有助于预先填写有效的数值。

根据这个模式的实现方式，它可以和**输入提醒（Input Prompt**，第7章）有点类似。在这两个模式里，都会预先建立一个初始化对象，用户通过编辑这个初始对象来输入真正有意义的数值。该初始对象也可以作为输入数据时的“提示信息”，来指导用户建立新对象。



图6-35 这个手机的联系人列表通过一个新对象行提供了在后面添加新联系人的功能。最后一行和其他行一样是可以选择的，然后单击它下面的“Select”按钮，来到一个编辑屏幕，让你为这条新记录输入数据。

级联列表 Cascading Lists

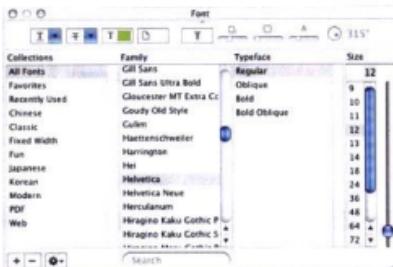


图6-36 来自TextEdit的字体对话框

它是什么

通过在每个层级上显示一个可以选择的列表来表示层级结构。对任何对象的选择都会将其子对象显示在下一个列表中。

什么时候使用

你的数据是通过树形结构来组织的。它的层次可能很深，也可能在每个层次上有很多对象。一个树状视图（大纲）可能适用，但是用户需要上下滚动很长的距离来看到所有的对象，因而他没法对这个层级结构上的对象得到一个更抽象层次的总体把握。

这个层级结构可能是文字的（例如文件系统），也可能是概念上的——这个模式通常用来让用户在不同类别里面移动并选择对象，或者进行一系列互相依赖的选择，就像前面示例中的字体那样。

为什么使用

通过把层级结构展开成几个可滚动的列表，你可以一次见到更多的对象。非常简单。当你处理复杂信息结构的时候，可见性的帮助很大。还有，把对象放在列表上可以很整齐地组织它们——和大纲格式相比，用

户可以更容易地跟踪他在什么位置，因为层级结构是一些整齐的、可以预知的，并且出现在固定位置上的列表。

如何使用

把层级结构的第一个层次放在最左边的列表上（这里应该适用单选规则）。用户从这个列表中选择了一个对象，那么在它右边的列表上显示它的子对象。在这第二个列表上，对它的子对象也可以进行类似的操作，并把选中对象的子对象显示在第三级的列表上。

一旦用户到达了一个没有子对象的对象——“叶子”对象，而不是“树枝”对象的话——你可能想要显示右边最后选中对象的细节信息。图6-37中Mac的Finder示例就显示了图片文件的例子：你可能反过来提供一个界面来编辑一个对象，读取它的内容或任何对于应用来说适当的操作。

这个模式的一个好处在于，你可以很容易地把那些操作按钮关联到每个列表上：删除当前对象、往上移动、往下移动。很多工具集可以让你在树形控件里面直接进行操作，但是对那些没有树形视图的工具集来说，这是一个可行的候选方案。

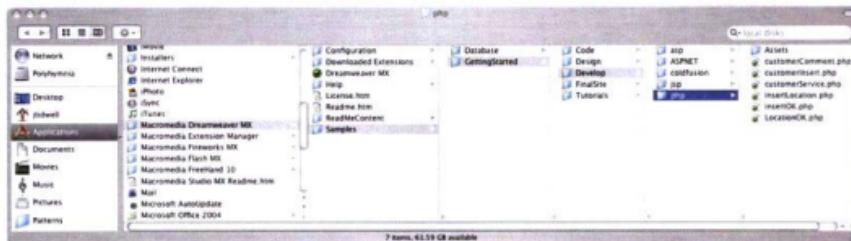


图6-37 这个Mac OS X的Finder屏幕截图是一个极端的例子，它有七个层次。但是它很好地显示了这个模式的规模，让用户向下深入挖掘到文件系统结构中，而仍然保持方向感。（警告：这个模式可能会让那些对层级结构不舒服的人非常迷惑，特别是在层级很深的情况下。）

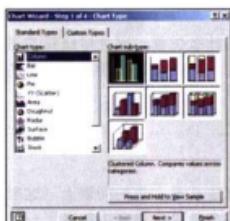


图6-38 它可能看起来不像这个模式，但是该Excel图表式的选择器是一个双层次的级联列表。这个双层列表采用了两种不同的可视化形式。不是文字层次结构的形式，它把各个对象分成几个类别。用户在“Chart sub-type”列表中的选择包括对象的名字和简要描述。同时要注意在这个对话框里使用了卡片堆（Card Stack）和图示选项（Illustrated Choices）这两个模式。

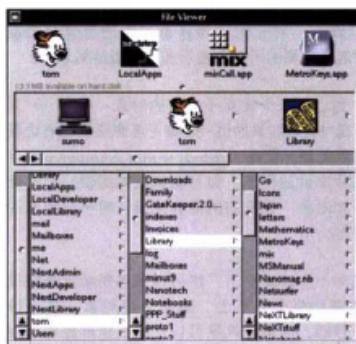


图6-39 NeXTStep最早在它的文件视图里面使用了这项技术，大约在1990年左右（这个屏幕截图来自<http://www120.pair.com/mccarthy/nextstep/intro.html#Workspace.html>）。

树状表格

Tree Table

Subject	From	Sent	Size
↳ [Social machine?]	Yuri Shukov	4/3/2002 12:14 PM	1KB
↳ [Problems with for loops...can someone help??]	Liz Montalvo	4/3/2002 12:18 PM	1KB
↳ [Re: Problems with for loops...can someone help??]	Dan Hendey	4/3/2002 12:36 PM	2KB
↳ [Re: Problems with for loops...can someone help??]	Elizabeth Mont...	4/3/2002 1:10 PM	2KB
↳ [Re: Changing the "zero" element in sparse matrices]	Gumprecht Sait	4/3/2002 1:31 PM	1KB
↳ [Re: Changing the "zero" element in sparse matrices]	Cleve Moler	4/3/2002 2:20 PM	3KB
↳ [Re: Changing the "zero" element in sparse matrices]	Lars Grøgneren	4/3/2002 5:01 PM	2KB
[Determining the number of Matlab Systems open?]	Joshua Staff	4/3/2002 2:08 PM	1KB

图 6-40 Outlook Express 的新闻阅读器

它是什么

把层级结构的数据放到各个列里，就像表格那样，并在第一列使用一种缩进的大纲形式来表现树结构。

什么时候使用

界面上显示了很多关于变量的信息，因此一个表格在显示这种数据的时候很合适（还有，也许允许它们能排序，就像可排序表格模式那样）。但是这项对象主要是以层级结构的方式组织的，因此你也希望在大部分时间里有一个树形结构来显示它们。你的用户对GUI的使用相对比较熟练，这不是一个初级电脑用户容易理解的模式（大多数层级视图都是这样，包括树和级联列表（Cascading Lists））。

为什么使用

把这两个数据视图方法综合到一个视图，这样，只要付出一些可视化和编程复杂性上的代价，就得到了两个视图的好处。你也可以在一个统一的结构里显示该对象的层级结构，再加上一个附加数据或对象属性的表格。

如何使用

这些例子显示了你要做的工作：把树（实际上是一个大纲）放在第一个列里，各个对象的属性放在后续的

其他列里。那些数据行——每行一个对象——通常是可以选择的。自然，你可以把这个模式和可排序表格模式（Sortable Table）结合起来成为一个浏览更方便的、交互式的结构。在这些列上排序，打断树的顺序，因此你需要提供一个额外的按钮或其他一些方式来重新把表格排序成树需要的顺序。

这项技术似乎是从邮件客户端和新闻阅读器中生成的，在那些应用里，讨论的主题常常形成了像树一样的结构。

示例

Bookmarks Manager		
New Bookmarks...	New Folder...	New Separator...
New Page...	New Blank...	New Favorites...
Search...	Properties...	Delete...
Search		
Name	Location	Description
↳ Bookmarks		
↳ Bookmarks Toolbar Folder		Add bookmarks to this folder to see them directly.
Preflife Help	http://dcurtis.net/preflife/	Preflife's Preflife Help site
Preflife Support	http://phoenix.mediorite.org/preflife/	Preflife's Preflife Support forum
Preflife FAQ	http://phoenix.mediorite.org/preflife/	Preflife's Preflife Frequently Asked Questions
↳ Preflife & Mozilla Information		Information about Preflife and Mozilla
Preflife Extensions	http://phoenix.mediorite.org/preflife/	Preflife add-ons and extensions
Preflife Themes	http://phoenix.mediorite.org/preflife/	Preflife's Preflife Themes
Preflife Decisions	http://phoenix.mediorite.org/preflife/	Preflife's Preflife decisions
Mozilla	http://www.mozilla.org	Mozilla community news and advocacy
↳ Quick Searches		Handy searches that can be performed in this window.
		http://startur.net/preflife.html

图 6-41 Firefox 浏览器在对话框里使用了一个外观与众不同的树表。它的分隔器——那些竖线——有助于可视化地把对象分组到不同的类别里，这个想法非常不错。

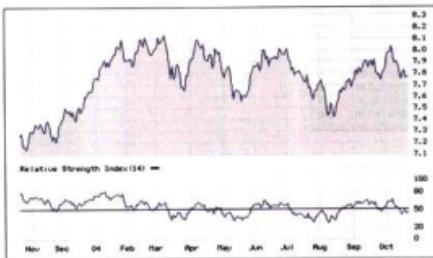


图6-42 来自<http://nytimes.com>

它是什么

把多个图表的线段堆积起来，一个接一个地从上到下摆放在相同的面板里，让它们共享同一个X轴。

什么时候使用

你在展示一个以上图表，通常是简单的散点线形图、柱状图、饼状图（或者这几者的结合）。这些图表里的数据都共享同一个X轴，例如时间轴。但是在其他方面，它们是“苹果和桔子”——它们描述的东西完全不同。你希望鼓励看图的人在显示的这些数据集中找到“纵向”的关系——相关性、相似性、意外的区别，等等。

为什么使用

把这些图表沿着X轴对齐，首先告诉看图者，这些数据集是彼此相关的。这样就可以让用户对数据进行一对一的对照。在图6-42里，两个图表的距离让它们的曲线形状相似性可视化了：你能够看到那些上升和下降的地方基本上一致，显示的格线可以让你进行更精确的观察。例如，“August（8月）”竖直的格线在两个曲线

上都出现在峰值的地方。

你可能已经把一个图表叠加在另一个图表上了。但是通过用它们自己的Y轴分别显示这两个图。每个图都可以用它们自己方式查看，而不会受到另一张图的干扰。

还有，这些数据集有着完全不同的Y值：一个数据集的范围是从7.1到8.2，而另一个的范围是从20到80以上。可能没法把它们放在同一个Y轴上，否则其中一个看起来会像一根水平线。你可能需要在左边再画一个Y轴，然后选择一个合适的缩放比例而不至于让图看起来很古怪。就算这样，直接的叠加还会鼓励用户使用同一个Y轴来思考这两个数据集，并在此基础上对它们进行比较——“苹果对苹果”而不是“苹果对桔子”。如果不是这样，那么把它们叠加在一起将引起一些误会。

如何使用

把一个图形放在另一个图形的上面。为两个图形使用同一个X轴，但是把Y轴的分布分开到不同的竖直位置。如果Y轴需要在某种程度上重叠，也可以，但是请尽量在视觉上不要让这两个图彼此干扰。

有时候你根本不需要使用Y轴：可能对用户来说，找到准确的数值（或图本身包含准确的数值，像带标签的柱状图那样）并不重要。在那种情况下，只要把图形曲线上下移动，到它们不会彼此干扰就可以了。

为每个图形加上标签，这样它们可以相互区分得很清

楚。如果可能，尽量加上格线；格线可以让看图的人随着X值从一个数据集看到另一个数据集，这样比较起来更容易。它们也让看图者有可能发现某个特定数据点（或它附加）的某个准确数值，而不必让用户拿出直尺和铅笔。

示例

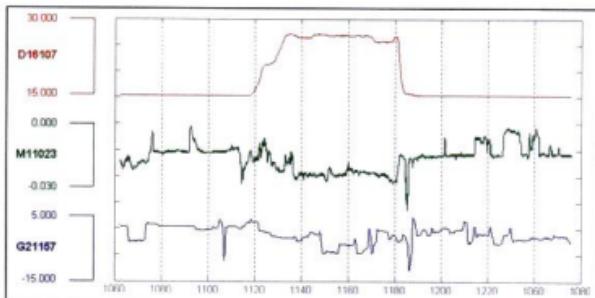


图6-43 这是一个来自Matlab的交互式多Y值图表。你可以用鼠标操作这三个Y轴方向的三个数据标记线，它们在左边用颜色进行了编码——可以把这个标记线在图的上下移动，通过滑动有颜色的阀在竖直方向上拉伸，并利用当场编辑Y轴的极限值来改变Y轴的显示范围。

这就是有意思的地方了。你可能发现这些标记线看起来很相似，好像它们是有关系的——例如，这三个标记线都在标记为1180的地方竖直下降。但是，它们到底有多相关？移动一下看看……

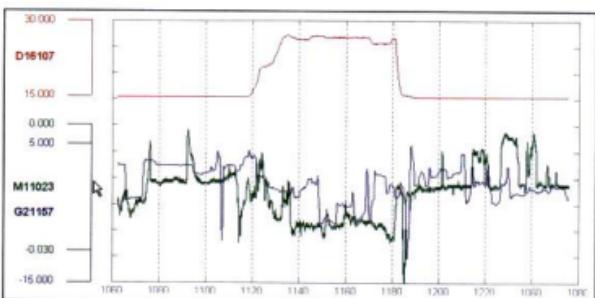


图6-44 我们的眼睛很擅长从可视化展示的数据元素之间找出关系。通过堆积和叠加这些比例不同的图形曲线，用户可以对产生这些数据的现象进行更深入的了解。

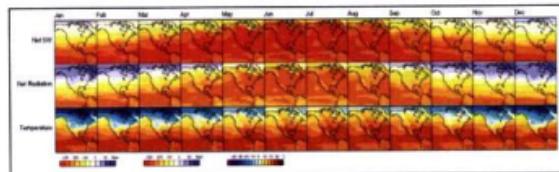


图 6-45 来自 <http://geography.uoregon.edu/shinker/paclim2001>

它是什么

用两三个维度创建很多小的数据图片。并按一到两个附加数据维度平铺到整个页面，要么平铺成连环画式的长条，要么平铺成二维矩阵。

什么时候使用

你需要用两个以上的维度或独立变量来显示一个大型数据集。很容易把这些数据中的一小片显示成一个图片——例如图形、表格、地图，或者图片——但是很难显示更多维度。用户可能不得不一次查看一个图，在它们之间来回切换，看有什么不同。

当使用多个小对象模式时，你需要用一个相当大的空间来显示它们。PDA和其他小屏幕设备很难提供这样的空间，除非每个单独的图片都非常非常细微。但是绝大部分用户会在大屏幕或打印页面上查看它们。

为什么使用

这些小对象有丰富的数据——它们以一种可以理解的方式一次显示了大量信息。每个独立的图片都会讲述一个故事。但当你把它们放到一起，并且展示出每个图片都和其他图片有所区别时，它们就会讲述一个更大的故事。

Edward Tufte曾经说过：“多个小对象的设计、多重数值，以及丰富的数据，通过可视化地增强变化之间的比较、对象之间不同之处的比较、多个可选范围之间的比较，让各种区别很直观。”² (Tufte在他有名的可视化书中给它命名并让这个模式流行起来。)

这样想想。如果你可以把一组维度编码到每个独立的图形里，但是你需要编码另一个额外的维度，它并不适合放到这些图片里，怎么办？

顺序展示

用这样一种方式来展示该维度：假设它是随时间变化的。例如，你可以像放电影一样来播放它，或者使用后退/下一步的按钮来一次查看一张图片。

3D展示

可以把这些图片沿着第三个空间轴——Z轴展示。

多个小对象的方式展示

在一个更大的比例上重用X轴和Y轴。

并行摆放的图片让用户可以在这些图片之间快速自如地扫描。她不需要记住前面显示了什么，在顺序展示里就是那样要求的（虽然一部电影可以通过在各个帧

2. *Envisioning Information*, 第67页。

之间显示细微区别很有效地对信息进行展现）。她也不一定要对一个复杂的3D图形进行解码或旋转，例如把2D图形沿第三个轴展示那样。顺序展示和3D展示有时候表现良好，但是并不适用于所有情况。它们也经常没法在一个非交互式的环境下达到效果。

如何使用

选择要不要展示一个或两个额外的信息维度。如果只有一个额外维度，你可以水平排列图片，竖直排列，甚至自动换行，像连环画那样——用户可以从开始的地方一直读到结尾。如果有两个额外的维度，你应该使用二维表格或矩阵——把其中一个维度放在行，而另一个维度放在列上。

不管你面对的是一个维度还是两个维度，为这些小对象加上很明确的标签，如果有必要，为每一个小对象都加上，要么就沿着显示的方向添加。确认用户会在这些大量的图形中理解数据维度之间的区别，不管你进行编码的是两个维度还是一个。

每个图片应该和其他图片相似：使用相同的大小和/或相同的形状、同样的坐标轴比例（如果是图形的话），以及同样类别的内容。当使用多个小对象模式的时候，你要尽量在所展示的这些东西上引伸出它们之间有意义的区别。尽量减少没有任何意义的视觉差别。

当然，你不应该在一个页面上使用太多的小对象。如果一个数据维度的范围是1-100，你可能不想显示100行

或100列的小对象。那该怎么办呢？可以把这100个数据打包到几个区间里，例如，5个区间，每个区间包含20个数值。或者你可以使用一种叫做“Shingling（鹅卵石）”的技术，这个技术有点像前面所说的区间，但是它允许在区间之间相互重叠。（是的，这意味着一些数据点会多次出现。但也许对那些试图通过数据得到模式的用户来说是一件好事，只要确认它的标签很清楚明白，用户知道它们是什么，就可以了。）

要注意的是，一些显示多个小变量的图上有两个额外编码的维度，它们叫做“格子图形”（trellis plots）或“格子图表”（trellis graphs）。William Cleveland，一个著名的统计图表方面的权威人士就使用这样的术语，软件包S-PLUS和R也是这么做的。

示例

在模式最开始的时候，那张北美气候图就显示了其中有很多编码的变量。每个小对象的下面是一个二维的地图，当然，覆盖在地图上的是一个颜色编码的图形，表示某种气候单位，例如温度。在任何一张图片上，你能看到彩色数据呈现出一些好玩的形状。它们可能会提示看图者来提出问题，如为什么这些彩色小图形出现在某些特定的地区。

整体上，这些小对象编码了两个额外的变量，每一列是一年中的一个月份，每一行代表一个气候单位。你的视线可能随着这些行上的变化，注意到一年之内的变化，然后，在列上进行比较也很容易。

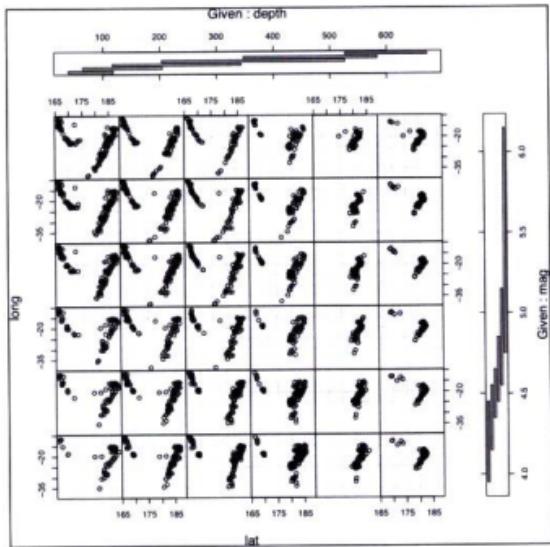


图6-46 这个屏幕呈现的是一个更抽象的二维网格图形。在William Cleveland的*Visualizing Data*一书中也叫做“Coplot”。它是用软件包R绘制出来的。这个例子显示了四个维度度量的数据值：经度、纬度、深度和数量级。在深度和数量级这两个维度的数据互相重叠——这就是之前提到过的Shingling技术。参见http://www.sph.umich.edu/~nichols/biostat_bbag-march2001.pdf。



图6-47 这里有一个更加商业化的例子。出售实体商品的网站经常使用这个模式。Levis的网站就用它来展示可供选择的牛仔裤型号。因为每个图看起来都非常相似——有同样的背景（选中的那个除外），差不多的姿势，以及比例——因此，你的眼睛能看出的区别实际上代表了这些牛仔裤型号上有意义的区别。

但是等一下。这些姿势实际上也有细微的区别，这一点可能对这种大量小对象的展示有帮助。它让整个图形有了一种轻微的粗糙感，就像真实布料上的不一样；这些显示不是百分之百完全统一的。因此它看起来没那么刻板，反而更加舒服。不同的姿态也让用户更容易记住每个不同的模特，并且当用户再次回到网站的时候，更容易从一大组型号中挑选出来。

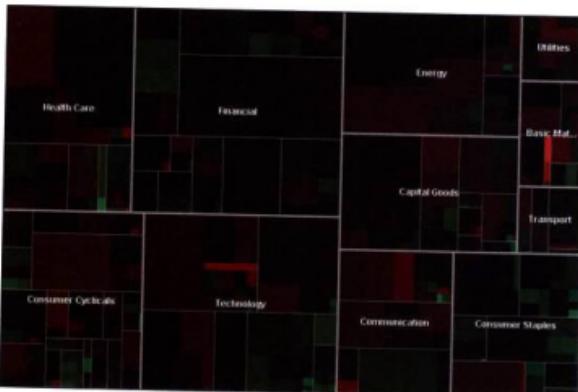


图6-48 Smartmoney的市场地图，网址为<http://smartmoney.com/marketmap>

它是什么

用不同大小的矩形来表示多维数据和/或层级结构的数据。可以把这些矩形级联起来显示其层次结构，并用不同的颜色或给它们加上标签来显示更多的变量。

什么时候使用

你的数据是树形的。或者，它可能是多变量的——每个数据都有好几个属性，例如大小、类别等。这样就可以允许我们根据这些属性对它们进行分组。用户想要看到很多数据点——数百个或上千个——的总体视图。他们使用的屏幕大小足够显示这些大量的数据。

你的用户应该很有耐心，非常想要学习如何使用一个不太寻常的界面。树状地图通常并不好阅读，尤其是对那些以前没有见过它们的人来说。还有，它们在屏幕上比在纸上效果更好，因为还可以同时应用数据提示、数据查询，以及一些其他的交互式机制来帮助用户理解这些数据。

为什么使用

树状地图把很多数据属性编码到一个密密麻麻的图表中，通过利用位置、大小、包含关系、色彩和/或数值、标签，一份树状地图把大量数据打包到了一个空间里，这样就鼓励人类用视觉系统来找出变量之间的趋势、关系和一些特别的数据点。

看看图6-48中的Smartmoney树形图，它显示了当前500支公开交易股票的表现。它显示了不同市场区域的相对大小，以及那些区域里的各个公司——那些实心色块就是一个一个的公司。你可以立即看到当天市场上，绿色区域里的大赢家是些小公司（以小的长方形显示）。红色区域里的大输家也是小公司。Consumer Staples这个板块大体上是绿色的，说明它今天表现良好，能源和健康板块基本上是红色的，说明它们今天表现不佳。但是总的看起来，颜色是中性的灰色，表示今天市场总体上很宁静。

这份树状地图让我们很容易一眼就得出总体印象和图形外观。它鼓励你去查看那些大小和颜色、大小和位置，以及位置和颜色之间的关系——所有这些让你对市场有着不同的理解。想从一个股票价格的大表中获得这些，你可能需要花费不少时间。

如何使用

设计树状地图时关键的一个步骤是确定用哪些可视化变量对哪些数据属性进行编码。

长方形大小

通常长方形的大小会对应一个数字进行编码，例如大小、价格、百分比等。让每个长方形的面积和那个数字成比例。如果该数字的范围太大，可以用一些巨大的长方形和细小的长方形。这时候要么可以让用户对它们进行放大来仔细查看，要么可以让他把那些大的过滤掉，使小的变得大一些。**动态查询（Dynamic Queries**，本章的另一个模式）就经常用来达到这样的效果。见图6-17，看看和树状地图一起使用动态查询模式的例子。

分组和嵌套

如果你的数据本来就有层级结构，例如文件系统或一个分类树，那么你应该相应地对这些长方形进行分组和嵌套。如果不是这样，看看那些数据对象的天然分组或类别信息，而这些信息对用户来说是有意义的。它们有几个可能的类别吗？可以考虑让用户来选择怎样分组。它们完全没有明显的类别？那么你可能会选择几个数值属性，例如价格，然后把它划分到几个类别区间（例如0到5、5到10，等等）。或者你根本就没法对数据进行分组。那么，考虑一下是否采用其他数据方式

会更合适：分组是树状地图的长处之一。

颜色

你可以使用颜色来代表数值或其他有顺序的值（例如上面图6-48中Smartmoney的例子那样），或者不同的类别。对于数值，选择两个颜色作为这些可能数值的起点和终点，例如红色和绿色，白色和蓝色，或者黄色和红色。在这些边界点之间的数值用不同的颜色深浅来表示。对于类别，用不同的色彩来代表每个不同的类别（如果采用的颜色太过类似，那么，看图者可能会假定一个顺序，尽管可能没有这样的顺序）。

位置

长方形在树状地图中的位置部分表述了它在这个层级结构中或层级分类中处在什么地方。但是在一类别中，你可能还是有一些自由来决定把某个给定的长方形放到哪里的。一些树状地图把最大的长方形放在某个类别的左上角，然后依次填充该类别，让最小的长方形出现在右下角。这样建立了一种良好的页面节奏，并且有助于用户可视化地比较每个主要类别中这些小长方形或大长方形的数量。在这种情况下，位置没有编码什么别的变量，相反，它是大小和分组的冗余。不过，其他树状地图的实现会把顺序编码进来，例如年龄、名称的字母顺序。如何编码依你要一次编码的变量数目而定。

绝大部分树状地图允许用户深入挖掘到具体的数据对象。例如，鼠标滑过时通常会看到大的数据提示，详细描述这个对象（参见本章中的**数据提示（Datatips）**模式）。你通常需要省略一些文字来把这些描述放到树状地图方块里面，因此这是一个良好的实践。另外，单击或双击通常会把用户带到和这个

对对象相关的另外一些页面或窗口。需要时显示《Extras On Demand》模式因此可以应用到树状地图的设计中。

在实现的时候，自己编写代码来以某种很好的方式排列这个树状地图通常很有价值。幸运的是，已经有很多现有的算法可以用来绘制一个树状地图。这些算法中一些来自学术论文，一些则来自开源软件或免费软件，其他一些来自产品。这些不同算法的表现和以下

因素有关：它们怎样选择长方形的长宽比例（例如，宽度和高度之间的比例：越方越好），它们怎样填充给定的类别区域，以及它们对于数据变化的稳定性。

树状地图是Ben Shneiderman在1990年发明的，后来，他和他在Maryland大学的同事改进了这项技术。可以从<http://www.cs.umd.edu/hcil/treemap-history/>得到一份关于树状地图的历史，以及很多相关论文和实现的链接。

示例



图6-49 Newsmap的Web应用展示了Google News所形容的“新闻视角”。在任何给定的时刻，Newsmap可以收集Google的头条新闻并画出树状地图，最大的矩形块代表报道最多的新闻故事。它采用的编码因素包括：

- 区块大小——表示新闻条目的“流行程度”，这条新闻的报道次数
- 色彩——表示标题
- 顶层分组——也表示标题
- 颜色值（深浅）——表示时间

因为大标题的“文字大小”和区块大小成比例，所以反过来和流行程度成比例，你的视线会很快留意到那些最大的新闻条目（你会先阅读什么，可能是“False Warnings”的头条，然后“Please work for free!”的条目？）从而，这张树状地图自动组成了视觉层次。参见<http://www.maramushi.com/apps/newsmap/newsmap.cfm>。

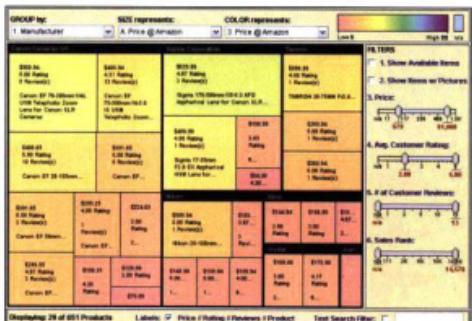
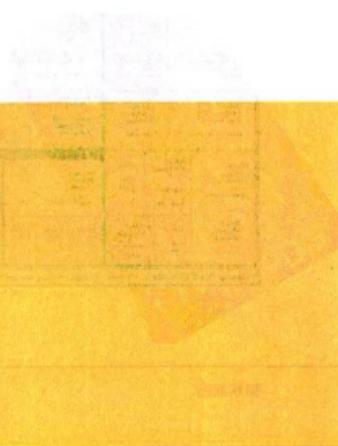


图6-50 来自Hive Group，这是一小组类似树状地图的可视化元素：可以从Amazon购买的商品，世界人口数据，等等。这个屏幕截图显示了某人在查找照相机镜头的例子。每个区块表示Amazon在售的一个产品。在右边是一组动态查询（参见动态查询（Dynamic Queries）模式）。可以让用户从海量数据集中过滤出某些商品。

用户可以设置一些像上面Newsmap例子里那些分组——区块大小、色彩、分组——用顶上的标题条（“GROUP by”、“SIZE represents”，以及“COLOR represents”）。这样的自定义对应用来说很方便。毕竟，有很多数据属性和每个产品相关——对于可以使用的三到四个变量来说，能够进行编码的属性太多了。树状地图的设计者们不知道每个用户最感兴趣的属性是什么。他们进行了很好的默认猜测，并提供了一份简单而又具有可学习性的界面来让用户完成他们自己的任务。参见hivegroup.com。

07

从用户获得输入：表单和控件
GETTING INPUT FROM USERS:
FORMS AND CONTROLS



早晚有一天，你设计的软件可能会要求用户回答一些问题。它甚至就发生在交互的前几分钟：这个软件应该安装到什么位置？你的登录名是什么？你要搜索什么关键字？

这种交互是最容易设计的。每个人都知道怎样使用文本输入框、复选框、组合框。这些输入控件通常是一个设计新手在建立他们第一个图形界面和网站的时候，所用到的界面元素。

不过，它也很容易生成可能会很糟糕的交互。这里是另一个例子：你希望天气预报报告哪个地方的天气？用户可能会想，我得通过城市、国家，还是邮政编码来设定一个地点吗？缩写可以吗？如果我拼错了怎么办？如果我想去的城市它不知道，怎么办？有个地图可以选择吗？还有，不管怎样，它为什么不能记住我昨天已经设好的地点？

本章讨论了很多可以让这些问题得到顺利处理的方式。这里讲述的模式、技术和控件主要应用在表单设计上——所谓表单，通常就是一些简单的问题/答案配对。不过，它们在别的上下文中也很有用，例如Web页面上或应用工具条上的单个控件。输入设计和表单设计是交互设计师的核心工作，你可以把它们用在各种类型的软件及各种平台上。

表单设计基础

THE BASICS OF FORM DESIGN

首先，在进行输入和表单设计的时候，要记住一些原则：

确定用户知道要求回答的是什么问题，以及为什么需要回答这些问题。这一点完全取决于具体的上下文，而且任何归纳总结都可能会没什么意义。但是不管怎样，让我们试试看，你标签上的文字应该能让目标用户容易理解——为初级用户或偶尔访问的用户使用平白的语言，仔细为领域专家用户选择专门用语或词汇。如果你在设计一份篇幅很长或者很枯燥的表单，可以花一点空间来解释你为什么需要用户输入这些正在请他们提供的信息。如果你在往工具条上（或其他类似受到空间约束的地方）放置控件，而且又不是很容易就能看到的话，加上描述性的数据提示或其他上下文相关的帮助来告诉用户它是做什么的。

如果可以的话，根本不要问什么问题。要求用户回答一个问题，特别是如果当他们正在进行其他任务的时候要求他们回答问题，有点类似于强加给用户一个负担。你可能在请他打断他的思考，来让他处理某种没预料到的情况。甚至在最好的情况下，人们也不会认为输入文字信息是一件很有趣的事。你可以在输入控件里预先填写已经知道的信息或猜测的信息吗，像自动完成（Autocompletion）模式推荐的那样？你能提供合理的默认值，来消除用户80%的输入负担吗？你可以从根本上避免这个问题吗？

对于这条原则有一个特别要注意的意外情况：安全性。有时候我们在某种挑战性或响应式的上下文中使用输入控件，例如询问密码或信用卡号码。你显然不希望通过随便预先填写敏感信息来绕过这些安全机制。

保存在“某个地方”的知识常常比保存在“脑袋里”的知识更准确。你不能期望人类能准确无误地记得一大堆东西。如果是要求用户从一段预先提供的选项里进行选择，试试把这个列表放在他们面前让他们可以参考（一个常见的意外可能是填写送货地址时的“州或省”，显然，大部分人能记住他们需要的那个）。下拉框、组合框、列表及其他一些类似的控件可以把所有的选项都显示出来，让用户过目。而且当这些对象有一些可视化的表现方式时（例如图片、版面、颜色等），你可以使用图示选项（Illustrated Choices）模式来显示。

与此类似，如果在请求一些要以特定方式进行格式化的信息时，例如日期或信用卡号码，你可能想为用户提供一些线索告诉他们怎样按格式填写。就算用户以前已经用过你的界面，他也可能不记得要求的格式是什么——一个善意的提示会很有帮助。良好的默认值（Good Defaults）、结构化的格式（Structured Format）和输入提示（Input Hints）这三个模式都可以达到该目的。自动完成模式（Autocompletion）则在通过告诉用户什么输入有效或提醒用户以前几次的输入方面往前多走了一步。

小心，别把表单变成了编程模型的字面翻译。很多表单是用来编辑数据库记录，或者在某个面向对象编程语言里面编辑某个对象的。如果有了一个这样的数据结构需要填写，确实很容易设计一份表单来完成。每个结构元素都有：（1）一个标签，（2）一个控件（或一组控件一起）。把它们按合理的顺序排序，并将其从上到下摆放，就大功告成了。不是吗？

不完全是这样。这种实现驱动的表单设计确实可行，但是它会给你一份枯燥乏味的界面——或很难使用的界面。例如，假设这个结构元素和用户期望提供的输入并不相符，怎么办呢？还有，如果数据结构长达30个元素，怎么办呢？在有些上下文里（例如编程环境里的属性表中），很显然所有的内容都是以它的实现方式来显示的——在那里是没有问题的。但是对于其他情况，一种更简洁更面向用户的展示方式会更好。

因此，这里有一个挑战：你可以使用数据结构之间的依赖性、熟悉的图形结构（例如地址标签）、暗含的假设，或者用户从之前的交互中学到的知识来让这些表单容易填写一点吗？你可以把这个问题转变成直接操作吗？例如在某个范围之内拖拽。来点创意！

进行可用性测试。出于某些原因，当涉及一些输入表单时，设计师和用户特别容易对术语、可能的答案，以及其他和使用上下文有关的问题做出完全不同的假设。本书在前面已经说到这一点了，而且也会再次重复说明：进行可用性测试，就算你很自信，认为自己的设计很好，也要进

行可用性测试。这样做会让你获得经验数据，来看在特定的情形下哪些设计是可行的，哪些设计并不合适。

你选择的控件会影响用户对所问内容的期望，因此要进行明智的选择。单选框建议从多个选项中选择一个，单行的文本输入框建议填写一个简短但开放式的答案。不管有意识还是无意识地，人们会从控件的物理形式——它的类型、大小，等等——来推测当前需要回答的问题，然后相应地设置他们的期望。如果你使用文本输入框来要求输入数字，用户可能会认为任何数字都可以；如果他们输入“12”而很吃惊地看到一个错误对话框，说“这个数字应该大于1小于10”，这就是在给他们添乱。在这种情况下，一个滑块或微调输入框可能更合适。

下面的章节为你提供了一个表格，该表格列出了各种可能用来输入的控件。你或者和你一起工作的工程师需要决定每个问题的语义。是二进制的数据吗？日期或时间？多选一？多选多？是开放的问题但是需要有效性检验？在这个表格里找找，然后根据你特定的设计约束来选择其中的控件。

控件选择

CONTROL CHOICE

下面的图描述了各种你可能要求用户输入的控件和模式，例如数字和从列表中选择。不过这并不是一份完整的清单。实际上，你也许还能说出其他很多控件。但是这里显示的类型都很常见，而且要达到简洁和可用性，它们是最好的选择。

当为某种给定信息类型选择控件的时候，请考虑以下因素：

可供使用的空间

一些控件会占用屏幕上大量的空间，其他一些则没那么多要求，但它们有可能比那些大的控件更难使用。Web页面上的短表单可以把空间花在单选按钮或图示列表上，而复杂应用就可能需要尽可能把更多的内容打包到一个小小的空间里。工具条和表格形式的属性表（Property Sheets）就很符合这种情况，因为它们总的来说只有一行文字高，而且通常也不会太宽。

用户的电脑使用经验

不管设计什么东西，几乎所有用户都知道文本输入框，但不是所有人都会喜欢用双滑块。还有，很多偶然使用的用户也不知道怎样操作多选列表框。

用户的领域经验

如果用户知道只有1~10和20~30之间的数字有效，那么使用文本输入框可能也没问题。但是初学者会在这种情况下遇到麻烦。不过，如果在总的用户群中，初学者只占很少一部分的话，可能也会没事——一个很小的文本输入框可能好过使用大量内部关联的控件。

来自其他应用的期望

大家都知道，粗体/斜体/下划线控件都是图标按钮；如果把它们变成单选按钮，会让我们觉得很奇怪。

可用的技术

在本书写作的时候，HTML里只提供了一小部分常用的桌面应用控件：文本输入框、单选框、复选框、滚动的列表、不能输入的下拉框。一些商业GUI工具箱和开源GUI工具箱会提供更丰富的控件。它们提供的控件各不相同，但是其中很大一部分可以通过编程扩展，来允许你为特定的情形创建自定义控件。

下面的内容总结了输入场景中可以使用的各种可选控件：对象列表、文本、数字、日期和时间。每种选项都用了一个典型的例子来讲解，这些例子来自Windows 2000的外观和感觉（记住，这些例子对于如何最好地展示这些控件的外观来说并不必要。你可以在一定程度上自由决定如何画出它们，特别是在Web上。参见第9章关于进一步讨论的介绍）。

对象列表

LISTS OF ITEMS

很多熟悉的控件允许用户从列表中选择对象或选项。你所选择的控件取决于需要选择对象的个数或选项个数（一个或多个），以及可供选择的对象格式（两个、几个或许多）。

用来选择一个或两个选项的控件（二元选择）。

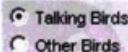
复选框

- 优点：简单，仅需少量空间。
- 缺点：只能表达一个选项，因此它的反面信息不明确，没有显式地声明出来。也就是说，当它没有选中的时候，人们可能不清楚到底意味着什么。



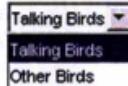
两个单选按钮

- 优点：两个选项都已经声明出来了，并且是明白可见的。
- 缺点：需要更多的空间。



双选项的下拉框

- 优点：两个选项都已经声明，仅需很少的空间，且可以预计需要的空间；如果将来有更多的选项，很容易扩展。
- 缺点：一次只能看到一个选项，使用的时候需要一些技巧。



“按下-保持”模式的切换按钮

- 优点：和复选框一样，如果是图标按钮的形式，将会特别节约空间。
- 缺点：和复选框一样；还有，和复选框不一样的是，它并不是文本选项的标准控件。



两个单选按钮组成的菜单

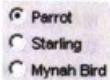
- 优点：在主界面上需要的空间非常小，因为它只会出现在菜单条或弹出菜单上。
- 缺点：弹出菜单可能很难被发现，使用的时候需要很多技巧。



用来进行n选1的控件，但是n很小。

n个单选按钮

- 优点：所有选项都是可见的。
- 缺点：需要n倍单选按钮的空间。



n个项目下的下拉框

- 优点：需要的空间很少。
- 缺点：除了在下拉框打开的时候能全部看到之外，一次只能显示一个选项，使用的时候需要一些技巧。



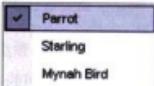
n个项目的一组互斥图标切换按钮

- 优点：需要的空间很少，所有选项都是可见的。
- 缺点：图标的含义可能不明确，需要工具提示来帮助理解；用户可能不知道它们是互斥的。



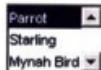
n个单选按钮菜单项的菜单

- 优点：在主界面上需要的空间非常少，因为它通常只出现在菜单条或弹出菜单上；所有选项都是可见的。
- 缺点：弹出菜单很难被发现；使用的时候需要很多技巧。



单选列表或单选表格

- 优点：一次可以看见多个选项；规模可以小到三个元素。
- 缺点：比下拉列表框或微调按钮需要的空间更大。



微调按钮

- 优点：需要的空间很少。
- 缺点：一次只能看到一个选项，使用的时候需要很多技巧，初级用户对它不熟悉。下拉列表框可能是更好的选择。



用来进行n选1的控件，但是n很大。

n个选项的下拉列表框，如有必要，可以滚动

- 优点：需要很少的屏幕空间。
- 缺点：除了打开的时候外，一次只能看到一个选项，在下拉项目中滚动选择需要很多技巧。



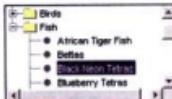
单选列表或单选表格

- 优点：一次可以看到多个选项，如果需要，可以让它的面积保持很小。
- 缺点：比下拉列表框需要更多的屏幕空间。



用来单选的树或级联列表，可选项目已经按类别排列

- 优点：可以看到很多可选的选项；这种组织方式在某些情况下可以帮助寻找选项。
- 缺点：计算机水平不高的用户可能对它不熟悉；需要占用很多空间；使用的时候需要很多技巧。



定制的浏览工具（例如文件对话框、颜色或字体对话框）

- 优点：适合用来浏览可供选择的选项。
- 缺点：某些用户对它们不熟悉；难以设计；通常是一个单独的窗口，因此不如那些放在页面上的控件直接。



用来从n个选项里头选择多个选项（选择顺序并不重要）的控件。

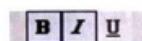
拥有n个复选框的数组

- 优点：所有的选择都已经声明，并且是可见的。
- 缺点：需要占用很多的屏幕空间。



拥有n个切换按钮的数组

- 优点：需要占用的屏幕空间很少；所有选项都是可见的。
- 缺点：小图标含义可能比较模糊，需要工具提示来帮助理解；可能看起来互相排斥。



一个菜单，菜单条上有n个复选框菜单项

- 优点：在主界面上只需要很少的屏幕空间；因为它只会出现在菜单条或弹出菜单上；所有的选项都是可见的。
- 缺点：弹出菜单可能难以发现；使用时需要相当的技巧。



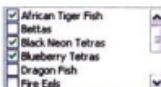
多选列表或多选表格

- 优点：一次可以看见很多选项；如果需要的话也可以只占用少量的屏幕空间。
- 缺点：如果不滚动，就看不到所有的选项；需要很多（但是大小可以限制）的屏幕空间；用户可能意识不到这是一个多项选择。



有复选框的列表

- 优点：一次可以看见很多选项；如果需要的话可以只占用少量屏幕空间；选择启示很明显。
- 缺点：如果不滚动，就看不到所有的选项；需要很多（但是大小可以限制）的屏幕空间。



多选树或多选级联列表（Cascaded List）

- 优点：一次可以看见很多选项；这种组织方式在某些情况下有助于找到选项。
- 缺点：计算机水平不高的用户可能对它不熟悉；使用的时候需要很多技巧；看起来和单选树一样。



定制的浏览工具（例如文件对话框、字体或颜色对话框）

- 优点：适合浏览可供选择的选项。
- 缺点：某些用户对它们不熟悉；难以设计；通常是一个单独的窗口，因此不如那些放在页面上的控件直接。



列表建造器（List Builder）模式

- 优点：已经选好的选项很容易看到；如果希望，选择可以是一个排序的列表；易于处理大量的选项。
- 缺点：因为要显示两个列表，所以需要非常多的屏幕空间；不容易处理大量已经选好的选项。



为用户输入的对象建立列表的控件，这个列表是不需要排序（无序）的。

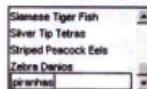
带有“Add”（添加）或“New”（新增）按钮的列表或表格

- 优点：添加的动作明显并且可见。
- 缺点：需要更多的屏幕空间；带来视觉混乱。



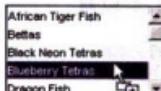
应用了新对象行（New Item Row）模式（第6章）的列表或表格

- 优点：需要的屏幕空间较少；可以当场完成编辑。
- 缺点：添加的动作没有按钮那么明显。



可以接受拖拽操作的列表或表格

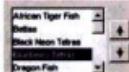
- 优点：视觉上很简洁，节约屏幕空间；拖拽动作效率很高，而且很直观。
- 缺点：添加动作隐藏在背后，因此用户可能不知道该列表是一个可以接受拖拽对象的容器。



建立一组有序对象的控件。

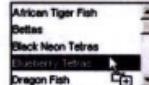
带“往上”“往下”操作提示的无序列表

- 优点：重新排序的动作是可见的。
- 缺点：需要占用更多的屏幕空间。



带内部拖拽功能进行排序的无序列表

- 优点：视觉上很简洁，节约屏幕空间；拖拽动作效率很高，而且很直观。
- 缺点：重新排列的动作是不可见的，因此用户可能不知道有这样的功能。



文字

TEXT

从用户方收集文本输入是最基本的表单任务之一。选择什么控件主要看可以输入的文本行数，不管这些文本行是不是预先定义的选择，以及这些文字有没有格式。

输入一行文本的控件。

单行文字输入框



要么输入一行文本，要么从多个选项中选择一个的控件。

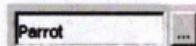
组合框

- 优点：比单独的对话框使用起来更快；用户熟悉这个控件。
- 缺点：可以放到下拉列表里的对象数目有限。



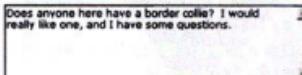
文本输入框，但是旁边有一个“More”（更多）的按钮（你也可以不使用文本输入框，而使用组合框）

- 优点：允许调用专门的选择对话框（例如文件查找对话框）。
- 缺点：用户对这种方式没有组合框那么熟悉；不够直接快速。



输入多行无格式文本的控件。

多行文本输入框



输入多行有格式文本的控件。

带内嵌Tag的多行文本框

- 优点：熟练的用户可以通过直接输入这些Tag来避免使用工具条。
- 缺点：不是真正的所见即所得。



富文本编辑器

- 优点：直接，因为编辑当中的文本又可以作为预览使用。
- 缺点：需要使用工具条，因此不能只用键盘操作。



数字

NUMBERS

因为经常需要遵循更复杂的格式和规则，所以在表单上的数字输入比基本的文字输入更复杂。采用哪种输入选项取决于需要输入哪种类型的数字，以及它所允许的范围。

允许输入任何数字类型的控件。

带容错格式（Forgiving Format）的文本输入框

- 优点：视觉上很简洁；允许输入各种不同格式或不同类型的数字。
- 缺点：采用这种控件，期望的数字格式并不明显，因此可能会引起一时的混淆；需要仔细的后台检查。

617 555-1212

采用结构化的格式（Structured Format）的文本输入框

- 优点：从控件的形式来看，期望的数字格式很明显。
- 缺点：可能需要更多的屏幕空间；带来更大的视觉复杂性；不允许任何违背特定格式的数字，甚至在用户需要这么做的时候也是如此。

617 - 555 - 1212

微调框（最适合用在整数或者不连续的步数上）

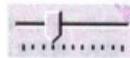
- 优点：用户可以通过鼠标单击得到数字，而不需要触及键盘；也可以在需要的时候直接输入。
- 缺点：不是所有用户都熟悉它们；你可能需要按下按钮很长时间来达到想要的数值；使用那些细小的按钮需要技巧。

4

从一个有界范围内输入数字的控件。

滑块

- 优点：这是一个很明显的隐喻：数值在范围中的位置得到了可视化的显示；用户不能输入超出这个范围的数字。
- 缺点：需要很多屏幕空间；没有明显的键盘访问方式；所使用的标签可能会让它看起来十分拥挤。



微调控制器

- 优点：当使用按钮时数值不会超出范围；只需要少量的屏幕空间；同时支持只使用键盘和只使用鼠标这两种访问方式。
- 缺点：不是所有的用户都熟悉它们；使用这些小按钮需要技巧；需要验证；不能可视化地看到范围内的值。

30%

带事后检查机制的文本输入框（可以有输入提示（Input Hints）、输入提醒（Input Prompt）等）

- 优点：每个人都很熟悉这种方式，只需要少量的屏幕空间；支持键盘操作。
- 缺点：需要验证；没有对可输入内容的限制，因此你必须通过其他方法来传达范围信息。

30%

带文本输入框的滑块（也可以采用下拉选择器（Dropdown Chooser）的形式，在下拉框里放置一个滑块）

- 优点：同时允许可视化输入和文字输入。
- 缺点：复杂，当两个元素都在界面上的时候需要大量空间；当用户输入的时候需要检查文本框的有效性。

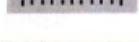
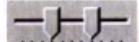


30%

从一个大范围内输入一个子范围的控件。

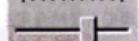
双滑块（如上所述，还可以和文本输入框一起使用）

- 优点：和使用两个滑块相比，需要的屏幕控件更少。
- 缺点：很多用户都不熟悉这种用法；除非使用文本输入框，否则不能通过键盘进行操作。



两个滑块（也可以随文本输入框一起使用）

- 优点：看起来不像双滑块那么吓人。
- 缺点：需要很多空间；除非使用文本输入框，否则不能通过键盘进行操作。



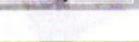
两个微调控制器（可以通过填空（Fill-in-the-Blank）进行连接）

- 优点：如果使用按钮，会让数值受到范围的约束；需要的屏幕空间较少；同时支持只使用键盘和只使用鼠标两种访问方式。
- 缺点：不是所有用户都熟悉这种方式；使用这些细小的按钮需要技巧；需要验证；不能可视化地看到范围内部的数值。



带错误检查机制的两个文本输入框（可以使用输入提示、输入提醒和填空这几个模式）

- 优点：所有人都熟悉这种方式；比滑块需要的屏幕空间少很多。
- 缺点：需要验证；对能输入的数值没有约束，因此你需要通过其他方法传达范围信息。



日期或时间

DATES OR TIMES

因为可能的格式和国际化的问题，从用户端接收日期和时间输入也许会很复杂。可以用来输入日期或时间的控件包括：

容错格式（Forgiving Format）的文本输入框

- 优点：视觉上很简单；允许各种输入格式和输入范围；支持键盘访问。
- 缺点：从控件的形式来看，期望的数据格式不明显，因此可能会引起一时的混淆；需要仔细的后台验证。

Wed 7/21/2004

结构化的格式（Structured Format）的文本输入框

- 优点：需要的数据格式可以从控件的形式上看得出来。
- 缺点：可能需要更多的屏幕空间，带来了更多的视觉混乱；不允许偏离特定的格式，哪怕用户真的希望这么做。

7 / 21 / 2004

日历或时钟控件

- 优点：这两个控件是明显的隐喻；输入被约束在允许的范围内。
- 缺点：需要很多屏幕控件；可能不提供键盘访问。



带日历或时钟的下拉选择框（Dropdown Chooser）

- 优点：结合了文本输入框和日历控件的优点；不需要多少屏幕空间。
- 缺点：交互很复杂；从下拉框中选择数值需要技巧。



模式

THE PATTERNS

你可能已经猜到，如果从上节中的“控件列表”一直读下来，绝大部分模式描述的是控件——确切地说，你怎样把控件和其他控件及文本结合起来让它们更容易使用。一些模式定义了各个元素之间的结构关系，就像下拉选择器和填空模式。其他一些模式，例如良好的默认值和自动完成，讨论的是控件的值和这些值的变化。

这些模式中有很多主要是在讨论文本输入框。这一点应该不会让你吃惊。文本输入框就像灰尘一样遍地可见，但是并不容易让用户弄明白该怎样对待它们。如果放在一个合适的上下文中，让它们的使用方法很明确，那么，文本输入框是最容易使用的控件。这六个模式为你提供了很多方法来创建这样的上下文：

68 容错格式
Forgiving Format

71 输入提示
Input Hints

69 结构化的格式
Structured Format

72 输入提醒
Input Prompt

70 填空
Fill-in-the-Blanks

73 自动完成
Autocompletion

接下来的三个模式要讲述的控件就不是文本输入框了。下拉选择器模式描述了一种创建定制控件的方法；图示选项模式在列表和其他控件里引入了可视化部件；列表建造器模式则描述了一种常见的控件组合方式，来让用户建立一个对象列表。

74 下拉选择器
Dropdown Chooser

75 图示选项
Illustrated Choices

76 列表建造器
List Builder

你应该把剩下这些模式设计到整个表单里。它们可以很好地应用在文本输入框、下拉框、单选按钮、列表框和其他有状态的控件上，但是，在表单内部（或者对话框内部，甚至整个应用里）使用它们的时候，应该保持一致。

77 良好的默认值
Good Defaults

78 错误显示在同一页
Same-Page Error Messages

容错格式 Forgiving Format

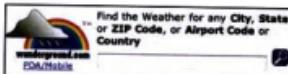


图7-1 来自<http://wunderground.com>

它是什么

允许用户以多种格式和语法输入文本，并让应用系统具有智能解释文本的能力。

什么时候使用

你的UI请求用户输入数据，这些数据可能会是空白、连接符、缩写、大写等各种方式的混合。总的来说，该UI可以从用户那里接受各种形式的输入——不同的含义、格式或语法。但你希望让界面保持简单。

为什么使用

用户只想完成某项工作，而不是考虑“正确”的格式和复杂的UI。计算机很擅长处理各种类型的输入（不过，这种能力也不是完全不受限制的）。这样的配合很好：让用户输入任何他需要的内容。如果这些内容合理，那么让软件用它们去做合适的事。

这样可以极度简化用户界面，也就是说用户可以节约大量的脑细胞，不用考虑这些复杂的问题。它也甚至去掉了输入提示或输入提醒的功能，尽管就像上面的例子所示，这几个模式常常一起出现。

你可能会把结构化的格式作为备选方案，但是那个模式在输入格式完全可以预测的情况下表现最好（通常是指数字输入，例如电话号码）。

如何使用

它的秘密（你知道这里头有秘密）是：它把一个UI问题转换成了编程问题，你需要考虑用户可能会输入什么样的文本。可能你要求输入日期或时间，而且只有格式不同——这种情况还是容易处理的。或者你在要

求输入搜索关键词，输入不同，软件的处理也不同。这就困难多了。软件可以为这些不同的情况建立相同的解释方法吗？怎么做？

这个模式在每个应用里的用法都不同。只要确定软件对于不同的输入格式都符合用户对它的期望就可以了。还有，别忘了对一些真正的用户进行测试。

示例

图7-2是来自Outlook建立会议的工具。看看屏幕底部的“start time”和“end time”两个输入区域——文本区域可以接受灵活的输入格式，你不需要提供一个完全定义好的日期，就像那些文本输入框里显示的那样。如果今天是4月24日，你希望在4月29日设置一个会议，就可以按以下任意一种方式输入：

- next Thu
- nxt thu
- thu
- 29/4/2004
- 29/4
- 4/29
- five days
- 5 days

诸如此类——可能还有很多可以接受的输入格式。然后，具体的日期会根据用户的语言和区域设置以合适的形式“回放”给用户，如图7-2所示。



图7-2 Microsoft Outlook

The screenshot shows a Photoshop dialog box with three main text input fields: 'Name:' (with a red asterisk), 'Company:', and 'Serial Number:' (with a red asterisk). Below these are six smaller input fields arranged horizontally, likely for entering serial numbers.

图 7-3 Photoshop 的安装屏幕

它是什么

和前面模式只使用一个文本输入框不同，这里使用一组文本输入框来反映所请求数据的格式。

什么时候使用

你的界面需要用户输入一些特定类别的文本，并且这些文本有一定的格式。用户对这种格式很熟悉，并且它的定义很清楚。还有，你不希望任何用户违背你所期望的格式。这方面的例子包括信用卡号码信息、当地的电话号码、许可证的字符串或数字等，如图 7-3 所示。

总的来说，如果有些数据的格式对每个用户都不一样，那么这个模式并不是很合适。特别要考虑如果你的界面将在别的国家使用的话，实际情况会怎么样。姓名、地址、邮政编码、电话号码——在不同的地方，这些类型的数据都有着不同的标准格式。如果遇到上面这些情况，考虑使用容错格式模式。

为什么使用

这些文本输入框的结构可以让用户知道需要提供什么样的数据。例如，她可以把图 7-3 中的六个文本框映射成 Photoshop CD 上的六组数字，并且知道那就是她需要输入的软件许可编号。在此，界面对输入的预期很明显。她可能也会意识到自己不需要输入任何空白或连接符来分隔这六组数字。

有趣的是，这个模式的实现形式通常是一组小文本输入框而不是一个大文本输入框。光这一点就可以减少数据输入错误。人们很容易仔细检查几个短的字符串（每个字符串大约两到五个字符）。这比检查一个长字符串容易多了。特别是在如果这些数字彼此相关的情况下。与此类似，当分隔成几组之后，很容易转述或记住一长串数字。人类的大脑就是那样工作的。

和容错格式模式相比，它们的目的完全相反：容错格式允许你以任何格式输入数据，不提供什么结构上的线索来说明要求的是什么数据（不过，你也可以使用其他线索，例如输入提示（**Input Hints**））。结构化的格式模式对任何可以预见的数据格式来说都更适合，而容错格式适用于开放的数据输入。

如何使用

设计一组文本输入框来反映要求的输入格式。让这些输入框保持简短，来作为输入长度的提示。

一旦用户已经在第一个文本输入框里输入了所有数字或字符，自动把输入焦点移动到下一个输入框（来确认前一项数据已经输入完毕）。当然，她仍然可以回到前面的输入框并重新编辑。不过现在她知道输入框要求的数据位数了。

你也可以使用输入提醒模式来为用户提供更多的格式或数据线索。事实上，结构化的日期输入框常常会使用输入提醒（**Input Prompts**），例如“dd/mm/yyyy”。

在最简单的情况下，结构化的输入在文字上可以用空白、连接符和括号来形成一串数据，如下面的表格所示。

电话号码 (504) 555-1212

(504) 555 - 1212

信用卡号码 1021 1234 5678 0000

1021 1234 5678 0000

日期 12/25/2004

12 / 25 / 2004

ISBN编号 0-1950-1919-9

0 - 1950 - 1919 - 9

Date:	March	15, 2005	00 : 11	(24 hour time)
Subject:				

图 7-4 LiveJournal 使用了结构化的格式模式及一个用来选择月份的下拉列表框来输入日期。它的默认值是当前的日期和时间。参见 <http://livejournal.com>。

**它是什么**

把一个或多个数据字段设置成散列的句子或短语形式，把那些输入变成“空白区域”，以便用户填写。

什么时候使用

你需要让用户输入数据，通常是输入一行文本、一个数字或一个下拉框里的选择项。你试图把它写成一组标签/控件对，但是典型的标签解释方式（如“名称”和“地址”）对用户理解当前要输入什么还不够清楚。不过，你可以用一种主动语气的句子或短语，通过文字去描述所有内容填写完毕之后应用将会进行的动作。

为什么使用

填空模式可以让界面变得更自我说明。毕竟，我们都知道自己如何完成一个句子（一个动词短语或一个名词短语也可以达到这样的效果）。在一种动词描述的上下文里看到那些输入项或“空白区域”，可以帮助用户理解当前的状况，以及需要提供什么数据。

如何使用

用你所有的修辞技巧来写这个句子或短语。不过这里用的是控件而不是文字。

如果你打算把控件放到中间，而不是最后的话，这个模式在使用文本输入框、下拉列表框和组合框的时候表现最好——换句话说，控件的外形因素（宽度和高度）与句子里的文字相同的时候效果最好。还有，确保句中文字的底部和控件的底部对齐，否则，它看起来会显得松散杂乱。为这些控件设置合适的长度，让它们恰好可以包括用户选择或输入的内容，并在这些控件和周围的文本之间保持适当的空白。

这一点在定义条件的时候特别有用，例如在显示搜索对象或过滤对象的时候就可以这么用。下面Excel和Photoshop的例子就演示了这一点。Robert Reimann和Alan Cooper把这个模式描述成了一种理想的查询处理方式；他们所使用的术语是“自然语言输出”（natural language output）¹。

不过，这么模式里面有个很大的隐患：它会让我们很难对界面进行本地化（把它转换成另一种语言），因为现在用户对它的理解依赖于某种自然语言下的文字。对于某些国际化的产品或网站来说，这是无法容忍的。如果换一种语言，你可能不得不重新组织UI；至少，要聘请一位能干的翻译来让这个界面可以进行本地化。

1. 参见他们的书籍，*About Face 2.0: the Essentials of Interaction Design*（Wiley出版社），第205页。

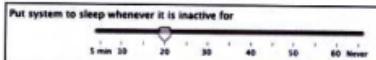


图7-6 Mac OS X系统参数中有几个地方使用了简单的填空方法，这里就是其中的一个。

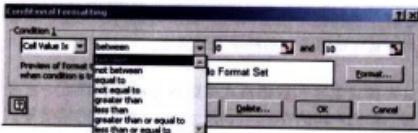


图7-7 这个Excel单元格格式对话框让你在“句子”中通过下拉框选择短语。如果所选的短语发生了变化，后面的文本输入框（这个例子里显示的是0和10）可能会被其他控件代替（例如一个简单的“greater than”文本输入框）。

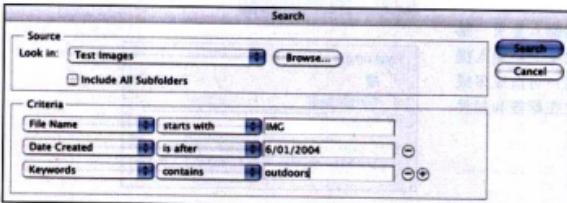


图7-8 Photoshop的图片搜索对话框使用了同上面Excel对话框很相近的方法，但是这个例子里用的是多重条件。

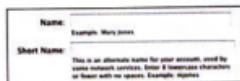


图7-9 Mac OS X的系统参数设置

它是什么

在一个空白的文本输入框旁边，用一句话或一个例子来解释需要输入什么样的数据。

什么时候使用

界面上有一个文本输入框，不过它的输入要求对所有的用户来说并不是很清楚。但是，你又不想把更多的文字放在它的标签里。

为什么使用

如果这个文本输入框可以自己解释它的输入要求，那么用户就不必再去猜测了。如果你在视觉上把输入提示和标签分开，那些知道怎么输入的用户可以或多或少地忽略这个提示，并把注意力一直放在标签和控件上。

如何使用

写一小段示例或解释性的语句，并把它们放在文本输入框的下面或旁边。如果有两个示例，可以用“or（或者）”把它们连接起来。让这些提示文字的字体变小，不那么显眼，但还是要具有可读性；可以考虑使用比标签字体小两号的字体大小。（小一号的字体区别看起来可能更像一个字体错误而不是故意进行的大小设置。）

还有，保持提示信息简短。如果超过两个句子，很多用户就会跳过去，忽略所有的文字。

这个模式常常和容错格式或结构化的格式模式一起使用，如图7-10的Word界面所示。可以使用的候选模式有输入提醒（把一个小的提示放到控制本身里面）、良好的默认值模式（它会把一个真实有效的数值放到控件里）。和输入提醒相比，输入提示允许的提示消息长一些，持续时间也更长，但是因为文本输入框中没有默认值，用户不得不自己考虑这个问题，并给出答案。

示例

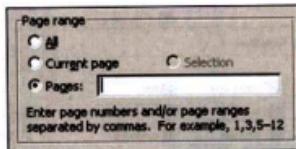


图7-10 几个Microsoft Office软件应用都用到的打印对话框就在一个容错格式文本输入框底下使用了输入提示模式——它包括页码、页码范围，或两者兼有。如果有人没有用过“Pages”选项，输入提示会对他们很有帮助；但对于那些已经知道了该选项的用户，则不用注意那些文字，而只要直接到输入框输入内容就行了。

输入提醒 Input Prompt

*City name (e.g. Boston), or airport code

From Where from?

To Where to?

图7-11 来自`http://orbitz.com`

它是什么

用提示信息预先填写文本输入框或下拉列表框，来告诉用户该怎么做/输入。

什么时候使用

在UI上有一个文本输入框、下拉列表框，或者组合框要求用户输入。正常情况下你会使用良好的默认值模式，但这次不能这么做——也许没有合理的默认值，如图7-11的Orbitz例子所示。用户只需要一个简短的提醒来告诉他要输入什么就够了。

为什么使用

这个模式有助于让界面不言自明。和“输入提示”模式一样，输入提醒也是一种委婉的为控件提供帮助信息的方式，让控件的目的或格式更清楚。

在使用输入提示模式时，某个快速扫描页面的人可能会很容易忽视（或整个错过）那些提示。有时候这是你想要的效果。但输入提醒会保留在用户将要输入的位置，所以不可能看不到它。其好处是，用户不需要猜测她是不是要处理这个控件——控件本身就告诉她这个肯定的答案（别忘了用户填写表单不是为了好玩——他们会尽可能只填写最少的内容，然后离开）。一个问题或一个“fill me in!”的命令更有可能被用户注意到。

这个模式有个有趣的副作用，用户可能根本就不去阅

读文本输入框前面的标签。再看看图7-11。从表单的意图来看，“From”和“To”这两个标签完全是多余的。因为用户的视线先被吸引到空白的文本输入框，很可能先看到那些提醒而不是标签。不过，不要删除那些标签——一旦用户开始输入，这些提示信息就没有了，以后再看这个表单的时候，她可能就忘了这些控件是输入什么的。

如何使用

选择一些合适的提醒文字，这些文字可能会如下一些用词：

- 对于下拉列表框，使用“Select”、“Choose”、“Pick”
- 对于文本输入框，使用“Type”或“Enter”

结尾的文字是一个形容这个输入的名词，例如“Choose a state”（选择一个状态）、“Type your message here”（在这里输入消息），或者“Enter the patient's name!”（输入病人姓名）。把这些短语放到控件里面（提醒文字本身不应该是下拉列表框里的一个可选项，否则，用户选择了它，但是软件并不知道该怎么处理这个输入）。

由于这个模式的要点是要告诉用户，在他们继续进行之前需要输入什么。因此，如果用户没有完成输入，就不要让任务继续。只要提醒信息还在控件内部没有改变，就禁止相应的按钮或其他设备可用，即不让用户完成余下的操作。那样，你就不用向用户显示错误信息了。

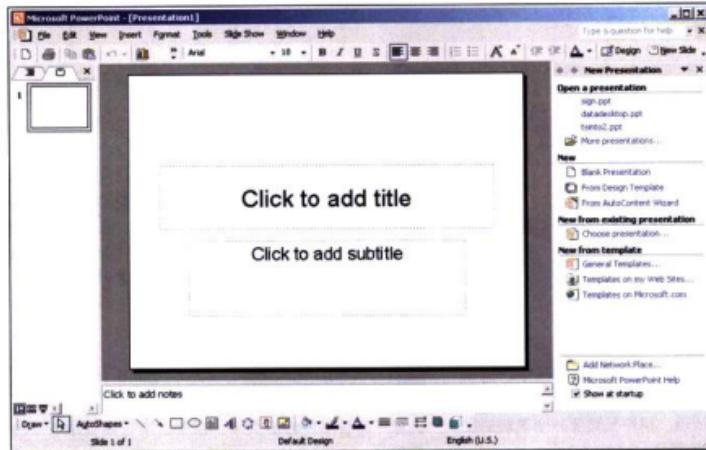


图7.12 在桌面应用历史上，输入提醒模式并不常见，但是这个模式在桌面上和在Web上一样有用。注意这里在所见即所得编辑窗布上的用法——它并没有那么强调“输入区域”，但是应用程序把它的意图清楚地传递给了用户，用户不会迷惑下一步该做什么。

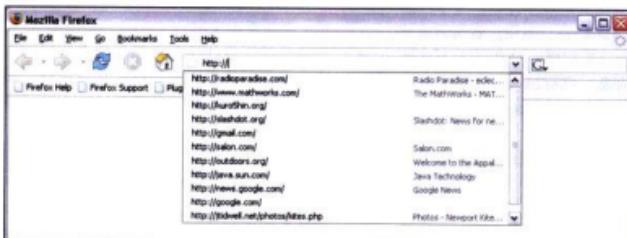


图 7-13 Firefox 应用了自动完成模式来输入 URL

它是什么

当用户在文本输入框内部输入的时候，猜测可能的答案并在合适的时候自动把答案填写到输入框里。

什么时候使用

用户在输入某些可以预见的内容（例如 URL、用户自己名字或地址、今天的日期、或者某个文件名），你可以对他们想要输入的内容进行合理的猜测——例如，系统内部可能有保存的历史输入记录，或者可能用户是在从一组已经存在的值里选择（如某个目录中文件名称列表）。

像URL和邮件地址那样，当输入内容很长或难以输入（或者难以记住）的时候，自动完成将会很有价值。

自动完成模式在文本编辑器和命令行界面中也很常见。当用户输入命令或短语的时候，应用程序或Shell程序可能会为用户提供完成建议。在代码编辑器和操作系统Shell中很适合使用这个模式，因为那些地方使用的语言比较简单，更可预见（和人类的语言相比，例如英语）。所以更容易预期用户将会输入什么样的内容。

为什么使用

让用户免于输入整个内容，并帮他填好。这样，就可以节约很多工作时间，并有利于成千上万个手腕的健康。还有一个额外的好处就是，这样可以防止错误发生：需要输入的文字越长，越不熟悉，用户输入错误的可能性就越大。自动完成后面的内容可以避免这个问题。

如何使用

用户每多输入一个字/字符，软件就会悄悄地建立一种列表，为当前输入列出那些可能的完整内容。如果用户输入的是几个可能的有效取值之一，就使用那些可能的有效值。如果可能的值很多，那么下面这些方法也许能帮助自动完成：

- 使用用户之前输入的内容，就是那些按照个人偏好或历史机制存储起来备用的数据
- 很多用户以前使用过的常用短语，由一个内建的“字典”提供
- 适合当前上下文的资料，例如用来自动填写内部邮件地址的公司通讯录

在这里，你可以通过两种方式达成自动完成模式的交互设计。一种是为用户显示一个可能的完成列表，例如通过按Tab键，让他显式地从列表中选择。很多代码编辑器都是这么做的（见图7-17）。当用户看见列表就能记起自己原来想输入什么的时候，使用这种方法就很合适。如果没有帮助信息，可能想不起来要输入什么。“把知识保存在某个地方好过把它保存在大脑里。”

另一种方式是等到只有一个合理候选项的时候，不进行提示，而是直接把它放在用户面前。Word是用工具提示做到这一点的：很多表单则会填写输入的剩余部分，但是全部选中所有的输入文字，所以另一次按键就会消除自动完成的那个答案。那样的话，用户得到了一个是否保持自动完成的选择——默认的方式是不保存。

你可以同时使用这两种方法，如图7-17所示。

要确定自动完成不会让用户不满。如果你猜错了，用户不喜欢——他将不得不去掉自动完成的答案，然后重新填写一开始想要写的内容，并避免再次让自动完成功能提供错误的答案。下面这些交互细节可以有

助于防止用户产生不满。

- 总是留给用户采用自动完成答案或不采用的选择，默认是不采用。
- 不要干扰正常的输入。如果用户想要输入某些特定的文字，并且不管那些自动完成的选项，而是一直输入，那么确定最后输入的内容就是用户想要的。
- 如果用户一直在某个地方拒绝自动完成功能提供的答案，那么不要一直提供，有些时候可以关掉它。
- 正确地进行猜测。

这里有一些方式能够让自动完成模式实现起来更容易。你可以把一个文本输入框变成一个组合框（结合了文本输入框和下拉列表框）。每次用户在输入框内输入某个唯一值的时候，为它建立一个新的下拉项目。如果你的GUI工具允许在组合框中提前输入（很多GUI工具集都允许），可以自动用下拉列表框里的项目来完成任何用户输入的内容。结合上面的图7-13来看一个典型的例子：很多Web浏览器都会在用户输入URL的时候，把最近访问过的网站保存在一个组合框里。

示例

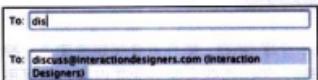


图7-14 自然，很多邮件客户端使用自动完成模式来帮助用户填写“收件人”和“抄送”这两个字段。它们通常会从地址簿、联系人列表或你曾经发送/接收过邮件的邮件地址列表中提取数据。这个来自Mac的例子显示了一个简单的完成项，它是在输入字母“c”的时候出现的；完成的文字自动高亮显示了，一个按键就可以去掉它。这样，你就可以在完成项错误的时候直接输入而不用管它。

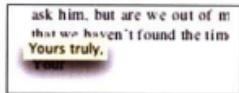


图7.15 Word有一个内建的单词和短语字典。它用这个字典来建议自动完成选项。在此图中，在一行的最开始输入单词“Your”，会让Word建议“Your truly”，那是用在文件签名处的用语。如果Word再聪明一些，它可能会注意到这个地方的文字出现在一本10 000字的书里，和一封信没有什么关系。不过，同样，你可以不管自动完成的项，继续输入。



图7.16 和邮件及Word相比，Google Suggest把自动完成变成了它内部的特性。当你输入的时候，它会显示到目前为止最符合你输入的那些最热门的搜索。不只是作为输入的辅助方式，它还把这个模式变成了一个导航，在一小部分公共知识领域的导航。



```
CMainFrame* pMainFrame = new CMainFrame;
if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
    return FALSE;
pMainFrame->
m_pMainWnd->commandLines();
// call DDE connected events
// In an MDI app
// Parse #_dispatchCount
// shell commands, DDE
// Commands, #_dndDropFromDisk
ParseCommand(_enableToolTips);
// Dispatch #_eventWaitForEvents
// app uses #_eventWaitForCount
// If PROG #_InfoToolMessage
    setcu _initialization
The main window has been initialized, so show and
pMainFrame->ShowWindow(m_nCmdShow);
```

```
CMainFrame* pMainFrame = new CMainFrame;
if (!pMainFrame->LoadFrame(IDR_MAINFRAME))
    return FALSE;
pMainFrame->EnableBooking(
    m_pMainWnd = pMainFrame; // void EnableBooking(DWORD dwBookStyle)
    // call DragAcceptFiles only if there's a suffix
    // In an MDI app, this should occur immediately after setting
    // Parse command line for standard shell commands, DDE, file or
    CCCommandEventInfo cmdInfo;
    ParseCommandLine(cmdInfo);
    // dispatch commands specified on the command line. Will return
    // app was launched with /RegServer, /Register, /UnregServer or
    if (!ProcessShellCommand(cmdInfo))
        return FALSE;
    // The main window has been initialized, so show and update it
    pMainFrame->ShowWindow(m_nCmdShow);
```

图7.17 最后，像Visual Studio那样的代码编辑器提供了一种非常复杂的自动完成机制。自然，Visual Studio的intellisense“会自动完成内建的编程语言关键词，但它也会提取用户定义的函数名称、类名称，还有变量名称等。它甚至还可以显示你所调用的函数参数（在右边的屏幕截图上）。还有，这里也支持“从列表中选择”和“使用最符合的一个完成项”两种方法，而且你可以在任何时候通过Control+Space把自动完成调出来。

Visual Studio里的自动完成从而成了一个输入辅助工具、记忆辅助工具，以及一个上下文相关的函数和类浏览器，很有用。

下拉选择器 Dropdown Chooser



图7-18 Word的颜色选择器

它是什么

通过使用下拉列表框或弹出面板，把菜单条的概念扩展成一个更加复杂的选择界面。

什么时候使用

需要支持用户从一个集合中进行选择输入（例如，如图7-18所示的颜色选择）。需要输入日期/时间、数字或任何不是从键盘上自由输入的内容。你想提供一个界面支持这样的选择（例如一个很好的选项或交互式工具的展示界面），但是你不想使用主界面上的空间。用一个小小的空间来显示现在的取值。这就是你所想要的。

为什么使用

很多用户都很熟悉下拉列表控件（如果和一个自由输入的文本框一起使用，就叫做“组合框”（Combo Box））。许多应用成功地把这一点延续到很多不仅包含简单列表的下拉框，例如下拉树、下拉二维表格，以及其他一些任意的布局。用户在理解它们的时候似乎没什么问题，只要这些控件有向下的箭头按钮表示它们会在单击的时候打开就可以了。

下拉选择器把复杂的用户界面封装到一个很小的空间里。这样，在很多情形下，它们都是很好的解决方案。现在，工具条、表单、对话框，以及各种各样的

Web页面都在使用它们。在用户看来，页面很简单，很优雅。选择器的界面只在用户请求的时候才显示它自己——一种很好的按需显示复杂性的方式。

（总的来说，我不会推荐大家在主要为初级计算机用户服务的UI上使用这些控件。但是Microsoft Office软件已经让大量的用户熟悉了下拉选择器——例如，颜色选择器和字体选择器都利用了这种方式——因此只要确认你的用户觉得舒服就可以了。）

如何使用

在下拉选择器的“关闭”状态下，会用一个按钮或一个文本框显示控件当前的取值。在它的右边放置一个向下的箭头。这可能在它自己的按钮上，也可能不在。做些试验看看哪种情况看起来更好，更适合你的用户。单击箭头（或整个控件）会打开选择器面板，然后再单击一次就把它关闭了。

对选择器面板进行剪裁，使它适合用户需要的选择。让它需要的屏幕空间更小一些，更紧凑一些。它的可视化组织方式应该是某种熟悉的信息图形格式，例如列表、表格、大纲类型的树，或者某种像日历或计算器那样的特殊格式（参见下面的例子）。

如果用户明白这是一个很大的集合中进行选择，那么可以让选择器面板进行滚动。（例如从文件系统中选择一个文件的时候），但是要记住对那些不能熟练

操作计算机的人来说，在一个弹出面板上进行滚动可不是一件容易的事。

面板上的链接或按钮可以打开第二个界面——例如，颜色选择对话框、文件查找对话框，或者帮助页面——可以帮助用户选择一个值。这些对话框通常是模态的。实际上，如果你有意使用某个模态对话框作

为用户选择取值时的主要方式（例如，通过一个按钮打开），你可以使用下拉选择器而不是直接使用模态对话框。弹出面板可以包括最常见或最近选择过的项目。通过让经常选择的项目容易选取，就减少了用户选择一个取值所需要的总体时间（或需要的单击次数）。

示例

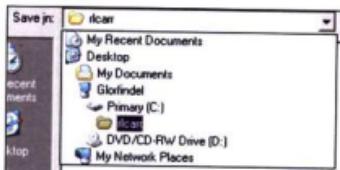


图7-19 不使用列表的下拉选择器之一就是Windows文件管理器中的路径选择器。它看起来像一个树形视图，你不能打开或关闭它的节点，所以它的真实行为看起来更像一个带目标项目的下拉列表框而不再是一棵树，但它在视觉上和下拉列表框不同。

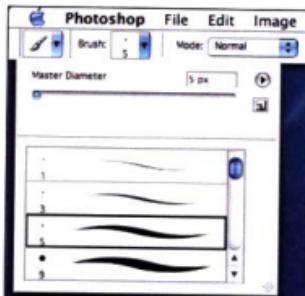


图7-20 Photoshop在它紧凑而充满交互的工具条上大量使用了下拉选择器。这里显示的是Brush控件；Opacity控件的例子在下面。Brush选择器是一个可选择的列表，有一点变形——它有额外的控件，例如滑块、文本输入框、一个向右拉的按钮（圆形的那个）来进行更多的选择。

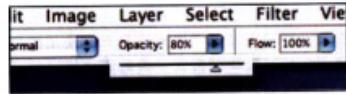


图7-21 Opacity选择器是一个简单的滑块，它上面的文本框反映了它的值（要注意这里神秘地使用了一个向右的箭头而不是向下的箭头）。



图7-22 这是一个来自Money的文本输入框。正常情况下用户会在这个文本输入框中输入一个数值。但是在这个金融应用的上下文中，如果文本输入框上有个计算器，那么会对用户很有帮助。例如，用户可能输入一个数字，然后把它乘以12。



图7-23 就算网站也会赶时髦。Orbitz的日历看起来有点不妥（它出现在离下拉按钮很远的地方），但这是日期选择器的有效表现方式。



图 7-24 Mac OS X 系统的属性

它是什么

用图片而不是文字（或在文字上附加文字）来显示可选项。

什么时候使用

一个界面，需要展示一组外观上不同的选择，例如图片、颜色、字体系列，或者对象对齐方式。

为什么使用

当可视化显示一个概念直接得多的情况下，为什么用文字来表示它？这样可以减少用户的认知负担——他们不需要去思考“Goldenrod”或“Garamond”字体看起来像什么样子——何况同时还让界面变得更吸引人了。

如何使用

首先，每个小图片（或彩色的样本）应该很准确。用户应该能理解他在图示选项模式中看到的东西。除了这一点外，还要显示主要的区别，然后就不需要再表

现别的了。不需要对该选项的效果进行完全缩微的再现。显示简化的、顺畅的、示意性的图片。

你可以在很多控件上使用图示选项的模式：下拉列表框、单选框、可滚动的列表、表格、树，以及像颜色选择器那样的专门的对话框。理想情况下，你可以一次在一个下拉框里，或者工具条上向用户显示一组图示化的选项。然后用户可以立刻对它们进行比较。如果一次只能看到一个（有时候没办法）就像预览（Preview）模式（参见第5章）那样——用户会按时间顺序看到它们。那么对于选项之间的比较就不太好。

有时候，需要同时显示图片和项目的名称。如果用户可以同时看见这两者的好处，那就这么做——他们将学会把图片和名称关联起来，从而可以再进一步跟概念关联起来。界面可以起到教学的效果。

如果需要定制的图标或小图片，那么考虑请一位好的图形设计师来进行设计。确定这位设计师对整个应用的可视化词汇很敏感，然后她才能理解这些选项的意思。

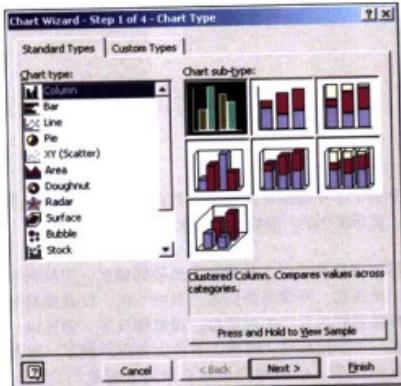


图7-25 当你在Excel中创建一个图表的时候，可以看见图表向导让你选择一个图表类型。这就是笛示选项模式的完美应用：不是所有人都知道雷泡图或雷达图应该是什么样子，但有些人记得它们的图片（还要记得在右下角使用帮助文字，而且，这三个界面元素结合在一起看起来像一个级联列表（Cascaded List））。

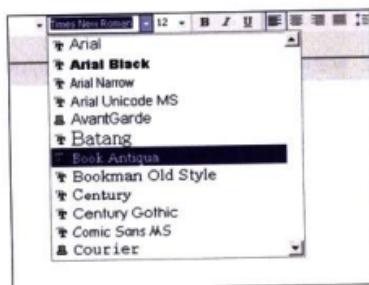


图7-26 字体选择器是一个使用笛示选项的经典范例。在这个来自Word的下拉列表框里，字体名称是用它们自己的字体书写的，因此不需要用户仔细在一个单独的字体对话框或工具里查找——很少有Word用户会在不看到它们的示例之前知道Book Antiqua和Palatino字体之间的区别。

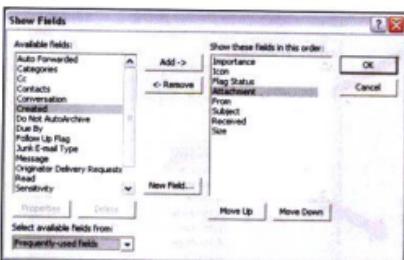


图7-27 来自MS Outlook的一个对话框

它是什么

在同一个页面上显示源列表和目的列表，让用户在它们之间移动项目。

什么时候使用

你希望用户通过从另一个列表中选择项目来建立一个列表框。源列表框可能很长——例如，长到不容易显示成一组复选框。

为什么使用

这个模式的关键是要在一个页面上同时显示两个列表。用户可以看到哪个是哪个——例如，她不需要在模态对话框之间跳转来跳转去（参考示例内容来看这样的例子）。

列表建造器的一种更简单的可选方案是一个简单的复选框列表。这两种方案都可以解决“选择一个子集”的问题。但是如果你有一个很大的源列表（例如整个文件系统），那么一组复选框可能达不到要求——用户不容易看到哪些选中了而哪些没选中，因此也就很难得到一个所选项目的清楚视图。她不得不上下滚动这个列表来回查看。

如何使用

把源列表放在左边，目的列表放在右边。这些项目就会从左边移动到右边。这比从右边移动到左边会更自然。在这两个列表之间，放上增加和删除的按钮；你也可以用文字、箭头或文字加箭头来标记它们。

这个模式也为其他按钮提供了空间。如果目的列表是排序的，那么可以使用上移和下移按钮，如上面的例子所示（也可以给它们加上箭头图标，放在文字旁边，或者不要文字，直接使用）。

这取决于你需要处理的是哪种对象。你可以在表面上把对象从源列表移动到目的列表——因此源列表失去了一些项目，或者维护一份不会变化的源列表。文件系统的文件列表就不应该变化；如果变化了，用户会觉得它很古怪，因为他们把这样的列表看做文件系统的一个模型。而那些文件实际上并没有删除。但是Outlook例子里“可用字段”的列表中的项目就会发生变化。这要依情况而定。

如果可能的话，在这两个列表之间实现拖拽功能，并让这些列表可以进行多项选择而不只是单项选择，这样用户可以在列表之间一次移动大量的对象。

示例

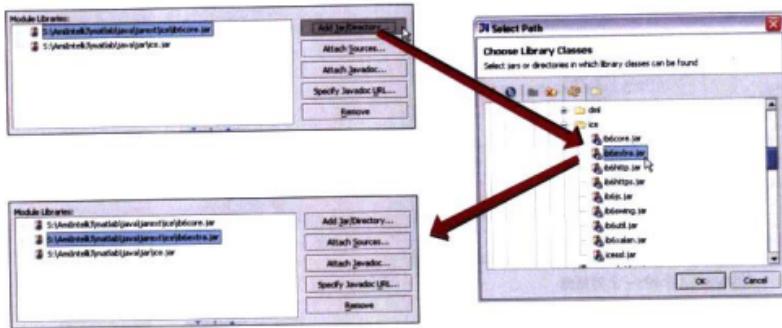


图7-28 IntelliJ Idea是一个复杂的Java开发环境。它有一个令人遗憾的对话框，在这个对话框里，用列表建造器会更合适。这个例子显示的是一个对话框的一部分。用户的任务是从文件系统中选择一些JAR文件；文件可能不多，也可能很多。每次用户想要增加文件到左边的“module libraries”列表时，她不得不打开一个模态对话框，就是右边图中那个，在这个模态对话框里可以选择需要的JAR文件。这样可能会多次打开这个对话框（幸运的是，对话框允许多选，但是如果用户是在慢慢建造这个列表的，她还是需要多次打开这个对话框）。

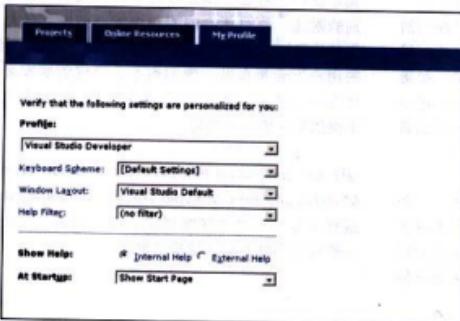


图 7-29 Visual Studio 的 Profile 屏幕

它是什么

无论在什么地方，只要合适，就预先为用户填写你猜测他们想要的输入值到各个字段中。

什么时候使用

你的 UI 要求用户提供任何表单那样的输入（例如文本输入框或单选按钮），而且你想减少用户需要做的工作。也许绝大多数用户会以某种特定的方式回答这些问题，或者用户已经提供足够的上下文信息来让 UI 进行准确的猜测。对于技术问题或半相关的问题，可能他并不想知道或并不关心这个结果，而且“不管系统决定什么”都是可以的。

但是当问题的答案会很敏感或与政治相关时，提供默认值并不总是明智的，例如密码、性别或公民身份。进行那样的假设或预先填写那些需要小心处理的字段时，用户可能会觉得不舒服，甚至会使用户生气（提醒你，不要让“请给我发送广告邮件”的复选框默认选中）。

为什么使用

通过提供合理的默认答案，你节省了用户的工作。确实就是这么简单。不用用户花时间去思考或输入答案。填写表单永远不是一件有趣的事，但是如果这个模式能减少一半的工作时间，他就会非常感激。

甚至就算默认值不是用户想要的，至少你也为他提供了一份示例来讲述可能的答案应该是什么样子的。这一点也可以节约用户几秒钟的思考时间——或者，避免一条错误信息。

使用良好的默认值模式时，有时候你也会遇到意外的结果。如果用户可以跳过一个字段，那么这个问题就没有引起他的注意。他可能会忘了曾经看到过这个问题；也可能不理解这个问题的含义，或者说，默认值的含义。输入一个答案，选择一个取值，或者单击一个按钮的动作会强迫用户有意识地面对这个问题，如果你希望用户了解到应用的有效性，这一点就很重

用一个合理的默认值预先填写文本输入框、组合框，或者其他控件。你可以在第一次向用户显示页面的时候这么做，也可以让用户在使用这个应用时早些时候提供的信息来动态地设置将来的默认值（例如，如果某个用户提供了一个美国的邮政编码，你可以从这个编码推导出这个州、郡，甚至有时候可以得到市和镇的名称）。

如果只是因为你觉得不应该留下空白的输入区域，那么不要选用默认值模式。只有在当你有理由确信绝大部分用户，绝大部分情况下，不会改变这个取值的时候才提供默认值——否则，你将给所有人带来额外的

工作。了解你的用户！

偶尔使用的界面，例如软件安装界面值得加上一个特别的附注。你可能会向用户询问一些技术信息（例如安装位置），这样他们有机会定制这个应用。但是90%的用户不会那么做。他们也不介意你到底安装在什么地方——这个问题对他们并不重要。因此在这种情况下提供默认值非常合理。

Jeff Johnson在*Web Bloopers: 60 Common Web Design Mistakes and How to Avoid Them* (Morgan Kaufmann出版社出版)一书里详细讨论了这个问题。他也提供了一些设计不好的默认值模式例子。

示例

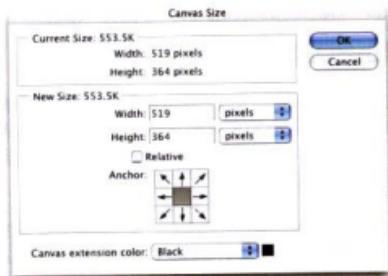


图7.30 当在Photoshop中调整图片画布的大小时，这个对话框出现了。如图所示，原始的图片是519x364。这些数据成了默认的宽度和高度。这样对于几种使用情形来说非常方便。如果想在这个图片周围放上一个窄窄的边框，我可以从这些默认的宽和高开始，为它们都增加两个像素；如果想让图片画布变得更宽而不是更高，我只要改变宽度就可以了，或者我可以单击OK，什么也不改变。

The screenshot shows a registration form on the Netflix website. At the top, a red error message is displayed: "Your email address must contain a '.' (dot) in the host name." Below the message, there are three input fields: "Email Address" containing "notvalid@nowhere", "Create a Password" with a masked password, and "Re-type Password" also with a masked password. A note below the password fields says "4-10 characters (case sensitive)".

图7-31 Netflix的注册页面

它是什么

把表单的错误信息和表单放在一起，直接放在页面上，在页面顶端标记错误信息。如果可能的话，在产生错误的控件旁边进行提示。

什么时候使用

用户可能在填写表单信息的时候，提供了一些不合法的输入。例如他们可能会跳过一些必填的字段或输入了一些不可解析的数字，或者输入了无效的邮件地址。你想鼓励他们再试一次。你想在那些输入成为问题之前指出错误的地方，并帮助迷惑的用户理解现在需要的输入是什么。

为什么使用

在传统情况下，GUI应用会通过模态对话框向用户报告错误消息。这些消息可能会很有帮助，指出问题是什么，可以怎样修复它。这样做的问题是需要单击关闭该模态对话框来修复错误。而随着模态对话框的关闭，你再也看不到错误信息了！（可能你需要记住这些信息。）

然后，当Web表单到来的时候，错误信息通常会用另一个载入的页面来报告给用户，在你单击“提交”之后才显示出来。再次，你可以阅读这个信息，但你需要单击“后退”按钮来修复这个问题；而一旦你这么做了以

后，错误消息又不见了。因此你需要扫描这个表单来找到发生错误的字段，需要精力，也容易出错。

现在，很多Web表单把错误信息放在表单本身上。通过保持消息和控件在一个页面上同时可见，用户可以读到错误信息，这样就很容易对错误进行更正，而不需要跳转或记住出错信息。

甚至还能做得更好，一些Web表单在物理上把错误消息放在引发错误的控件旁边。现在用户可以一眼就看到是哪里出错了——不需要根据控件名字去搜寻出错的地方——而且用于修复错误的指示信息就在旁边，很容易看到。

如何使用

首先，设计表单的时候要防止一些特定的错误发生。如果可以的话，使用下拉列表框而不是开放式的文本输入框；提供输入提示（**Input Hints**），输入提醒（**Input Prompts**），容错格式（**Forgiving Format**），以及其他技术来支持文本输入。清楚地标记那些必填的项目；还有，根本的一点就是，不要在表单中要求填写太多的问题。

如果错误真的发生了，即使你把详细的错误信息显示在控件旁边，也应该在表单的顶部显示一些错误信息。表单顶部是人们首先会看到的地方（对视力受损

的用户也是如此——表单的顶端是他们最先读到的部分，因此他们很快就会知道这个表单出错了）。在那个位置放上一个吸引注意力的图片，然后使用和正文一样或小一些的字体来显示错误信息：例如，让它们变成红色。

现在来标记那些引发错误的表单区域。如果可以的话，在它们的旁边放上详细的信息——这样需要在那些区域的旁边、上面或下面使用额外的空间——但是至少，要用颜色和/或小图片来标记这个字段，如上例所示。

在这里，用户通常把红色和错误联系起来。可以放心这么用，但是由于很多有色盲的人看不到红色，因此也要使用线索：语言、黑体（不要太大），以及图片。

如果你在为Web设计或其他一些客户端/服务器系统设计，试着多增加一些验证，就像你在客户端通常会做的那样。这样会快很多。如果可能的话，把错误消息放在已经载入的页面上，避免额外的页面载入等待。

关于错误消息的手册超出了这个模式的范围，但是这里有一些快速的指引。

- 让它们保持简短，但仍然要足够详细地解释哪个地方出错了。错误的原因是什么：比较“没有提供地址”和“信息不足”这两条提示信息。
- 使用常用的语言，而不是计算机语言：比较“你的邮政编码里有一个字母”和“数字验证错误”。
- 要有礼貌：比较“对不起，出现了一个错误！请再次单击‘Go’”和“Javascript 693号错误”或“这个表单没有包含数据”这几条提示。

示例

The screenshot shows the eBay registration process at step 1: Enter Information. The page title is "Register: Enter Information". It lists three steps: 1. Enter Information, 2. Choose User ID & Password, and 3. Check Your Email. Step 1 is active. A yellow warning icon with an exclamation mark is positioned above a list of validation errors. The errors are:

- Last name - Please enter this information.
- Address - Please enter this information.

A note below the errors says: "Register now to bid or buy on any eBay site. It's easy and free!" Below the note are several input fields: First name (with a red asterisk), Last name (with a red asterisk), Street address (with a red asterisk), City, State / Province, Zip / Postal code, and Country (set to United States). Each field with a red asterisk has a corresponding error message above it: "Please enter this information." The entire registration form is enclosed in a light blue border.

图 7.32 Ebay 的用户注册页面为这个模式提供了一个非常好的例子。表单顶部的错误总结很容易被用户看到，并在注册继续进行之前显示在需要修复的字段旁边（它甚至还为那些有问题的字段提供了链接）。每个有问题的字段都有它们子集的错误信息。这是冗余的做法，但是在该例子里是可以的——一些人在读完顶部的错误信息之前，只会关注两个红色标签的字段。

器
影
像

BRIDGES AND EDITIONS

180

08

Builder和编辑器
BUILDERS AND EDITORS



本章要讲述在各种“编辑器”中的想法和模式。很多用户已经很熟悉一些常见的应用了，如Word和Powerpoint，但是本章中要讲述的想法通常比这些应用更古老，影响也更广。本章的模式主要来自下面这些习惯用法：

页面布局和格式化的文本编辑器

例如Word、Quark和InDesign

图像编辑器

Photoshop、Gimp、Paint shop Pro、Fireworks和Macpaint

矢量图形编辑器

Powerpoint、Illustrator、Coreldraw、Visio和OmniGraffle

网站Builder

Dreamweaver和GoLive

GUI Builder和代码编辑器

Visual studio、Eclipse和Interface Builder

常见的文本编辑器

Notepad、Emacs、Email Composers和大量Web页面

除了它们编辑的媒介之外，这些应用有大量的共同之处。首先，也是最重要的一点，它们都是画布——为用户提供一个空白的壳，让用户可以创造性地用一些辅助的工具来填充它。它们中的绝大部分应用自然地使用了画布加调色板工具条（Canvas Plus Palette）模式（见第2章），展现的方式都是一个空白的画布区域旁边有一个图标化的工具条。

再往回倒退一点，回到第1章，你可能还记得递增构建（Incremental Construction）模式和“心流”（flow）的概念。这两点对Builder类型的软件来说很重要。人们如果在创建某个东西的时候，不会按某种线性的方式工作——他们先创建一点儿，看它外观怎么样（或工作起来如何），然后进行一些修改，增加一点，删除一点，退回去再看看效果。当辅助工具反应迅速，而且适当地针对任务进行了设计时，它们就从用户的注意力中淡出了，用户会完全投入到创造性的活动中。

要做到这一点，用户必须能熟练有效地使用这些工具。幸运的是，用户们会发现，很多给定情形下的编辑器已经发展得非常相似了，因此在一个编辑器里学到的技巧和习惯可以“转换”到其他编辑器中去。并不需要你把编辑器设计得和其他编辑器一样——在编辑器设计上，还是有很充分的创新空间的。不过，最好不要一次发明太多的轮子。

然后，我们来看看工作效率。别让任何东西打断心流状态。例如没有必要的一长列对话框，浪费时间也让人疲劳不已的鼠标移动，或者不得不在软件响应之前的等待。在Builder和编辑器里，尽量做到直接：常规的动作在最少的按键次数或鼠标移动下执行；对于各种动作，快速给出结果。

对任何Builder和编辑器来说，很多交互设计决策与所见即所得的编辑、直接操作、模态或选择有关。要做到高效熟练地使用，把这些概念处理好很关键。

编辑器设计基础 THE BASICS OF EDITOR DESIGN

在绝大部分编辑器中，有四个元素非常重要：进行所见即所得编辑的能力；直接操作界面和数据元素的能力；各种不同的操作模态；选择文本或对象的各种方式。

所见即所得的编辑 WYSIWYG EDITING

所见即所得（WYSIWYG）是“what you see is what you get”的缩写（你可能早就知道了）。它可以让一些界面更容易使用，因为用户不需要在设计的时候进行复杂的心智转换，来计算设计结果看起来会像什么样子。

所见即所得最适合用在图片、矢量图像、视频和动画，一些GUI Builder，一些“硬”格式化的文本上——不管在什么上下文中进行查看，最终产品看起来对每个人都一样。

但是现在想想HTML。当你设计网页的时候，可能没法全盘控制最终的产品。根据你的代码，页面可能要面对各种最终用户操作（如样式表、文本字号、窗口大小、浏览器差别、显示设备、布局偏好，甚至定制的内容）。如果每个读者看到的最终结果都不一样，所见即所得还意味着什么呢？这是一个有争议的问题。

还有，假设你正在设计一个产生格式化文本的文本编辑器，想想这一点：几乎每个使用Windows的人都会有Microsoft Word。这些Word已经让人们建立了对文本编辑器的期望。在另一方面，每个计算机用户都有邮件，而绝大多数邮件编辑器都会产生纯文本邮件（它们也应该这样，因为邮件基本上是一种纯文本的媒介）。然后，有上千万的Web页面在编辑消息板或Blog——绝大部分也是纯文本的，只是其中有些编辑器包括简单的基于Tag的格式化。即时消息通讯中使用的也是纯文本。

在任何一种给定的方式下，这个世界上有多少文字是通过Word编辑的，又有多少是通过纯文本编辑器编辑的？这个问题同样没有定论。别低估了从纯文本及非所见即所得编辑器中得到的可用性方面的好处——它们设计简单、使用容易。

直接操作 DIRECT MANIPULATION

“直接操作”意味着一个界面允许用户抓住、按下、拖拽、放开、重新定义外形、堆积、绘画、打开、关闭，或者操作界面上的对象。达到一种真实的感觉，这个是关键——尽管这些对象是虚拟的，对它们的操作是通过鼠标或其他设备来进行的，但用户还是会有一种直接操作这些对象的感觉。

而且这是一种强烈的错觉！下次你在一个装了Windows系统的计算机前坐下，在屏幕上贴上一个纸

质的即时贴。尽量不去管它。在某个时候，你的意识可能会忘掉那不是一个窗口，你会想用鼠标抓住它，把它移走。

如果设计得好，直接操作的反应应该很迅速——动作会立即发生。没有明显的等待时间，意味着实践中系统必须在少于0.1秒的时间内进行响应。而且，它应该很精确——用户需要好好控制鼠标指针的位置和该位置上可以操作的对象，尽管那在真实世界里是很容易的事。

直接操作也应该遵循习惯用法。前面曾提到过，人们假定他们可以把一些常用的技能从一个软件转换到另一个软件：例如，复制和粘贴，拖拽，通过句柄重新定义大小，通过画矩形框进行多选等等。

模态

MODES

界面的模态是一种状态，在这种状态里，动作的表现和正常情况下有所不同。例如，“单击-拖动”（click-and-drag）在一个画图程序里通常意味着“画一个矩形，选中这些对象”。但是在线段绘画的模态下，“单击-拖动”的意思是“从这里到那里画一根直线”。

有些人说界面的模态不好，因为它们很容易让用户陷入麻烦。其实并不完全如此。使用图形编辑器的人已经习惯了模态这种东西。事实上，如果只能使用鼠标和键盘操作，而且不使用模态，我不知道怎样去设计一个复杂的图形编辑器——如果你打算实现很多不同的绘图特性，那些输入设备将不得不重载很多功能。

不过，如果遇到以下情况，模态可能会带来问题：

- 用户不清楚界面已经在某种特定的模态里。
- 显式地打开或关闭模态带来不必要的额外工作。

第一个问题容易解决，我们可以利用充分的界面反馈。例如，鼠标光标应该总是代表当前的模态。用户的注意力将集中在光标所在的地方，因此这样的线索不容易错过。如果使用按钮来打开和关闭模态，记得让“打开”的按钮和其他按钮不同（例如，把它显示成按进去的状态，而不是向外凸起）。

关于第二个问题，本章有两个和模态相关的模式试图解决它：一次性模态（One-Off Mode）和弹性加载（Spring-Loaded Mode）模态。这两个模式都让模态容易关闭；而弹性加载模态让模态打开也很容易。把这两种模式和如下情形对比一下：用户不得不一直移动鼠标光标，移动到屏幕角落里的一个调色板工具条上，找到一个小按钮，单击它打开模态，然后把鼠标光标移动到画布上，画个对象，再回到调色板工具条，打开另一个模态，然后回到画布……无限循环下去。这么做需要大量的鼠标移动，也必然会带来重复性的肌肉劳损。

选择

SELECTION

表面上，选择似乎很容易设计。很久以前，就已经建立了各种选择的习惯用法，例如文本选择、列

表项目选择、图形对象选择等；人们已经习惯了这些用法。下面的表格描述了三组类型的界面中对象选择最常用的习惯用法：

- 文本编辑器，包括纯文本编辑器和格式化文本编辑器
- 基于矢量的图形，绝大多数画图程序和GUI Builders
- 列表、表格、树（在第6章中已经讲过了，但它们仍然出现在这个表格里，这样方便你对选择动作进行比较）

下面的表格总结了一般的选择相关操作中最常遇到的动作（不是所有的操作都影响选择，那些和选择无关的用灰色显示）。这些是你的用户最可能假定你的应用程序会遵守的习惯用法。

	文本编辑器	矢量图形编辑器和GUI Builders	列表、表格、树
单击	把文本输入光标移动到当前位置	只选择这个对象	只选择这个对象
双击	选中这个单词	打开或编辑这个对象	打开或编辑这个对象
三击	选择此行或选择段落	无	无
单击，拖住，放开	在开始的位置开始选择，在放下的位置结束选择，选中这两个位置之间所有的文本	选中这个方框内部所有的对象（又叫做“橡皮框（Rubber Band）”选择、“天幕（Marquee）”选择或“行进的蚂蚁（Marching Ants）”选择）	把这个对象从一个地方拖动到另一个地方
Shift + 单击	在文本输入光标位置开始选择，在单击的位置结束选择，选中这两个位置之间所有的文本	把当前对象加入已选择的集合中	把当前对象加入已选择的集合中，再加上所有它内部的对象（邻近选择）
Control + 单击	各个应用之间的定义不同	各个应用之间的定义不同	把当前对象加入已选择的集合中（不连续选择）

现在，好玩的事开始了。如果只实现这些习惯用法，就可以解决所有和选择相关的问题了吗？也许不能。想想用户可能想要执行的较难的任务。例如，假设：

- 用户在一个图形编辑器里选择了几个对象，想要让它们左对齐，但是对齐到哪个图形的左边呢？用户怎么知道第一个选中的对象还是最后一个选中的对象？有些应用会把第一个选中的对象标记成“主控”对象。它看起来和其他对象不同，例如它会带上绿色的选择句柄而不是蓝色的（见图8-1）。当对齐完成的时候，主控对象保持不变，其他对象会移动位置和它对齐。

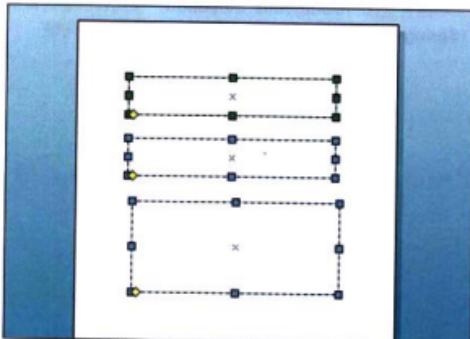


图8-1 Visio中的主控选择

- 现在让我们说用户想要选择图片里面的红色区域。这个区域有几百个像素范围，边缘也不整齐，要选中所有红色的像素非常困难。**智能选择（Smart Selection）**模式就是在这种情景下使用的。如果一个应用可以预料到用户想要选择大块内容，它可以通过自动选择这些内容来提供帮助。这个大块内容应该也可以是一段文本，或者一组图形。

不管在哪种情况下，都需要了解用户的常用任务。预先设想他们在传统选择中遇到的困难，增加你的选择设计来帮助他们解决这些困难。



模式

THE PATTERNS

本章的大部分模式都符合前面提到过的功能分组概念。就地编辑（Edit-in-Place）模式是一种所见即所得的文本和标签编辑手段。智能选择（Smart Selection）模式和组合选择（Composite Selection）模式讲的是选择机制。一次性模式（One-off Mode）模式和弹性加载（Spring-Loaded Mode）模式解决模态切换之间的问题。

- 79 就地编辑
Edit-in-Place

- 80 智能选择
Smart Selection

- 81 组合选择
Composite Selection

- 82 一次性模式
One-Off Mode

- 83 弹性加载模式
Spring-Loaded Mode

强制调整大小（Constrained Resize）模式、磁性吸附（Magnetism）模式和对齐指示线（Guides）模式讲述的是用户在面对直接操作的时候通常会遇到的精确度问题。所有这三个模式都可以帮助用户完成某个特定的任务——一开始就正确地调整一个对象的大小，在其他两个模式中把对象移到准确的位置。

- 84 强制调整大小
Constrained Resize

- 85 磁性吸附
Magnetism

- 86 对齐指示线
Guides

粘贴变种（Paste Variations）模式则提供了在应用系统中剪切、复制、粘贴时那些特别的设计问题的一种解决方案。所见即所得的编辑器往往比其他应用更需要这个模式。

- 87 粘贴变种
Paste Variations

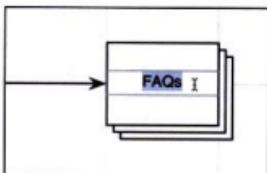


图8-2 OmniGraffle里的标签就地编辑

它是什么

使用小的动态文本编辑器，让用户可以“就地”修改文本：把这个编辑器直接放在原始文本的上面，而不是使用一个单独的面板或对话框。

什么时候使用

Builder界面有一些文本，有时候用户想要改变这些文本，对象名称、图形对象上的文字、标签，甚至属性值都是很好的应用对象。

为什么使用

让用户去另外的某个地方——去另一个窗口上空间比较远的地方，或者和原始文本没什么关联的地方——常常不是很好的做法。例如，用户可能找不到编辑器。同时，需要花时间来把用户的注意力从一个地方切换到另一个地方，而且也增加了界面的认知复杂度。

也就是说，这个模式的另一个可选方案是在另一个面板上编辑，例如在属性表（**Property Sheet**，参见第4章）上或对话框里。你应该只有在从技术上很难做到就地编辑的时候，或者需要编辑的文本很长又很复杂，值得专门提供编辑和格式化工具的时候才这么做——例如，需要改变字体的工具，改变文本大小的工具，以及那些你在文本编辑器工具条上能看到的那些。

如何使用

当用户单击，或者在更多情况下是双击要编辑的文本时，简单地用包含文本的可编辑文本框替换原来的文本。之后，用户输入的任何内容都将输入到这个文本框里。当用户单击其他地方的时候结束编辑。

确定文本输入框精确地出现在原来不可编辑文本相同的位置。如果在编辑开始的时候位置似乎有些跳动，这样可能会让用户生气（这种情形看起来并不严重，但它也可能会很严重）。还有，如果可能的话，保持和原文本相同的字体。简单地说，让它尽可能做到所见即所得。

通常当编辑器显示的时候，文本周围会有一个边框告诉用户编辑开始了。不过这也并不特别必要。其他一些线索可能已经足够了：应该会显示一个文本输入光标（通常是闪烁的），还有，如果要全部替换原始的文本，那么应该在编辑器出现的时候自动选中整个原始文本。

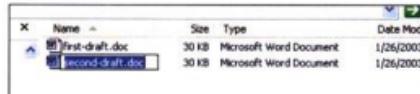


图8-3 很多人就是在Windows Explorer中第一次看到就地编辑模式的。Explorer不是编辑器，但它建立了人们如何同就地编辑模式交互的先例。Explorer和绝大多数编辑器不同，因为你是通过一次慢速双击（这么做通常不容易，也不推荐这么做）来调用就地编辑的，而其他编辑器使用普通的单击或双击。不过别的地方就完全一样了。之前的文本自动选中，文本周围显示出一个输入框，闪烁的文本插入光标出现了。



图8-4 Flickr也不是文本编辑器，但是它允许用户就地编辑图片的标题和描述信息。从交互设计的角度来看，Flickr有趣之处在于编辑过程的结束方式。绝大部分的应用会在用户单击其他地方的时候结束编辑，但是Flickr的设计师决定让它更明确一些：他们提供了Save和Cancel这两个按钮，动态地显示在文本输入框附近。用户第一次看到它们的时候也不会有什么麻烦（只要他们学会了首先单击文本）。



图8-5 Photoshop的魔棒选择工具选中了一块红色的区域

它是什么

使软件足够聪明，自动选择一组相关对象，而不是让用户自己去选择。

什么时候使用

用户在面对任何种类的可选择对象——文本、像素、几何对象，甚至树的项目和表格的单元格。可选择的对象组织成了相关的可视化分组，例如文本编辑器里的单词或句子、一组矢量图形，或者图片上一片颜色相同的连续像素。

正常情况下，选中上述的一组对象需要用户在某种程度上单击并精确拖动鼠标光标。那么做并不容易。例如，用套索选中所有红色的像素可能很困难，因为它们没有包含在一个整齐的矩形框中。

为什么使用

而且，不是所有的用户都擅长细微精确的鼠标运动。还有，有时候环境可能会让大家难以做到这一点，甚至不可能做到这一点（例如，笔记本电脑的触模板）。一旦软件知道用户在试图选择一组特定的对象，或者用户下意识地开始进行智能选择时——软件应该帮助用户，自动选择那组对象。让计算机做计算

机所擅长做的事，让人做人擅长的事。

如何使用

想想应该使用哪种UI操作来进行“智能选择”。这个操作可能是下述动作中的一个：

- 双击或双敲
- 在某个工具或某个特定模态下的单击，例如Photoshop里的魔棒
- 某个键按下时的单击
- 扩展选择的区域，直到这组对象的边界，例如很多文本编辑器都这么做

如果一个大一些的分组包含更小的分组时，考虑使用某种方式让智能选择选中所包含的分组——就像选择从一个单词扩展到一个句子，然后选择整个段落那样。

不过要小心。如果在设计的时候考虑不周，智能选择可能会让人觉得很恼火。想像一下，如果一个用户试图精确选择一组像素，但是每次他靠近一个颜色边界的时候，选择器突然变化，把他不想要的部分也包含进来了，这个时候他会有多沮丧。类似的问题也发生在一些文本编辑器里，参见下面的例子。



图8-6 Word在单词和文本行上进行智能选择。这些屏幕截图显示了各种操作的结果，鼠标单击和拖动（第一张），双击一个单词（第二张），三击整个段落（第三张）。

注意当整个单词选中时的扩展区域。你希望这些区域也被选中，对吗？不对？噢，那太糟糕了，因为当通过双击调用智能选择的时候，你不能不选择它们。如果你不想选它们，就必须自己单击，并在这个单词上拖动鼠标光标，小心地在不经意包含这个区域之前停下来。

这样的行为让Word踩到了真正有帮助和带来麻烦之间的分界线上。当你设计的软件在为用户作决定时，确保在绝大部分时间里，对绝大多数用户而言，在绝大多数环境下，你的决定是对的。



组合选择

Composite Selection

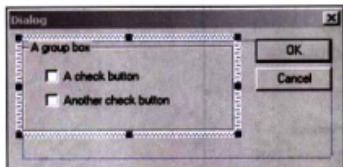


图8-7 选择一个Visual Studio的组合对象

它是什么

在不同的屏幕区域使用不同的操作——或鼠标单击，例如组合对象的边缘相对它的内部，来确定应该选择这个组合对象本身还是允许它所包含的那些对象也被选中。

什么时候使用

你的图形编辑器可能包括组合对象——可以移动或者操作那些对象，但是它们内部包含别的对象。这种情形在画图程序和GUI Builder之中非常常见。

为什么使用

当对象都显示在画布上时，一个单击通常会选择这个对象。在这种情况下，你想让用户能够在组合对象内部选中它的子对象（或创建一些新的子对象），但那意味着用户必须单击组合对象的背景。这种情况下，应该选中这个组合对象吗，还是不应该？鼠标单击有两种解释方式，反映了组合对象中的双重角色——父对象和子对象——就像组合对象所扮演的那样。

显然，其中一种解释方式必须胜出：当用户单击组合对象背景的时候，他需要能够预先知道会发生什么。

两种不同的选择方式——一种是选中组合对象本身，另一种是选择它内部的包含对象而不是它本身——解决了这个问题（虽然是以一种强制的方式做到的）。这两种选择模态很相近，但是对单击和拖拽那样的单击事件的响应不一样。

如何使用

Visual Studio（见图8-7）演示了对这个问题最常见的解决方案。除非在分组框的边框附近单击，不然用户不能选择它的分组框（在技术上，它并不能算组合对象，但是不管怎么样，它看起来就是）。在组合对象内部的单击将会作用在内部的那些对象上。如果是“单击-拖拽”，会出现一个套索；如果只是单击，内部包含的对象就选中了。也可以通过边框来拖动整个组合框；通过角落和边框上的八个选择句柄来重新定义组合框的大小。这样限制了直接操作，但是当用户知道具体用法的时候，这种机制很简单，也容易理解。

当一个组合对象选中的时候，以一种确定的方式拖动它的边框和重定义大小的句柄，就像图8-7中的带点边框那样。这样的视觉线索给用户提供了可预期的操作方式。



图 8-8 HTML里的表格是组合对象——它们通常会包括像文本、图片、嵌套表那样的对象。因此像Dreamweaver那样的HTML编辑器需要让用户在访问表格这个对象本身的同时，还可以访问它内部的对象。这里，左边的屏幕截图显示了在表格内部单击的情形——这样的单击会选中图片或文本。右边的图显示的是选中表格边框（从而选中整个表格对象）的情形。

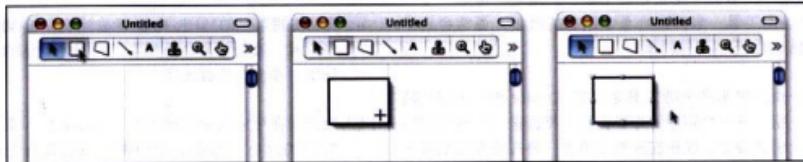


图8-9 在OmniGraffle里的一次性图形创建

它是什么

当一个模态打开的时候，执行一次操作。然后自动切换回默认模态或前一个模态。

什么时候使用

在你建立的编辑器里，正常情况下，用户不会重复或循环进行某些特定的操作，例如对象创建。通常，用户只会执行一次那样的操作，然后马上去做些别的动作，例如操作刚刚创建的对象。

为什么使用

用户会觉得切换到一个模态，做一点点操作，然后又显式地切换出那个模态很烦人。那么做通常会需要在很小的“目标”上单击，而这个目标还远在当前工作区间之外的调色板工具条窗口上，既需要时间，也需要精力。我们知道，在一个界面上，太多的“单击”就是一种刺激。

相反，界面本身应该让用户的工作变得更容易，哪怕那样做会使编程不太容易，或者在整个调色板工具条项目上不太一致。有好几个Builder程序就使用了一次性模态模式，因此你的用户可能会对这种做法很熟悉。

还有，模态很诡异。用户可能会忘记当前程序中作用的是哪个模态，然后进行一些没有意料到后果的操作。当使用一次性模态时，那种可能性降低了：用户不可能“堵”在那个模态里。

如何使用

实现一次性模态模式最难的部分在于决定应该将其使用在哪些操作上。对象创建通常会使用它。缩放、套索和画笔（除了编辑意义上的对象之外）则通常不会这么做。找出那些你的用户最熟悉的图形编辑器，看看它们是怎么做的。

你最好在确实知道了用户的工作习惯以后才去做这样的设计决定。如果他们的确想在一个行上创建几个类似的对象，然后每次创建对象时不得不回到调色板工具条上重新开始，这样会让他们很恼火。

很多使用一次性模态的应用会有一个该模态的秘密版本，允许多个操作（因而它的行为看起来像一个正常的模态）。你可以通过在单击模态按钮的时候按下一个键来触发它，例如Shift，但是这种操作在每个应用中可能都不同。

示范一次示例可能会让这个概念更清楚。考虑一下一个绘图工具，当用鼠标单击画布的时候，通常会选中鼠标指针下的那些对象。

1. 用户单击调色板工具条上的“Create Rectangle”按钮。过一个矩形的光标提示，界面进入一种特别的创建模态。现在在画布上单击，将在画布上放置一个对象而不会选中任何东西。

2. 用户单击鼠标按钮，确定左上角的位置。

3. 用户移动鼠标，并松开鼠标按钮，确定右下角的位置。
4. 界面支持著名的“单击-拖动-释放”用法：创建这个矩形对象，离开矩形创建模态，并回到默认的模态。这时，单击意味着选择。
5. 用户现在可以自由选择对象，移动它们，重新定义它们的大小，而不必回到调色板工具条去改变模态。

如果一开始没有注意到，单击-拖动-释放的逻辑，你可能觉得好像有点奇怪。但仔细想想，它其实非常自然。当你在使用绘图工具时，你希望在单击鼠标时，你的笔触能立即开始。所以，单击-拖动-释放的逻辑，是再自然不过了。

如果从头开始设计，你可能会认为单击-拖动-释放的逻辑，应该和“单击-双击”的逻辑一样。但问题是，如果这样设计，那么在第一次单击时，用户将无法知道他们是否已经触发了一个操作。如果第一次单击之后，用户必须再单击一次，才能开始操作，那么第一次单击就完全没意义了。

然而，如果从一开始，你就把这个逻辑设计到空气里，不要用户手动地去区分，那么这个逻辑就很容易理解。而且，如果用户第一次单击之后，就能立刻知道发生了什么，那么第一次单击的意义就变得非常大了。

当然，对于这个例子来说，单击-拖动-释放的逻辑并不一定是最合适的。但无论如何，如果把这个问题归结为“单击-拖动-释放”和“单击-双击”，单击-拖动-释放的逻辑显然要好一些。

然而，如果从一开始，你就把这个逻辑设计到空气里，不要用户手动地去区分，那么这个逻辑就变得非常容易理解。

当然，对于这个例子来说，单击-拖动-释放的逻辑并不一定是最合适的。但无论如何，如果把这个问题归结为“单击-拖动-释放”和“单击-双击”，单击-拖动-释放的逻辑显然要好一些。



图8-10 键盘上的Shift键

它是什么

让用户通过按住某个键或鼠标按钮来进入某种模态。当用户释放它时，离开这个模态并回到之前的模态。

什么时候使用

你在建造一个图形编辑器，或者其他任何广泛使用模态的界面。有时候用户临时要进入某个模态——他可能想快速完成某个任务，然后立即回到之前的那个模态。

你的用户希望去学习这个界面：可能会每天使用它。对于想要的那些功能，他们不介意去学习和记忆它们的快捷方式。

为什么使用

模态会导致的一个大问题是用户可能忘了他现在处在哪个模态。如果没有明显的状态指示器，例如一种特别的鼠标光标，他可能会在自己所认为的A模态里执行一个操作，而实际上他当时是在B模态里。这样可能导致意外发生。当然，你可以用一次性模态部分地解决

这个问题，还有，也可以通过持续的状态指示器来提醒他处在某个模态下（例如鼠标光标）。但是，只要用户在物理上按着一个键，他就不会忘记那个当时加载的模态。

但是弹性加载模态模式将为你带来另外一个好处：方便。再次说明，就像一次性模态模式，弹性加载模态模式减少了用户不得不在调色板工具条上单击一个小按钮，然后回到主工作区域的次数。在这里，不需要这样的往返操作——用户只要按下一个键，然后进行所有需要的操作，再释放这个键即可。一旦学会了，这么做就会毫不费力。它有助于快速完成操作，形成一种心流状态。

弹性加载模态模式的问题在于，它们是不可见的。界面上没有任何东西——没有标签，没有菜单项——来宣告它们的存在。用户必须艰难地学习它们，通过阅读使用手册或向其他人学习。

如何使用

它的运作机制很简单：当用户按下某个键的时候，启

用某个模态。不过，和一次性模态模式一样，最难的一点在于选择哪个模态值得采用弹性加载模式而不是一般的操作。了解你的用户，哪些任务需要特别暂时的模态。

和所有的界面一样，确认这个“秘密的”键盘操作并不是到达该模态的唯一方法。例如，可以为它添加一个调色板工具按钮，或者可以从工具条或菜单条触发它的替代方案。

示例

Shift键是经典的弹性加载模态例子——当键按下的时候，将输入某些大写的字母。释放它，会回到小写模态。和Caps锁键对比，哪一个更像典型的图形编辑器模态：按下打开模态，再次按下，关闭该模态。

Photoshop、Paint shop Pro和Fireworks（可能还有一些其他的绘图程序）都为滴管工具提供了弹性加载模态。为什么？想像一下你在试着用一个区域的颜色填充另一个区域。

1. 当处在画笔模态时，按下Alt键（或其他键，和平台有关），模态就会切换到滴管。
2. 单击你想要的那个颜色。
3. 释放Alt或其他键，用新颜色继续绘画（回到画笔模态）。

这样很快，也很方便。你的注意力仍然保持在画布上，也就是你所希望的地方，没有转到调色板工具条上。

（顺便要说的是，Photoshop中到处是这样的弹性加载模态。可以在图书或快速参考卡片上找到它们，但是你可能不会在随便看看的时候找到。）

还可以在面向列表的界面上找到另一种弹性加载模态的例子，例如在Windows Explorer、Mac Finder和邮件客户端中。你通常可以通过按下Shift或Control键来进入多选模态，然后选择多个对象。因为“单击”的动作有两种不同的解释——单选和多选——从定义上看，这是两个不同的模态。

如果你习惯了这么做，那么想想看，假设你不得不通过Explorer（或Finder）工具条上的一个按钮来从单选状态切换到多选状态，而不是按下一个键，那会怎么样？你将来来回回地跑到工具条去操作！如果你离开当前窗口，去别的应用上停留了一会，然后再回来，还能记得刚才离开的时候是在单选模态还是多选模态吗？不一定吧，然后你可能不会马上意识到当单击文件名的时候会发生什么事。你不知道新的选择会替代旧有的选中对象还是会选中多个对象。

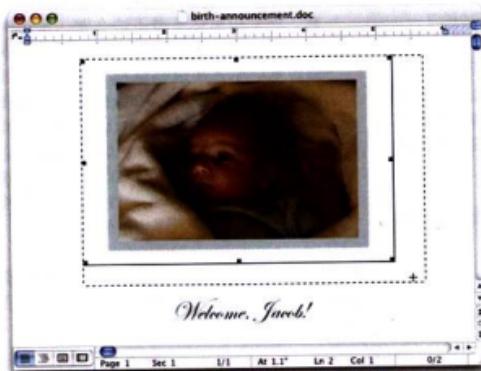


图8-11 调整图片大小：当拖拽某个角落顶点时，Word保留了图片的长宽比例

它是什么

为调整大小模态提供不同的行为，例如保持原始比例，以便在特别的环境下使用。

什么时候使用

你在建立一个图形编辑器，它可以让用户交互式地调整对象大小，例如通过顶点拖动。但是有时候用户想要保留一些特定的属性，例如对象的原始比例——特别是对图片和格式化的文本来说——或对象的几何中心点。正常的顶点拖动难以实现这样的任务。

为什么使用

很简单，这项技术可以节约用户大量的工作。如果UI强制调整大小按某种特定的方式工作，例如强制调整宽和高来保持彼此的比例，那么用户可以把注意力集中到他想要的对象外观上。

如果用户对界面没有这种程度的控制，他可能会通过往一个表单输入数字来做到这一点。那样也可以，但是往往没有直接操作那么好。它会打断用户的心流状态，迫使他去进行数字化思考，而不是可视化地思考——不是每个人都擅长数字化思考。直接操作可以更好地支持递增构建（**Incremental Construction**，参见第1章）。

如何使用

这项技术基本上就是一个修改过的调整大小模态，因此它的行为应该接近正常的调整大小模态——通常是拖动目标对象的顶点或边框来实现大小调整的。那你应该怎样区分这种方式和正常的调整大小模态呢？考虑一下使用修改键：很多应用会在拖动对象顶点时通过一直按下Shift或功能键来进入强制调整大小模态。

或者，如果你认为绝大部分用户通常都想要这样的强制调整大小，那么让它成为默认的调整大小操作。Word中图片和剪贴画就是这么做的，如图8-11所示。这样，用户就不会很突然地看到压扁或拉长的脸了。

当用户拖动鼠标光标的时候，应该画出一个表示大小调整的框来显示新的方向和/或位置。该框让用户能很快看到当前操作的反馈：如果用户反应很快，他甚至可以从这个表示大小的框中看出这是不是强制的调整大小模态。

可以用几种方法来强制调整大小。下面就是几种可能的方法。

- 保持对象的原始比例（也就是长宽比例）
- 保持对象的几何中心不变，然后围绕着它对称地调整大小

1. 见 *Face 2.0: The Essentials of Interaction Design* (Wiley出版社)，第311页。

示例

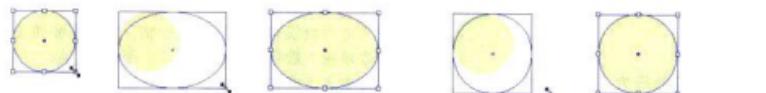


图8-12 Adobe Illustrator至少有两种强制调整大小模态：一种是保持对象的原始比例，另一种是保持对象的几何中心。这张图显示了一个正常的顶点拖动调整大小操作，包括“调整之前”（有可以抓住的句柄）、“调整中”和“调整之后”的三个状态。蓝色的轮廓会随着用户的顶点拖动实时变化，显示了对象的新大小和新位置，以及用户是否可以释放这次拖动。

- 用大于像素的单位来调整大小。例如，一个GUI Builder可能想让一个列表框的高度是文本行的整数倍（总的来说，这可能不是一个好主意，但是如果你不得不这么做，那么UI应该反映这一点）。这样的话，调整大小的默认行为就会从一个大小级别跳到另一个大小级别。与此类似的是，图形Builder的画布应该使用一种“贴在网格线上”(Snap-to-Grid)的机制；还有，调整大小的动作应该在网格之间跳跃。
- 限定对象大小。一旦用户到达了某个维度的大小限制时，别让大小调整的方框在那个维度上继续放大（或缩小，视具体情况而定）。

Alan Cooper和Robert Reimann把这项技术叫做“Constrained Drag”（强制拖动）¹。它的历史至少可以追溯到20世纪90年代中期。

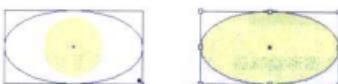


图8-13 这些屏幕截图显示了两种强制调整大小模态。当用户一直按着Shift键的时候，原始比例保持模态将起作用。当用户反过来一直按住Alt键的时候，对象的大小可以在两个维度之间自由调整，但是它的中心会保持不变。

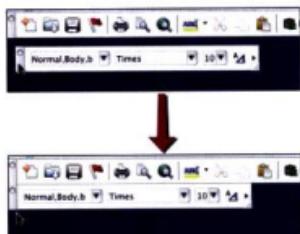


图8-14 贴在某个地方的Word工具条

它是什么

让对象具有“磁性”，可以吸住靠近它们的其他对象。当用户拖动一个对象到这种“磁性”目标对象附近时，它就会贴在目标对象上。

什么时候使用

用户需要把对象放在非常精确的位置上，例如紧贴在其他对象之后或靠近对齐指示线（Guides）。当然，这样的情况经常发生在图形编辑器里。但是它们在窗口管理器和桌面框架上也很常见，例如，在用户需要移动窗口和工具条的时候。

为什么使用

磁性吸附模式可以弥补用户在精确操作鼠标上的不足。如果用户确实想把移动的对象紧紧贴着另一个对象——你需要确认事实如此——那么计算机可以通过这个模式帮助用户做到。

磁性吸附模式的行为是让鼠标的“目标区域”变大，比它

真实的区域更大。如果用户试图把一个对象精确地放在另一个对象旁边，那么目标区域将只有1个像素宽；如果超过1个像素，要么会重叠在一起，要么会彼此分开。这样她就需要让这个位置刚刚好。但是1个像素的目标实在太小了。如果当用户进入区域的时候，例如离边缘4个像素远的地方，磁性吸附模式就把对象吸附在目标位置上，那么目标区域的大小就有 2×4 个像素（也就是8个像素）那么大。这就容易多了。

还有，这样也快多了。这样的机制还能帮助那些确实需要精心把对象放在区域宽度为1个像素的准确位置的用户。磁性吸附模式节约了他们的时间和精力，不然，他们就需要亲自去做到这一点。而且总体上说，这样会让应用的响应性更好，更加有用。²

如何使用

当用户把一个对象拖动到另一个对象的边缘时，让它贴近那个对象。与此类似，如果把一个对象拖离另一个对象时，在几个像素内让它保持不动，然后把它移开。

2. Martijn van Welie首次在他的“在线”（Online）模式目录里讲到了这种模式：<http://www.welie.com/patterns/gui/magnetism.html>。

可以具有磁性的对象包括：

- 和移动对象类别相同的对象，例如窗口和窗口、形状相同的两个对象
- 画布边缘、空白地区、屏幕边缘

- 对齐指示线（Guides）和格线——用来严格对齐各个对象的设备
- 图片层上容易察觉到的“硬边缘”，例如Photoshop里那样

示例

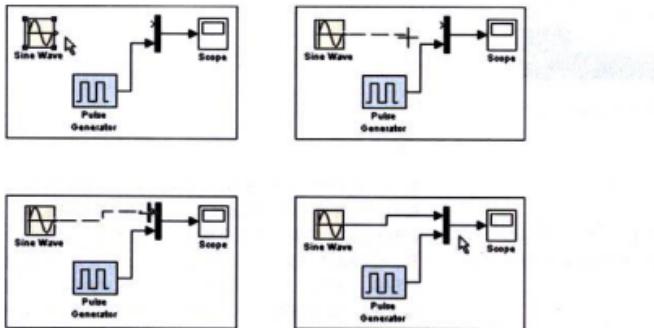


图8-15 图表Builder和可视化编程环境有时候会使用磁性吸附模式帮助用户把图标连接起来。在这里，你可以使用Simulink应用把一个微型信号发生器模拟组装在一起。用户需要把正弦波的输出部分和一个多路复用器的输入部分连接起来，如黑色的竖直长条所示。

如果没有磁性吸附模式，这个任务需要用户把鼠标放在非常微小的目标对象上，单击鼠标按钮，然后拖动并把对象连接到另一个同样非常微小的对象上。不过，源输出端口通常比这里显示的几个像素要大。当用户拖动连接——用一条虚线表示——到多路复用器的输入端口时，这个连接在鼠标靠近目标对象大约10个像素的时候就贴在输入端口上了（Simulink甚至自动把顶点放到连接上）。在最后一个画面上，用户释放了鼠标按钮，完成了这次连接。



对齐指示线 Guides

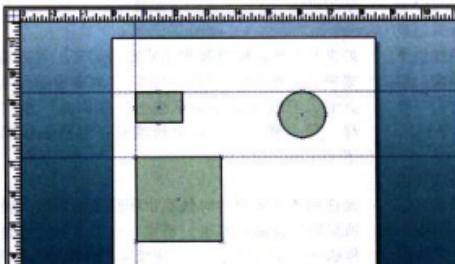


图8-16 Visio里的对齐指示元素

它是什么

提供竖直方向和水平方向的参考线，帮助用户对齐各个对象。

什么时候使用

用户需要让各个对象彼此之间精确对齐，或者和页面边缘对齐。这种情况经常出现在基于矢量的画图应用、GUI Builder、页面布局编辑器和图片编辑器中。

为什么使用

和磁性吸附（Magnetism）模式有点类似，对齐指示线模式可以弥补用户在精确操作上的不足——它们帮助用户保持对象对齐。还有，它们可以让用户更容易看到已经完美地对齐到像素级了。在放大倍数较小的时候，如图8-16的Visio例子所示，用户不能肯定这两个矩形已经完美地对齐了，但是对齐指示线模式可以保证这一点。

有些对齐指示线的实现会在它自己移动的同时移动已经对齐的那些对象。这样很好，因为用户原来的想法就是一定要把这些对象对齐到同一条直线上，通过自

动移动这些对齐的对象，对齐指示线不需要用户再次把这些对象拖动到新的对齐位置上。

你不一定要使用指示线作为传统对齐工具的补充（例如让选中的对象底端对齐，或者让选中的图形按竖直方向中心对齐），因为这些工具对于复杂的对齐任务来说更快，不过，指示线在直接操作时表现更好。

最后，指示线还可以在同一文档的多个作者之间传递布局信息。如果Amber开始了一个复杂的页面布局，她可能沿着页面布局的边距和格线创建了对齐指示线。然后她把这个布局交给Bert来完成，这时候Bert就可以看到这些对齐指示线。

如何使用

对齐指示线是漂浮在画布上的参考直线——它们不属于当前文档，看起来也不应该像当前文档的一部分。绝大多数的指示线颜色很鲜艳。一些是红色的，一些是蓝色的，还有一些是草绿色的；这么做的原因是能够清楚明白地看到它们，不管它们底下有什么对象。要达到它们的效果，这些线并不需要有宽度，因此一个像素宽就可以了，并且不需要随比例放大。

让用户创建水平方向和竖直方向的对齐指示线，并且，如果愿意的话，让他们自由地创建多个指示线。很多应用在画布周围提供直尺，如图8-16的Visio例子所示，指示线常常通过拖动直尺上的某个东西来创建（例如左上角的两条蓝色交叉线）。其他应用提供菜单项或工具条方式创建对齐指示线。因此，再次，你可能想要在任何有对齐需要的时候自动创建它们（参见图8-18中Interface Builder和图8-19中Word的例子）。

任何时候用户想要调整布局，都应该可以拖动这些指示线，因此考虑让它们在很多不同模态里面都是可以拖动的，而不是某一个模态里。还有，考虑一下指示线的锁定，这样你就不会意外地移动它们。一些应

用，包括Visio和Visual Studio，让用户可以把对象粘到指示线上，因此这是一种半永久的粘附：当指示线移动的时候，这些对象也会跟着移动。

如果和本章的磁性吸附模式结合起来，对齐指示线模式将会工作良好。当用户把一个对象移动到指示线附近时，它就会贴在指示线上，并和指示线对齐。这样，用户要把一个对象精确对齐到指示线就容易多了。

允许用户在需要的时候显示和隐藏指示线，特别是在所见即所得编辑器里。毕竟，它们本身并不是当前文档的一部分。而且，它们可能会导致视觉上的混乱。

示例

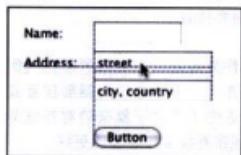


图8-17 很多GUI Builder，包括Visual Studio和Mac的Interface Builder，都提供了对齐指示线。在Interface Builder里，当用户在画布上拖动元素的时候，蓝色虚线的对齐指示线会自动出现。这些指示线带有磁性，有效地鼓励用户在放置控件的时候让它们彼此对齐——当你看到指示线出现的时候，就能知道是不是已经对齐了。

在这个屏幕截图里，“Street”文本框已经被拖动到几乎对齐其他文本框及“Address”标签的地方了。这几条指示线有效地帮助用户把文本框再推动几个像素来到达正确的对齐位置。



图8-18 Photoshop的对齐指示线是青色的直线。这些指示线是由用户显式创建的（不是自动创建的），用户可以通过对话框把它们放在非常精确的位置上——这么做在某些情景下非常有用。用户也可以拖动它们。如本图所示，和图片相比，指示线没有什么物理上的宽度，因此在放大图片的时候，指示线仍然保持一个像素的宽度。

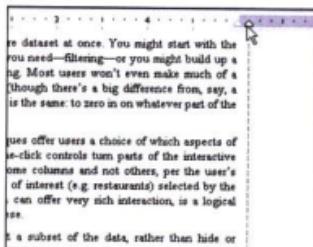


图8-19 Word的对齐指示线是自动创建的，它控制了边距和缩进的位置。当它们移动的时候，会改变文字在页面中的填充方式。如果不从顶部的直尺上开始拖动，Word的指示线就不会显示出来。



图 8-20 Paint Shop Pro

它是什么

在标准的粘贴操作之外，另外提供专门的粘贴功能。

什么时候使用

用户可能从剪贴板粘贴某些内容到应用里。但是他也许想对它执行几项任务，每项任务都需要一个有点不同的粘贴操作——例如，他可能要粘贴到不同的格式，或者粘贴到一个新的目标文档而不是现有的文档中。

这个模式很少用到。只有几个应用会需要，但是对一些设计难题来说，它是一种可行的解决方案。

为什么使用

有时候没有办法：你不得不支持两种或两种以上的粘贴相关任务，并且没有足够的智商让你把它们综合到

一个超级聪明的粘贴操作里。你可以只选择提供其中一种方式，这样能让设计变得简单。但是执行另一种任务的用户就没那么好运了。他们将不得不手工做很多工作来完成另一种粘贴。

所以你需要提供两种或者两种以上的粘贴。其中一个会成为主粘贴，因为当用户按下Control+V的时候（或其他任何平台设定的粘贴快捷键），某种合理的操作必须进行——不能在按下Control+V的时候弹出一个菜单或对话框。那样做会打断用户即时操作的期望。不过，你可以在默认的粘贴行为之外，以其他方式提供一些选择。

如何使用

在编辑菜单条上，除了默认的粘贴操作外，把其他粘贴变种也显示出来。用更多的详细信息来标注它们；把它们的确切行为描述出来。你可以直接把它们放到编辑菜单里——这样可能是最容易使用的——或者你可

以把所有的粘贴操作，包括默认的粘贴放在一个级联菜单或直接放在编辑菜单里（不过，级联菜单需要进

行讨厌的精确鼠标运动，因此只在你需要减少顶级编辑菜单项的时候才这么做）。

示例

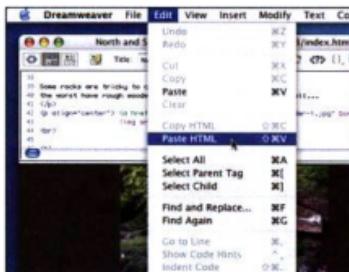


图8-21 Dreamweaver是一个复杂的网站构建应用。你可以通过一个所见即所得的网页编辑器或一个显示HTML代码的文本编辑器来编辑网页，或者同时在这两个编辑器里编辑，并把它们显示在同一个窗口上，如图所示。Dreamweaver会让这两个编辑器保持同步。在这个屏幕截图里，我复制了一小片HTML代码到粘贴板。

现在，如果我把这些代码粘贴到所见即所得的那个编辑器，会怎么样呢？事实上，刚才粘贴的只是文本。它有一些尖括号标记，但是难道不应该按字面上复制，让你在网页上看到“`<h4>`”的字样吗？还是说应该把它解析成一个第4级的header（标题）？还有，如果它是一个定制的Tag，Dreamweaver根本就不认识，例如`<customheader>`，又会怎么样呢？

也许是，也许不是。Dreamweaver不会明白我们的想法。所以它提供了一个粘贴变种：“粘贴HTML”。如果我们选中了这个菜单项，Dreamweaver会假定它是一些代码，并且不会在网页上显示这些Tag。如果使用了默认的粘贴操作，Dreamweaver会把它当成文本，并在网页上显示Tag。

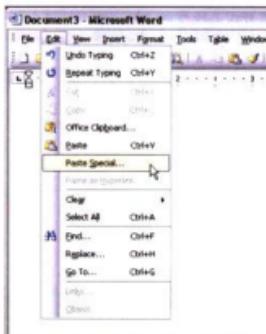
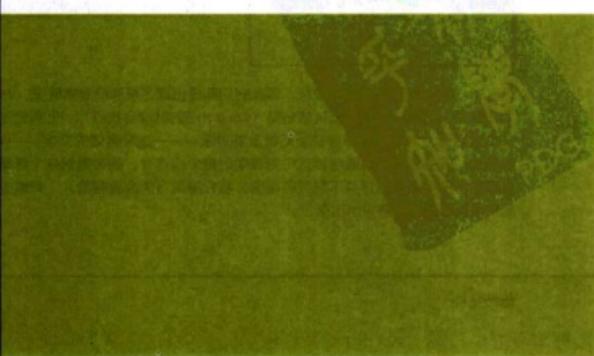


图8-22 Word的粘贴变种很复杂，因此它们需要出现在单独的对话框里。Word的编辑菜单显示了两个粘贴项：粘贴，也就是默认的粘贴（Control+V的时候会执行）；还有特别粘贴，这个粘贴会打开一个对话框。对于现在保存在粘贴板上的对象——一些带粗体的文字——Word不会列出少于7个目标格式。但是当用户有意选择这个列表中的每个格式时，她会看到这个结果的一个简要描述（这里显示的例子中，该描述并不是很有帮助，但它确实可能有帮助）。有趣的是，这个对话框里也把剪切板项目的源内容显示出来了。

09

修饰外观：视觉风格和美感

**MAKING IT LOOK GOOD:
VISUAL STYLE AND AESTHETICS**



在2002年，一个研究小组得到了一些有趣的发现。斯坦福Web可信性项目¹开始了解是什么因素让人们对网站产生信任或者不信任，而很多因素在他们的发现中都很明显：公司声誉、客户服务、出资人，还有广告，这些都会让用户决定该网站是否值得信任。

但是最重要的因素——在他们的列表中排在第一位的——却是网站的外观。用户不信任那些看起来很不专业的网站。那些努力提供美观的、专业设计的网站给用户留下了很好的印象，哪怕用户并没有其他很多理由来信任这个站点。

这里还有另外一条数据补充说明了这一点。Donald Norman，交互设计领域一名非常著名的大师，总结说：“积极的情感增强了创造性和广度优先的思考，而负面的情感集中在认知上，增强深度优先处理并把干扰降到最少。”他还补充说：“积极的情感让人们更能容忍一些困难，在寻找解决方案的时候变得更灵活而有创造性。”²当人们喜欢某些界面的时候，它们事实上会变得更可用。

看起来美观很重要。

到目前为止，我们已经花了很多章节来讲述结构、表单，以及应用系统的行为。现在我们要把注意力集中在“外表”上，或者说“外观和感觉”（Look-and-feel）上。第4章“布局”，讲述了一些图形设计基本知识。包括视觉层次结构、视觉流、焦点，格士塔原则中的相邻性、相似性和连续性，以及闭合性原则。这些话题组成了页面组织基础，应该好好学习。

但是一所很好的房子除了布局之外还有更多的因素。当你购买一所设计良好的新房子时，你还希望它有漂亮的地毯、粉刷颜色、墙饰和其他外观处理。没有这些，可能房子在功能上非常出众，但是并不会让人心动。要完成整个工程意味着需要注重细节，在调整和修饰上付出努力。

漂亮的细节并不一定会影响人们在这所房子里（或界面上，尽管研究表明有时候确实会）完成任务的效率。但是它们确实会影响人们的心情。反过来，影响到其他行为——例如，他们会在这里逗留和探索多长时间，是否选择再次回来访问，以及是否会推荐给别人。

你甚至可能会把它看成一个道德问题。你希望你的用户会有什么样的体验？你想把应用弄得灰蒙蒙的让他们觉得很无聊，还是充满了飞来飞去的广告，让他们觉得很恼火？还是愿意为他们提供一些他们喜欢，甚至会看上几个小时的东西？

1. 参见<http://credibility.stanford.edu>。

2. 参见Donald Norman的“Emotion and Design: Attractive Things Work Better”一文，地址在<http://www.jnd.org/dn.mss/Emotion-and-design.html>。还参见他关于这个主题的书*Emotional Design: Why We Love (or Hate) Everyday Things* (Basic Books出版社)。

当然，除了视觉风格之外，还有很多因素会对用户的情绪反应（情感）产生影响。第1章讲到了一些其他方面，例如你应该预计他们的使用习惯。软件可以用贴心的设计给人们带来惊喜。满满地挤在一起的布局和宽松开放的布局会触发人们不同的情感反应。语言和语气在这种反应里影响很大，软件本身的质量也是——它是可以工作的，还是又快又反应灵敏的？

设计良好的界面需要把这些因素都考虑进去。当内容、含义和交互式的行为都与你的视觉风格和谐一致的时候，你可以非常有效地唤起某种意想之中的情感反应。

产品和网站的风格通常会用来支持品牌。任何软件产品或网站的设计都表达了某种关于创建它的组织的信息（哪怕它是开源软件开发人员的松散组织）。它表达的意思可能很中性，或者传达出一种强烈的信息，如“我们值得信赖”，“我们很酷”，“我们的产品令人激动”。品牌识别不止是图标和广告词。它遍布在一个组织的产品设计、它的网站，以及它的广告材料中——事实上，品牌对配色方案的选择、字体、图形、词汇表，体现在各个地方。如果规划得好，一项完整的品牌识别是有凝聚力的，而且具有明显的意图。

品牌识别很重要，因为它建立了熟悉感，并为人们对这个公司的产品体验设定了期望。最终，一个良好的品牌应该让人使用该品牌产品的时候感觉良好。看看苹果公司在品牌忠诚度上的成绩：很多人喜欢苹果的产品并求之若渴。

在任何情况下，不管它们是否有意对品牌进行增强，风格元素都在表述你的产品。它们传达了各种产品个性，例如可靠、令人激动、好玩、充满活力、冷静、有力、紧张、快乐等。你想让它们传达什么？

本章讨论更多的视觉设计概念，这次不会再那么关注视觉结构，而是更多地关心它们基于情感的特性。本章不会让你成为一名艺术家——那可需要严格的训练和学习。但是，本章会讲述一些可以在设计良好的产品中找到的模式，并解释它们的运作原理。

内容相同，风格不同

SAME CONTENT, DIFFERENT STYLES

要了解风格会引发什么样的内在的及情绪上的反应，我们可以试试对同样的内容应用不同的风格。实际的内容并不是那么重要——我们是在寻找即时的、下意识的反应，而不是通过对内容进行阅读和交互之后的印象。

“CSS Zen Garden”（CSS禅意花园）网站 (<http://csszengarden.com>) 正好为我们提供了这样的机会。为了展示基于CSS的网站设计，它为所有的参与者提供了一个HTML页面——每个人看到的正文内容、HTML标签、链接列表都是一样的。然后，各位参与者自己创建独特的CSS文件来为这个页面定义新的视觉设计，并把它们发给网站。网站的访问者可以浏览所有的CSS设计。在这个网站上浏览两三个小时是一个开心的旅程，特别是如果你在自学视觉设计，在试图理解自己喜好的时候。

下面八张图片展示了了几份CSS设计。在每份设计里，基本的内容都是一样的。只是设计有所变化。花一点时间来观察每份设计。当你看到每份设计的时候，你的直觉反应是什么？你的大脑里会闪过什么样的词汇来形容这个页面？它让你觉得亲近，还是反感？它让你觉得开心，还是不舒服？

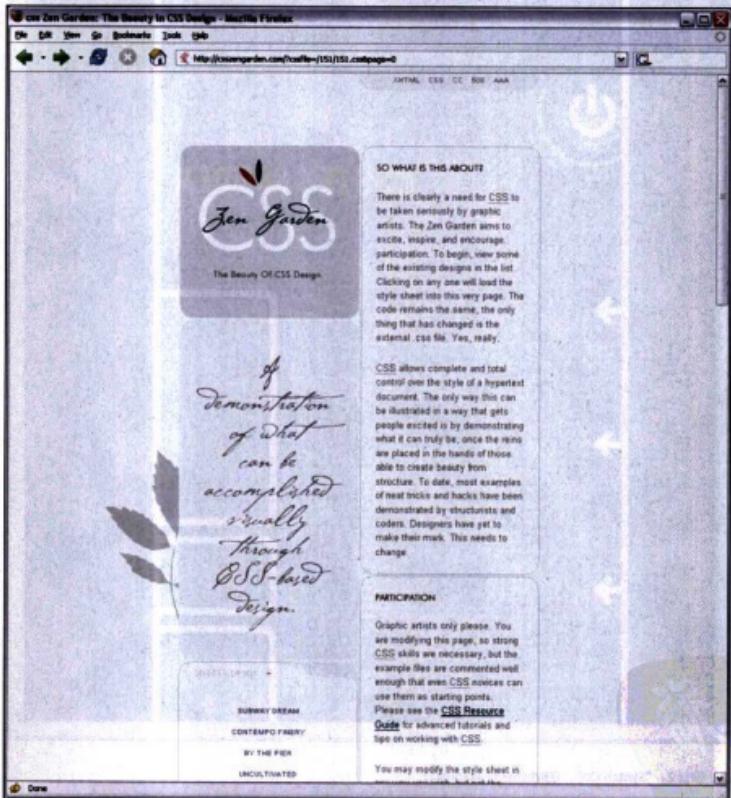


图9-1 设计1：“Contempo Finery”，由Ro London设计



图9-2 设计2：“Switch On”，由Michael Fasani设计

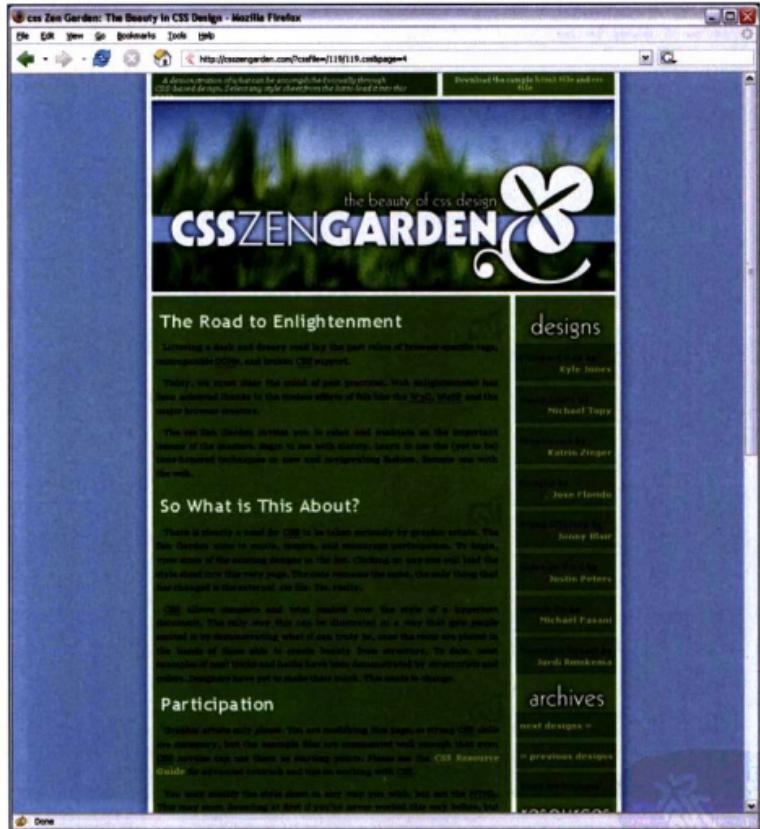


图9-3 设计3：“Pleasant Day”，由Kyle Jones设计

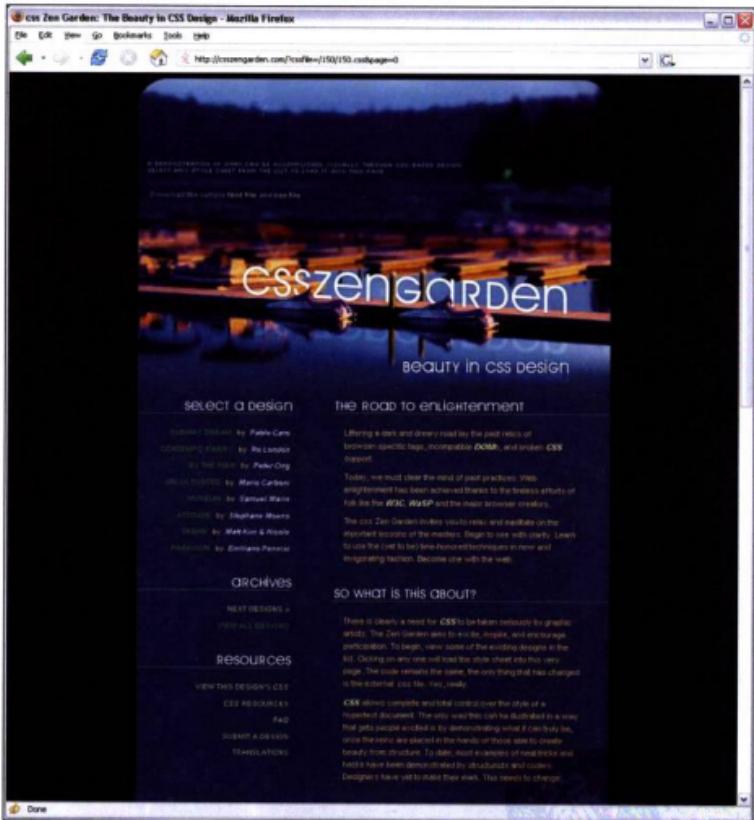


图9-4 设计4：“By the Pier”，由Peter Ong设计

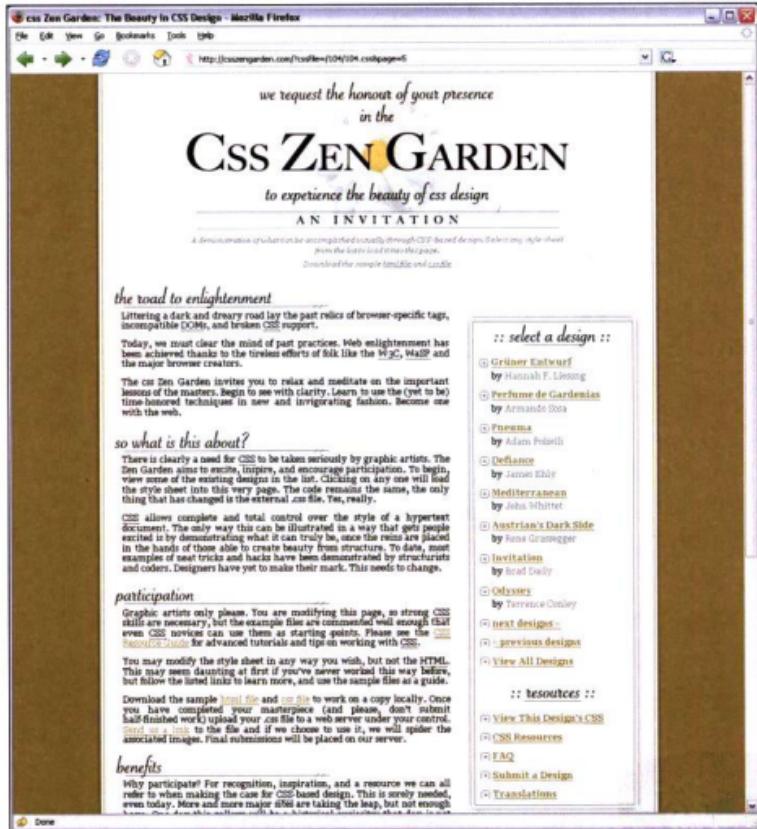


图 9-5 设计5：“Invitation”，由Brad Daily设计

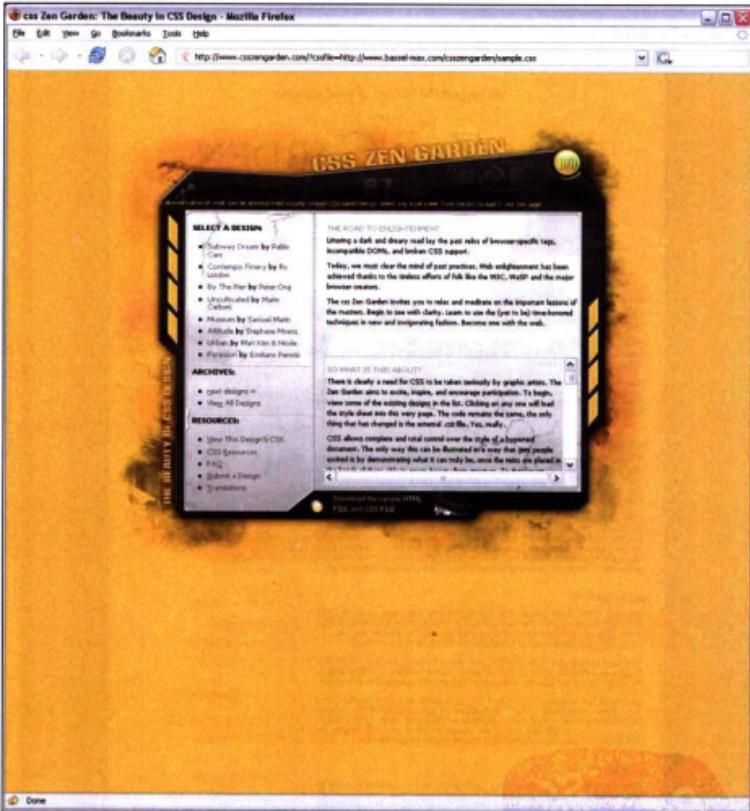


图9-6 设计6：“Yellow Jet”，由Bassel Max设计

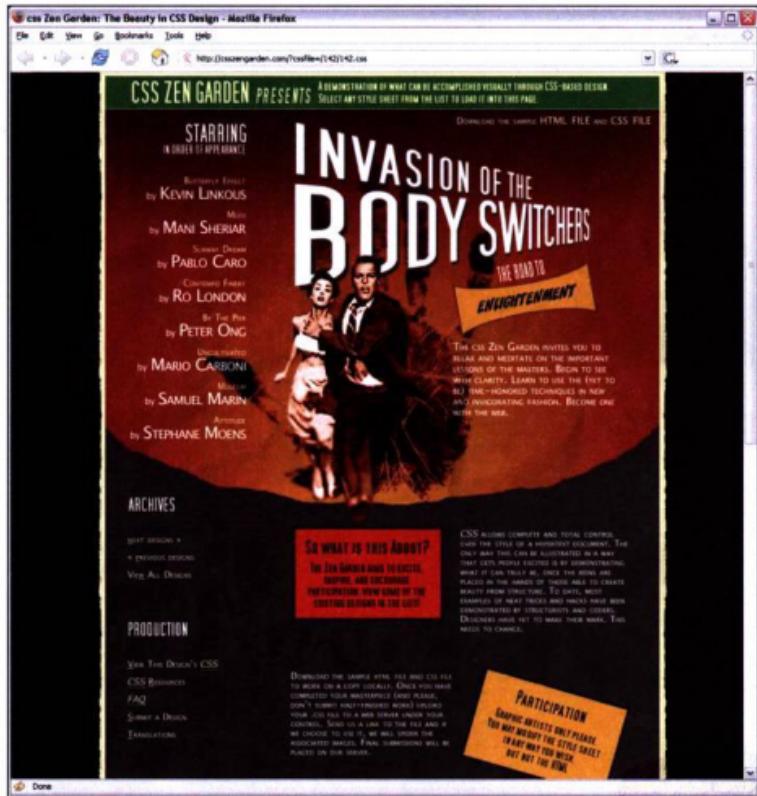


图9-7 设计7：“Invasion of the Body Switchers”，由Andy Clarke设计

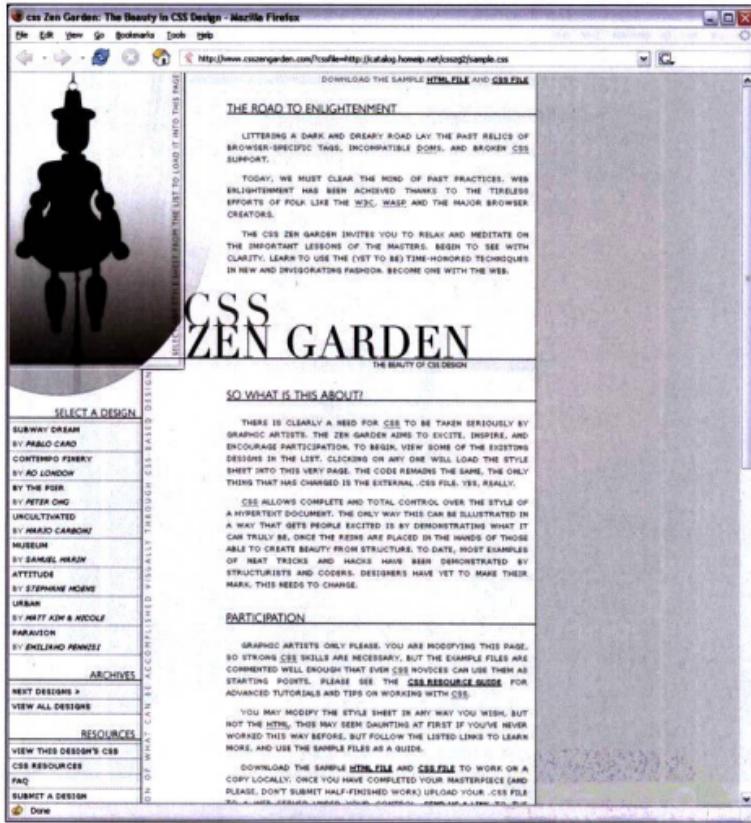


图9-8 设计8：“Rain”，由Pierre-Leo Bourbonnais设计

当看到CSS禅意花园示例的时候，你可能已经注意到他们是怎样达到这些不同印象的——例如，页面的配色方案会让你微笑或者厌烦。把这些示例作为试验，我们可以来看看好的视觉设计有些什么原则。

你可能还记得前面我们已经讲述了一部分视觉设计原则，第4章的页面布局，以及第6章的信息图形。这些章节探索了人类的视觉系统对特定输入的认知反应。例如，某个人花在蓝色区域上单击一个橙色方块需要的时间并不依赖于他的审美感觉或文化期望。

但是现在我们讨论的是情感上的反应及内心的反应——一个橙色的方块会给设计增加压力、智慧、平衡或者什么作用都没有？这个问题的答案和很多因素有关，因此如果没有大量实践，很难得到好的方案。对于这些设计选择的认识当然会起到一定作用：对于初学者来说，你可以让一个页面阅读起来更容易，或者更困难（认知效果）。但是每个人都是独特的个体，都有不同的以往经验、各种关系，还有个人偏好；每个人也是他所在文化的一部分。文化会在颜色、排版和图片上有它们自己的含义。

还有，设计的上下文也会影响到用户的反应。用户认为你的设计属于某种类型（例如Office应用、游戏，或者电子商务网站），他们会对设计有一些期望，合适不合适，陈旧还是新奇，沉闷还是有趣。品牌也会设立期望。因此有这么一个问题：一旦你学到了一条“规则”来使用一项设计原则引发某种情绪反应，就可以找到无数不符合这条规则的例外情况。

也就是说，如果你很了解目标用户，那么他们的内在反应和情感反应是可以预见的，而且可以预见的程度相当惊人。例如，本书的绝大部分读者可能会认为第1个CSS示例很平静，让人心情舒坦，而第2个设计看起来很嘈杂，更加充满压力。为什么？

答案在于实际情况下很多因素的结合：颜色、排版、规模、角度和形状、重复出现的视觉主题、文本、图片和文化因素。

颜色

COLOR

我们对颜色的反应很快。当你看到一份设计时，最先看到的就是颜色、基本的形状和外观。不过，颜色在艺术和设计中的应用非常微妙——绘画大师们已经研究了几个世纪。我们在这里只能描述一下它的皮毛。

在为界面设计色彩方案的时候，最最重要的一件事就是让它具有可读性。

- 总是在深色的背景上放置浅色的前景对象，或者反过来——要测试，把这个设计放到一个Photoshop那样的图像工具中，并让它饱和（让它变成灰度图片），看看效果怎样。
- 永远不要采用红色和绿色来区分重要的元素，因为很多色盲患者将看不到它们的区别。统计数据表示，10%的男士和1%的女士有某种程度的色盲。
- 永远不要在明亮的红色和橙色背景上显示蓝色的小字，或者反过来，因为人眼不容易阅读这种互补色（在色彩轮上位于相对的位置）之下的文字。

除了上面这些原则，这里还有几项非常类似的色彩使用指南。

冷色调和暖色调

人们认为红色、橙色、黄色、褐色和米色都是暖色调的颜色。而蓝色、绿色、紫色、灰色（各种灰色）及白色都是冷色调的颜色。在CSS禅意花园的第6个设计中的黄色让人觉得很温暖，除了内容后面灰色金属光泽的冷色之外。需要表示尊敬、保守的网站和界面常常主要使用冷色调（特别是蓝色）。不过，暖色调和冷色调的颜色可以有效结合在一起达到平衡的效果——在经典的印刷和海报设计中，常常就是这么做的。

深背景和浅背景

如果采用浅色的背景——白色、米色和浅灰色，页面的感觉和深色背景的感觉完全不同。浅背景更常用于计算机界面（和印刷页面）；深色背景的感觉更深刻、更阴暗，或者更有活力，依其他设计因素而定。实际上，深色背景很少用在表单风格的界面中，因为桌面控件例如文本输入框和按钮在深色背景中不是很好看。

强对比和弱对比

不管背景色是深还是浅，在这个背景中的元素对它来说要么对比很强，要么对比较弱。强烈的对比会让人觉得紧张、有力和大胆；而弱的对比会更平滑，更让人放松。

饱和色和不饱和色

高度饱和的颜色，或者说纯色——例如，明亮的黄色、红色和绿色——让人觉得生动、明亮、有力、温暖。它们很大胆，并且色泽鲜明。但是如果过度使用这些高饱和的颜色，会让视觉疲劳，因此绝大多数的UI设计师们都会保守地使用它们：他们往往只选择其中一种或两种。而暗淡的颜色，不管是深色还是浅色（或者说不同的色调或色彩）构成了调色板中的大部分区域。“green-and-blue”的CSS禅意花园的第3个设计通过使用白色的边框，白色的文字及深色的渲染来分离蓝色和绿色，这样就让两个互相融合的颜色分开了。（就算这样，你可能也不想在一个桌面应用中整天对着那片绿色。）

色调的结合

一旦开始结合多种颜色，就会出现一些有趣的效果。两种饱和的颜色比单独使用一种更有表现力，更动感，更丰富。如果一个页面上结合了一种饱和的颜色和一组不饱和的颜色，将把读者的注意力直接吸引到那个饱和的颜色上，并建立颜色层次——更明亮更强烈的颜色看起来离读者更近，而灰色和暗淡的颜色将离读者更远。强烈的色彩层次会让设计非常生动。柔和的设计，使用更多模糊的或浅的颜色，会让设计更平静。下面的少一点颜色，多一些价值（Few Hues, Many Values）模式将进一步讨论这个问题。

排版

TYPOGRAPHY

通过为文字选择字体（“font”，叫做“字样”（**typeface**）更合适），你就决定了这些文字将“发出”哪种声音。可能很嘈杂，或者很柔软，很友好，或者很正式，很通俗，或者很权威，很新潮，或者很保守。

和颜色一样，在选择字体的时候首先要考虑可读性——认知部分。小的文字，或者在网站或印刷品上的“正文”需要仔细选择。下面这些对正文的选择也同样适用用GUI上的标签字体、文本和其他控件的标题部分。

- 在计算机显示器上，无衬线（Sans-Serif）字体往往在小字体上表现更好；而印刷品上，正文采用带衬线的字体更有可读性。显示器的像素不足以表现这些细小的衬线（不过，其中一些衬线字体看起来仍然不错，例如Georgia）。一些显示技术例如反锯齿效果对易读性影响很大，因此需要在应用的目标平台上测试这些候选字体的效果。
- 避免使用斜体、草书或其他装饰性的字体，它们在小字体的情况下可读性很差。
- 高度几何化的字体，它们的圆形字母（e, c, d, o, 等等），在小字体的情况下也很难辨认，难以区分。Futura、Univers和其他一些20世纪中期的字体就属于这个类别。
- 所有的Cap字体不适合用作正文，它们很难阅读。不过，如果经过仔细考虑，用在标题和短文本上是可行的。大写的字母看起来都差不多，读者很难区分它们。
- 可能的情况下，把大量文本放在中等宽度的区域里——例如，平均约10-12个英文单词那么宽。不要让很窄的文本区域右对齐，让它们的右边自由分布。

现在来看内在的、情感上的因素。字体有着不同的声音，或者颜色。它们在页面上有不同的图形特性和特质。例如，一些字体更密集，颜色更深；而另一些字体更开放。从笔画的宽度和字母的相对大小可以找到一些线索。一些字体比别的字体更窄，一些字体族有“变窄”的版本让它们更紧凑。文本之间的分隔可能很近，也可能比较宽，让整个文字看起来更开阔或更紧密。

衬线和曲线为字体的颜色或特质增加了另外一种特点。衬线为字母本身增加了很小的比例。这样也让字体变得更精细——对比之下，宽厚一些的无衬线字体看起来比较生硬、强烈，或者更粗糙（特别是helvetica字体）。在每种字体格式中使用的曲线和角度，包括那些衬线的部分，组合形成整个文字。对比一下以前流行的字体（例如Goudy）和别的经典衬线字体（例如Didot）：它们在页面上看起来区别很大。

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Didot

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Georgia

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Goudy Old Style

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Palatino Italic

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Futura

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Verdana

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Arial Narrow

Latin ipsum dolor sit amet, consectetur adipiscing elit. Sed a sem. Nullam nonummy libero id libero. Donec libero erat, consequit in, tincidunt at, malesuada id, urna. Fusce tincidunt consectetur ante. Nam sit amet lorem. Nulla nec ante ac risus tincidunt suscipit. Aliquam luctus. Vivamus lobortis odio at risus porttitor ultrices. Maecenas odio libero, rhoncus et, dignissim id, rhoncus et, quoniam. Vivamus dolor. Quisque feugiat fringilla enim.

Comic Sans MS

图9-9 八种字体，展示在Mac OS X上。注意不同的大小、笔画粗细、文字结构和正式程度。

虽然要解释其中的原因并不容易，但一些字体显得更正式，而另一些字体看起来更随意。有趣的 Sans字体和其他一些好玩的字体当然更随意，但是Georgia字体和Didot字体比起来，也没那么正式。所有大写的字母和单词比小写单词看起来更正式；斜体看起来比较随意。在之前看到的CSS禅意花园中，第8个设计使用了全部大写的Sans-Serif字体，它们看起来比较冷，有种隔离感。同时，第5个设计（它用的是Georgia字体）就更温暖，更加随意。

在这里，文化因素也会有所影响。以前流行的那些字体通常有衬线，看起来就很过时了，尽管 Futura（一种Sans-Serif字体）看起来依然像是来自1963年的科学课本一样。Verdana字体在Web上用得很广，现在已经成为这种媒体上的标准字体。不管在什么样的上下文里面，Chicago都是默认的 Mac字体。

空间和拥挤

SPACIOUSNESS AND CROWDING*

CSS禅意花园的一些设计使用了大量的空白，而另一些就把页面元素挤在一起。和其他设计因素一起，页面上的空间会给人一种透气、开放、从容、平静、自由的感觉，或者正式、严肃的感觉。

拥挤的设计在某些环境下会引起紧张和压力。为什么呢？因为文字和其他图形元素需要“呼吸”——当它们彼此挤在一起，或者和各种边缘、页面边框挤在一起的时候，就会导致视觉压力。我们的眼睛需要在元素周围看到空白。CSS禅意花园的第2个设计（黑色背景上有白色的箭头）会让我们觉得有点不舒服，因为它把标题和文字挤在一起了。和它类似，第6个设计的紧凑布局也让页面显得拥挤，带着工业化的味道，虽然它并没有像第2个设计那样把标题和文字挤在一起。

不过，也不是所有拥挤的设计都会引发那种紧张。也有一些会让人觉得很友好很舒服。如果你给文字和其他元素的空间刚刚好，并把行间距减少到最低的可读程度（读起来仍然要觉得舒服），那么你可能会达到一种比较友好而又不那么浪费的效果。有雏菊图案的第5个设计，就是这样的代表。

曲线和角度

ANGLES AND CURVES

和带有对角线及非矩形图形的页面相比，一个由直线和直角组成的页面看起来往往更冷静，更没有生气。同样，一个有着不同角度的页面也比一个总是重复使用一个角度的页面更有动感——第7个设计就是一个生动的例子。第6个设计使用角度来建立一种不安，并引起视觉兴趣。

曲线也会增加动感和活力，不过也不尽然。一份有着很多圆形和圆弧的设计也可以很平静很舒服。但是，一条贯穿整个页面的曲线会让设计更动感；而在矩形设计中，一些精心挑选的曲线会让它显

得更老练，更好玩。第8个设计使用了一段大的椭圆弧线来达到一种戏剧性的效果——它和其他直线型的设计元素对比很强烈，因此它有很大的影响。

不管两条曲线在哪里交叉，要注意这些曲线相切的情况。相切的角度合适吗？它们会产生更平静更固定的组合；如果角度更尖锐，设计就会更紧张，更不稳定。（要注意的是，这不是什么必须遵守的信条，不过总的来说的确如此。）

当使用角度、曲线和非矩形的元素时，考虑一下视觉焦点的位置：例如尖锐的角度，也就是那些线条交叉的地方，以及多个线条汇聚的地方。使用这些焦点把读者的眼光引导到你希望的地方去。

底纹和旋律

TEXTURE AND RHYTHM

底纹会让视觉设计更加丰富。在“排版”一节中已经提到过，文字会形成它们自己的底纹（纹理）³，你也可以通过精心选择好的字体来控制这种“纹理”的外观。对于很多页面和界面来说，字体就是最重要的底纹。

但是其他种类的底纹也值得引起我们的注意。在空白区域，例如网页四周的空白地带，如果填充上底纹，将会更好看。你也可以在强烈的视觉元素周围添加底纹，成为衬托，就向第6个设计和第7个设计那样。底纹增加了视觉兴趣，而且根据其外观，它们会让界面变得更温暖、丰富、兴奋或紧张。

界面设计中效果最好的底纹是非常细小的，而不是明亮的西洋跳棋似的形式。它们使用温和的颜色，细致的纹理。当分布在大片范围上的时候，它们带来的影响比你想像中的更大。图9-10就显示了一些CSS设计中的底纹。一个像素的点、平行线、精细描画的格子都是很好的几何底纹。它们很容易产生也很容易呈现，而且会让设计变得更为细腻。参见发丝（**Hairlines**）模式。

在计算机屏幕上的文字背后使用底纹要小心——它们往往不可用。因为精致的底纹会干扰小字体文字的可读性。你可以把底纹放到大一些的文字背后，但是要注意字体边缘和底纹之间的相互作用，因为那样会让文字变形。可以这么做：在靠近文字的地方让底纹淡化成一种单一的颜色。

3. 在一份有趣的词源笔记里，英文单词“text”，“texture”和“textile”都来自同样的拉丁语词根。“texere”的意思是“编织”。这会让你产生联想。



图9-10 四种CSS设计中的底纹细节

图片 IMAGES

这里重新设计的每个CSS禅意花园作品都使用了图片。一些用的是照片，另一些用的是半抽象的图片。在所有的设计中，图片的存在是为了给设计设定一种感觉。这些特别的设计可以尽量去设定那种感觉，因为在CSS禅意花园里，设计比内容更重要。

你的情况可能不同，在大多数应用和Web应用里，内容和容易使用比设计风格更重要。你应该有节制地使用纯装饰性的图片，在功能性的GUI设计里谨慎使用图片，因为它们会分散人的注意力。

也就是说，你应该看看设计里的功能性图片和图片——例如工具条图标和图示选择（**Illustrated Choices**）模式（第7章）——看它们是不是表达了你希望整个设计想要表达的那种感觉。同样，对颜色、Texture、角、曲线、空白等元素用同样的标准进行检查。特别是颜色方案、角度和曲线，应该在一组图标中保持一致。不过也不要让它们看起来太相似了。那样的话用户难以看到它们之间的区别。大一些的图标常常比小的图标感觉更好，部分原因是因为你可以把它们画得更大。以及之前说过的拥挤和空间的缘故。

回到装饰性的图片。照片可以唤起强烈的情感反应。你见过的多少网页使用了开心的笑容：一群小朋友在放风筝；身着正装，看起来充满竞争力的商人；道路延伸到美丽的山峰，日落或者海滩；晴朗的蓝天下起伏的草坡？（你可以在某个别的东西而不是网页上找到这样的照片。）

这些照片会唤起我们最深的人类本能，它们也都倾向于让看到的人产生积极的反应——只要是在合适的上下文里。如果你想在一个朴素的小应用程序里放上这些华丽的照片，用户可能会发笑，或者批评它过头了。要小心使用图片，如果你不是很肯定，那就做一些用户测试。

文化因素

CULTURAL REFERENCES

一份设计可能会让你想起某种文化上的东西——某个品牌、电影、艺术风格、历史时代、文艺流派，或者一个隐藏的笑话。熟悉的含义可能会引发强烈的记忆或情感，胜过其他所有的设计因素，尽管最好的设计会让文化和其他设计因素保持一致。

第7个设计可能会让你回忆起20世纪50年代的电影海报。这当然是有意的。感觉这个页面很生动好玩——注意那些角度、颜色和排版方式。很多美国成年人的情感反应可能是“有点傻”、“怀旧”、“复古”。类似的感觉。这份设计里的所有元素一起产生了这种具体的直觉反应。还有一些别的CSS设计重现了Bauhaus建筑学派的风格、近代艺术风格、达达主义风格、搞笑书籍风格，甚至苏联时期的共产主义宣传海报风格。

显然，如果你表达明显的文化含义，考虑一下你要面对的受众。10岁的小朋友领会不到20世纪50年代那种电影海报的感觉。印度的年轻人也领会不到。但是如果你的受众很明确，你知道某种文化含义对他们来说很熟悉，那么这种文化含义可能是一个很好的引子，把对方深深地吸引到你的设计中来。

在功能性的应用设计中很少会使用文化含义，但是你也会在一些应用的皮肤中看到。例如Winamp。它有一个用户制作的皮肤库，包括一个 Klingon 风格的皮肤，一个模仿 iPod 风格的皮肤，还有一个看起来像 Etch-a-Sketch 风格的皮肤。另外，在用户自己制作的 GNOME 背景和主题中，可以找到更多文化含义的参考。

你也可能在一些应用中找到文化含义，例如 Quickbooks，在这个应用中，一些页面看起来像支票和账单。它们实际上已经超越了风格设计，成为一种交互隐喻，不过这个隐喻仍然是文化上的一如果某个人从来没见过支票簿，就不会和见过的人产生同样的反应。

重复的视觉主题

REPEATED VISUAL MOTIFS

好的设计具有完整性：它会形成一个整体。每个单独的元素会在结构上内地对其他元素提供支持。要做到那样很难。我没法给你一个快速公式化的规则来做到它，它需要技巧，需要实践。

但是，对视觉完整性起很大作用的一个因素是视觉元素或视觉主题的不断重复。我们已经谈到过角度和曲线：你可以使用同样角度的斜线，或者类似弧度的曲线，作为设计里的重复元素。**角落处理（Corner Treatments）**模式讲述的就是这样的处理方式。

还有，也要考虑一下排版方式。只采用一种正文字体，虽然其他字体可以很好地应用在一些小区域里，例如侧边栏和导航链接（它们和正文字体相比更突出）。如果你有几个标题或带标题的栏目，那么对它们使用同样的字体。你也可以把小的图形元素（例如线条宽度和颜色），从你的字体中拉出来放到其余的设计中，参见**边界回应字体（Borders that Echo Fonts）**模式。

当类似的文本或控件沿着一条直线重复出现的时候，就产生了一个视觉旋律。在第3个、第4个、第8个设计中的“Select a Design”栏目中，这一点尤为清楚。它们在一个定义良好的分组中显示了每个设计的名字和作者，然后在那个竖栏里继续重复这个分组。你可以很容易地在表单字段、工具栏按钮和其他UI元素上达到同样的效果。

这样的视觉旋律可以成为非常有力的设计工具。小心使用它们，并把它们应用在一组可以比较的元素上——用户会假定外观上的相似意味着功能上的相似。顺便要说的是，如果设计得好，视觉旋律就是为什么大量小对象模式（**Small Multiples**，第6章）看起来那么吸引人的原因。

这对桌面应用来说意味着什么

WHAT THIS MEANS FOR DESKTOP APPLICATIONS

如果在为Web设计，你可能已经非常熟悉前面讲到的内容。人们期望网站——并且延伸到Web应用——在图形上很美观，而且你很少发现它们看起来非常平淡没有特色。

但是，如果你是在设计桌面应用怎么办？如果你试图把这些原则只应用到控件的外观和感觉上——也就是说，控件怎样描绘——那你的选择不多。Java应用可以从一些外观和感觉选项中进行选择，但它们中的绝大多数都是土土的，相当朴素（就像Swing的金属外观和感觉）。Linux应用也有一些不错的选择，例如GNOME的应用主题。但是纯粹的Windows或Mac应用总体上使用标准的平台外观和感觉风格。除非你非常努力想要自己另外开发一份。

在这样的情形下，如果只使用平台风格的外观和感觉标准，我们会原谅你。你可以把你的图形设计注意力集中到别的地方。

但是，和以前相比，一些应用现在看起来越来越“Web化”或“经过精心设计”了，而且它们总的来说看起来更漂亮了。微软的Money 2000就是第一个打破常规的主流应用。它的设计师选择在顶部边距使用背景图片，应用了渐变填充、平滑的标题字体，以及一套不寻常的颜色方案。其他应用也进行了类似的努力。Winamp，这是另一个例子，它已经在很长时间以来使用了富有创造性的皮肤设计；现在很多其他媒体播放器也看起来非常有个性。

甚至就算你确实在控件上使用了朴素的外观和感觉，也还有其他一些方式变得更有创意。

背景

在大的背景区域应用平缓的图片、渐变填充和细致的底纹或者重复的模式，可以让一个界面变得眼前一亮。可以在对话框或页面背景上使用它们，在树、表格或列表的背景上使用，也可以在方框的背景上应用（和它的边框一起）。参见深色背景（Deep Background）模式。

颜色和字体

也常常可以在本地外观的UI中控制颜色方案和字体选择。例如，你可能会在标题处使用一种特别的字体，比标准的对话框字体大上几个pt，也可能再加上一块彩色的背景。如果是设计带标题的栏目（Title Sections）的页面外观，可以考虑使用这种方法。

边界

边界常常是另一个体现创意的地方。同样，如果你使用带标题的栏目（Title Sections）或其他任何一种物理分组，你也许可以改变这些边界的样式。宽度不大的固定颜色方框就很好，下凹的框看起来比较过时。参见角落处理（Corner Treatments）和边界回应字体（Borders that Echo Fonts）这两个模式。

图片

在某些UI工具箱里，有一些控件能让你改变它们的标准外观，可以换个给它们加上背景颜色。例如，按钮就通常可以这么做，所以你的按钮，包括它们的边界可以看起来像任何你想要的效果。表格、树和列表有时候可以允许你决定它们的元素描画方式（在Java Swing里，你完全可以通过这些元素的渲染方式，其他几个工具箱则至少可以让你使用自定义的图标）。你也可以在UI布局上放一些静态图片，这样就可以在任何地方放置任何尺寸的图片了。

这里最大的危险是可访问性。Windows那样的操作系统允许用户改变桌面的颜色/字体主题，这么做并不完全是为了好玩——视力受损的用户可以使用那些带有高对比的颜色方案和大字体的主题，这样他们会看得更清楚。确定你的设计可以适应这些高对比的主题。有必要这么做。⁴

同样，你也可能会让那些常规的文本标签变得更丰富，采用图片，加上特别的字体，可能带有光晕或阴影效果，以及复杂的背景。这么做在Web页面上很常见。如果你坚持用图片代替文本，那么需要为这个图片提供足够的信息来让屏幕阅读器（例如JAWS）能读到它们（具体怎么做取决于你正在使用的具体UI技术）。

另一个潜在的危险是会让你的用户变得疲惫。如果你在设计一个全屏幕或长时间使用的应用，那就调整饱和的颜色、字体大小、强烈的对比，以及那些抢眼的底纹——让设计更安静，而不是更喧

4. 还有，取决于谁来购买你的软件，它也可能觉得和法律有关。例如，美国政府要求所有联邦机构使用的软件都要保证那些残障人士可以使用，参见<http://www.section508.gov>了解更多内容。

闹。更重要的是，如果你的应用将会使用在高压力的环境下，例如重机器的控制面板，那么去掉任何会分散用户注意力的多余元素。在这种情况下，认知因素远远要比审美重要。

也就是说，我希望这些讨论能为你提供一些方法，让你的桌面UI变得更有趣一些。图9-11就是几个控件的例子，它们有着自定义的外观和感觉，但同时仍然保持相对可用。



图9-11 一些非标准（但可用）的控件

模式

THE PATTERNS

这些模式（除了皮肤模式）应用了前面介绍部分讲到的那些概念。它们讲述了如何应用这些概念的具体方法。例如，角落处理模式就捕捉了一种重复视觉主题的方法，边界回应字体模式则反映了另一种做法。深色背景模式讲的是底纹选择，发丝模式也是。还有，在对照字体大小模式中讨论了字体问题。

-  深色背景
Deep Background
-  少一点色彩，多一些价值
Few Hues, Many Values
-  角落处理
Corner Treatments
-  边界回应字体
Borders that Echo Fonts
-  发丝
Hairlines
-  粗细字体对比
Contrasting Font Weights

皮肤模式有些不同。它讨论得更多的是元设计 (meta-design) —— 没有说到如何设计皮肤，或者如何设计应用的外观和感觉。它说的是如何设计你的应用来让其他人用他们自己的设计来取代你提供的外观。

-  皮肤
Skins

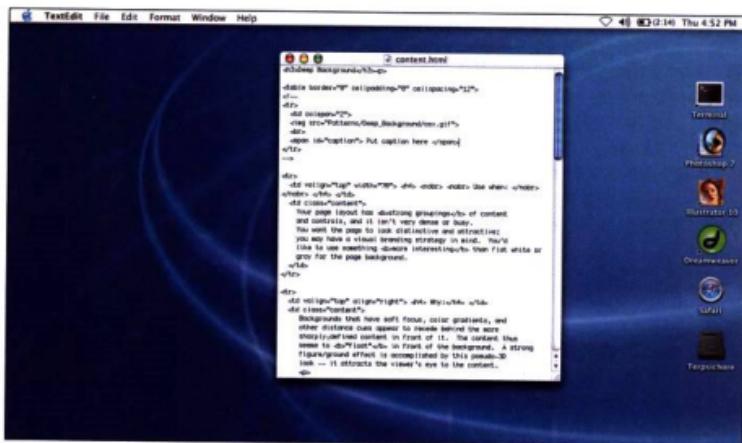


图9-12 Mac OS X的桌面

它是什么

用图片或渐变颜色作为页面的背景，这样能可视化地拉开和前景元素的距离。

什么时候使用

你的页面布局有着显眼的可视化元素（例如文本块、一组控件或一些窗口），而且它并不是很密集很拥挤。你希望页面看起来与众不同，富有吸引力；可能在意识里有某种视觉品牌策略。你想使用某种更有趣的东西而不是白色或灰色的页面背景。

为什么使用

柔和的、颜色渐变的，以及其他一些距离暗示的背景看起来会远离前景中的元素。这样，内容就似乎“漂浮”

在背景之上。这种拟3D的外观产生了一种强烈的人物/地面效果——它把观看者的眼球吸引到内容上。

除了好玩之外，它就是看起来也很不错。

如何使用

使用的背景要有一个或多个特性，列举如下。

柔和的焦点

让线条变得模糊，并避免太多的细节——明显的线条会干扰它上面内容的可读性，如果内容是文字或小图标，这种干扰就特别严重（如果对比不强烈，可以不用去除这样的线条，但是就算那样，它上面的文字看起来可读性也不好）。

颜色渐变

明亮饱和的颜色是可以的，不过，再次要说明的是，它们之间的坚硬线条不行。让颜色把它们彼此融合起来。实际上，如果没有一幅图片用这样的背景，你可以用自己喜欢的画图工具来创建一份简单的颜色渐变效果——这样也比一种固定的颜色要好。（你不用把这样的渐变效果保存或下载成图片。在Web上，可以通过不断重复一个1像素宽的窄条来达到这样的效果，可以是横向重复，也可以是纵向的重复。）在那些可以编码产生大块颜色的系统里，也很容易能得到渐变效果。

深度暗示

模糊的细节和垂直的颜色渐变是两种线索，可以给我们的视觉系统一种距离感。要理解其中的理由，想像一下一幅关于山的风景画——距离越远，景物的颜色就越柔和，也越朦胧。其他深度

暗示线索包括底纹渐变（越远的东西越小），以及线条的辐射状分布。

没有强烈的焦点

背景不应该和主要内容竞争用户的注意力。分散（微弱）的焦点也可以起到作用，但是要确定它们有助于自己成为整个页面的平衡元素，而不是把用户的注意力从他们应该看到的页面内容中引开，如图9-13所示。

当你设计一个具有深背景的界面时，考虑一下当用户改变页面大小的时候会发生什么事。这个背景如何适应一个更大（或者更小）的页面？它会重新缩放来适应大小，还是保持不变。只是一张不会缩放的图片？保持不变可能会让用户觉得更安心，就像很多Web页面所做的一样，它也让人觉得更稳定。另外，你也不用担心改变长宽比例，那种改变会给很多图片带来问题。

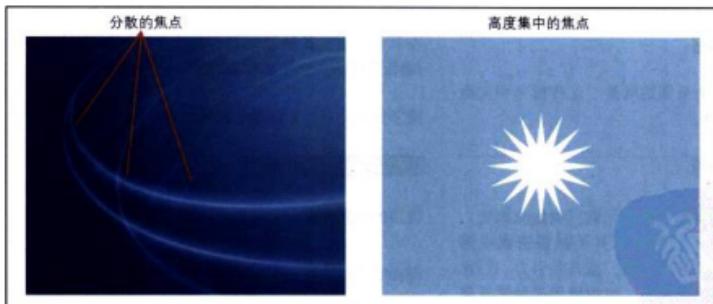


图9-13 模糊 vs. 强烈的焦点

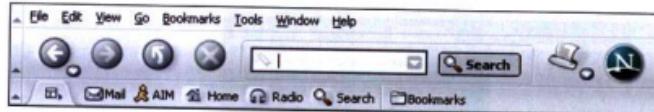


图9-14 Netscape 7的外观显示了UI控件背后基于渐变的雕刻效果使用。渐变效果和其他视觉特性一起配合得很好——几乎看不见阴影，“Home”和“Radio”按钮周围的效果，甚至搜索区域周围的下凹阴影——创建了一种完美的2.5D的半立体效果。

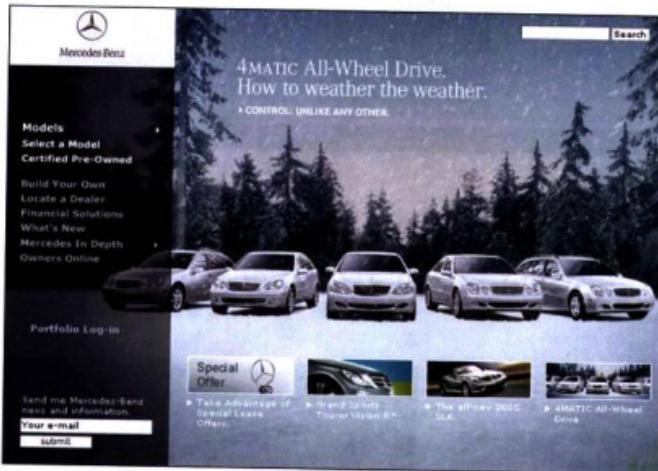


图9-15 Mercedes奔驰的网站使用了图片作为背景。这张图片有着很强的焦点——当然，是汽车——它们是页面的中心元素。但是图片的周围部分，非常柔和，是其他内容的深背景：搜索框、底下的四个小图片，以及“4MATIC All-Wheel Drive”的口号。

本图最有意思的部分在于左边深色的区域，网站需要一个导航条，以及一些小一点的文字，但是不能把这些链接直接放在背景图片上——文字在小细节的背景下也许会不可读。在构成上也会让人迷失。这种半透明的烟色玻璃背景通过增强对比高亮显示了这些白色的链接，也平衡了整个页面（要不然会显得向右倾斜）。它并没有遮盖漂亮的背景图片；而且还增加了一种层次感。参见 <http://www.mbusa.com/brand/index.jsp>。



图9-16 来自<http://thebanmappingproject.org>

它是什么

选择一种、两种、最多三种主要颜色样式应用在页面上。通过选择这很少几种颜色的混合值（不同亮度）来建立一个颜色板。

什么时候使用

你在决定某个应用或网站的颜色方案。想避免一种俗气的、五颜六色的、“愤怒的水果沙拉”(angry fruit salad)效果，但你仍然想要界面有自己的特色。

为什么使用

当考虑颜色时，有时候颜色少一点比较好。太多颜色会使界面变得很散乱。特别是如果它们很饱和的时候，会让设计很嘈杂、很杂乱。颜色会对用户的注意力形成竞争。

但是当你使用一个颜色的多个变种时，可以创建一种具有深度的设计。想想前面例子里使用的灰色和棕

色，重现了下面的色带（见图9-17）。注意，更深的颜色和更灰的颜色看起来是在向后倒退，而更浅更明亮的颜色似乎在往前移动。这就有助于达到一种拟3D的效果（例如下凹、阴影、渐变）形成了UI的高度标志。



图9-17 TheBan Mapping项目UI里使用的两种颜色

之前提到过，选择一种、两种，或者三种主要的颜色。你可以任意使用黑色和白色，但是用灰色要看情况。事实上，灰色在多种颜色值和亮度上效果不错。它是一种通用色，特别是如果你再给它增加一点点颜色的话：更蓝一点（更冷），或者再米色一点（更暖）。

有了这些颜色，改变它们的值可以得到一个亮度和暗度范围。你也可以同时改变饱和度。和只改变颜色数

值相比，这样可以得到更多的细微颜色组合。在这些颜色里，可以尽量多用一些来形成这个应用的调色板。

当然，除了这些颜色，你也可以在界面上使用其他颜色。不过使用的时候一定要谨慎。图标、广告，以及其他占用空间相对不多的元素不必局限于这个有限的颜色方案。你可能只想使用一种或两种着重的颜色，例如红色或青色来标记兴趣点。事实上，使用一种色调作为UI的“背景”会让那些少数颜色更显眼，因为它们不会在一种类似颜色的海洋里迷失。

示例

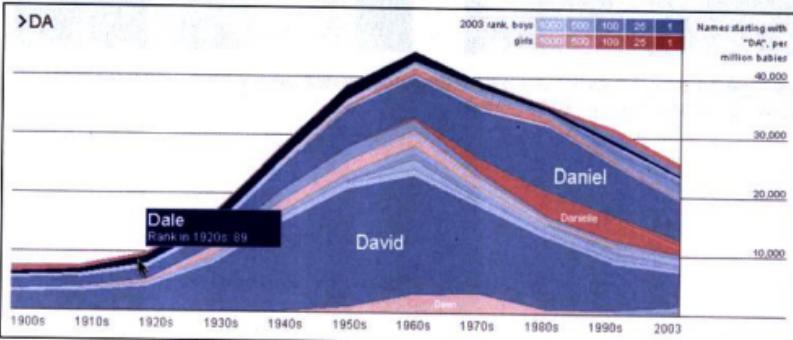


图9-18 这张图片使用了两种颜色，蓝色和粉红色来显示数据。蓝色代表男孩的名字，粉红色代表女孩的名字。在这些颜色中，颜色值代表了这些名字在2003年的受欢迎程度。第三种颜色，深灰色，应用在这些数据周围的边框上——格线、数字、标题——以及一种深蓝色高亮显示了选中的名字（“dale”）。

不管是在认知上还是在审美意识上，这种颜色组合都非常有效。这些颜色和颜色值对于数据来说是有意义的，而且这种编码也很容易理解——一旦看过一次，就基本上不需要图例了。在审美上，整个页面层次丰富，也不会过于俗艳，不像那些充斥了五颜六色的页面。而且，在美国的文化里，人们可以理解浅蓝色和粉红色是“婴儿”的标志色，因此这里也很好地体现了情感和文化上的联系。参见<http://babynamewizard.com>。

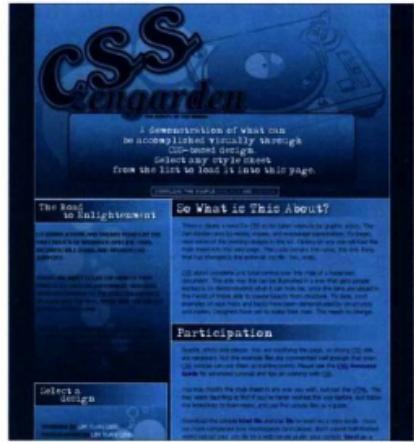


图9-19 甚至一种颜色也能产生这种细微和有趣的效果。这里有两个来自CSS禅意花园的单色例子（绿色的那个有几种绿颜色，一些会更黄一点，但是在用户看起来，它们是一个色调）。

角落处理

Corner Treatments



图9-20 <http://starbucks.com>上的圆角

它是什么

不是使用普通的直角，而是使用斜纹、曲线或镂空来装饰界面上的方框角落。让这些角落处理和整个界面保持一致。

什么时候使用

界面使用直角元素，例如方框、按钮和Tab页。

为什么使用

视觉主题的重复有助于设计的完整性。当你设计好一个角落主题并一致地把它用在很多地方时，它为整个设计带来一种独特的外观。和常见的直角角落相比，它肯定不会那么单调。

如何使用

很多网站使用圆角。其他一些使用斜线，还有少部分

使用镂空效果。你要选择哪一种取决于网站的整体外观——你有一个站点图标、一张图片，或者某种字体作为吸引视线的元素？使用这些可视化元素的其中一个。你想得到某种让人舒缓（曲线经常会达到这样的效果），更尖锐，或者充满活力的效果吗？可以试试几种不同的做法。

不是界面上所有的直角元素都需要角落处理——一个东西就算很好，也不要过度使用。但是分组框或面板通常需要，Tab页也经常这样处理。如果你在一个分组框进行了角落处理，那就对所有的分组框进行处理，这样有助于一致性的形成。

还有，不是给定方框的每个角落都需要进行处理的。有时候可以只处理两个对角，例如左下角和右上角。有时候甚至只需要处理一个角落，通常是左上角或右上角。

每个这类元素重复的地方，都要确定它看起来和其他的地方差不多。换句话说，曲线角落应该使用同样类型的曲线（尽管这个模式开始的时候显示的星巴克网站，很好地利用了它站点图标上的曲线，并把那些

曲线应用在页面上所有的角落——面板边界、图标，甚至“Go”的按钮。整个效果很休闲，也很统一。而不必使用一模一样的曲线；角度应该保持不变——例如，不要把45度的角和20度的角混在一起使用。

示例

本模式一开始的例子是星巴克的网站，它具有圆型的商标，并将该主题概念应用到整个页面的所有矩形角

落——包括面板边框、图示、甚至“go”按钮。整体的效果是让人觉得放松，且统一、一致。

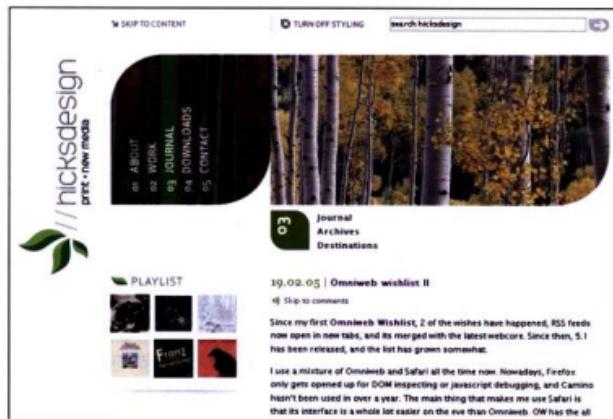


图9-21 这个个人网站使用了圆角叶子形状的角落作为主要的设计元素。不是每个角落都是圆角的，也不是每个曲线都一样，但它仍然是一个重复的主题，设计一样有效。参见 <http://hicksdesign.co.uk>。



图9-22 这个K2的网站有一个完全不同的外观。它使用了莱檬绿（lime-green）的边界和角形的底部角落。注意，大部分的角度出现在右下角，不过在左边，也有一个左下角是这样处理的。也要注意，所有的角度和K2站点图标上的右下角一样。



图9-23 角度也可能看起来很平和。这类有两个Tab上使用的角度例子——一个是在Web上，另一个出现在桌面应用上。对于Tab来说，角度可以起到很大作用——它们让Tab在视觉上有一点重叠，但又不会遮盖这个Tab的整个右边或左边。

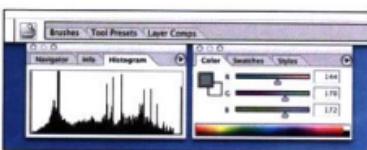


图9-24 Photoshop里的角度Tab

边界回应字体

Borders That Echo Fonts

图9-25 来自`http://www.moma.org/exhibitions/2002/brucke/`**它是什么**

在描画边界和其他线条的时候，使用设计中某种主要字体所使用的颜色、粗细和曲线。

如何使用

首先从你的设计中选择一种字体。小标题或大标题的字体通常很合适，广告标识上的字体也可以，但有时候甚至正文的字体都可以。观察它的特征：颜色、主要线条的粗细、纹理、曲线半径、角度，以及字间距。

什么时候使用

你的设计包含某种精心选择的字体，它有良好的视觉效果，例如用于大标题、小标题，或者广告标识的字体。

为什么使用

可视化主题的重复有助于设计的统一。而字体和边界在设计中处于同样的级别——只有几个像素宽——而且，如果它们在视觉上彼此加强的时候，效果会更加明显。当它们互相冲突的时候（特别是如果你使用多种不同边界的时候），它们的作用会被削弱。

现在试着使用这些特征来描画边界和线条。虽然你可以在粗细上做一些手脚，让它比字体的笔画粗一点儿，但是颜色应该和字体保持一致，如果该字体有很明显的圆形曲线，例如很多现代的Sans-Serif字体那样，那么在边界的角上试试同样弧度的曲线。

如果这是一种特别有意思的文字，那么问问自己它到底为什么这么有意思。看看是否能把那些视觉元素从字体中提取出来，放到其他地方的设计中去。

当然，你也不必对界面上所有的边界都这么处理，只要处理其中一些就可以了。特别是如果线条比较粗的时候。小心，不要让边界变得太宽或者太粗糙。宽边界会很明显，而且超过某种程度的话，就会盖过它们

所包含的内容。例如，图片的边界就可以比正文字体笔画再宽一点点。你可以把一个像素粗的线条和再宽一些的边界有效地结合起来使用。

示例

上面这个模式开始时的那个例子（图9-25）就使用了回音广告标识（“Artist of Brücke”）字体的边界。这个广告标识看起来是手写的，有一些毛毛的边。而UI的其他字体很光滑也很小：如果该设计中不在别的地方多次重现

这种粗的手写效果的话，它就不会成为一个这么强烈的设计元素。但是既然有了边界与它呼应，整个页面在视觉上整体一致。和那些光滑的字体和线条相比，这样形成了很好的对照。



图9-26 你可以在很多网站上看到这个模式。流行的Sans-Serif字体和宽宽的实心彩色边界遥相呼应。在这里，“Earnings”的大标题选取了HP站点标识字体上的元素——白色，四到五个像素宽——然后，五个像素宽的白色边界又再次使用了这种元素。这里还有一些需要放大镜才能看得到的地方：“Earnings”字体中的竖线实际上只有三个像素宽，但是这个字体看起来仍然和五个像素的边界十分相似。感觉比精确度更重要。参见<http://hp.com>。

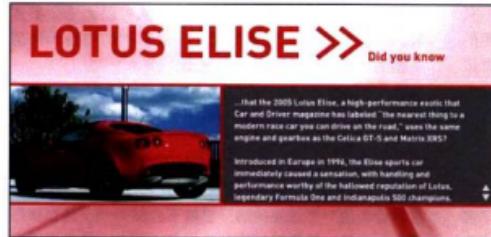


图9-27 这个例子有点类似，但是，字体和边界的宽度并不完全一致。
不过这样是可以的。参见`http://www.planetkaizen.com`。



图9-28 Dakine的网站把它们混在一起了。它使用了很多不同的设计元素，但是不同角度的白色线条编实对站点标识字体形成了呼应。总的来说，它们给页面带来了动感、张力和尖锐的感觉，那当然毫无疑问是设计师想要追求的效果——Dakine向年轻人提供运动器材。参见`http://dakine.com`。

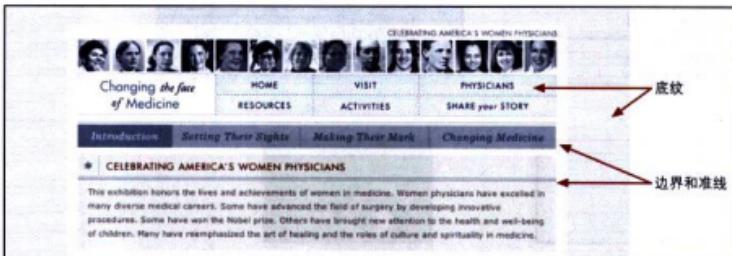


图9-29 来自http://nih.gov

它是什么

在**边界**、**水平标尺**和**Texture**上使用的一个像素宽的线条。

什么时候使用

你希望界面看起来很优雅、很精致。

为什么使用

当你的设计里有一些大元素，例如大块正文、一个不同颜色的区域、图片的时候，发丝为设计增加了一种现在没有的精度。这样，设计看起来会更精致。

如何使用

下面是一些可以在界面上使用发丝模式的方式：

- 通过为标题加下划线来区分带标题的栏目 (**Titled Sections**)
- 用水平分隔线、垂直分隔线或封闭的边界来划分不同的内容区域

- 引导视线穿过一个组合元素的指示线
- 在不同背景颜色的区域之间，清楚标识它们的边界
- 用在底纹上，例如格线或一组水平线
- 用在图标、图片和画出的图形里
- 用在控件周围的边界上，例如按钮

如果放在非常细的Sans-Serif字体旁边，发丝模式的效果会特别好。记住，一根灰色的线条比黑色的线条看起来更细。哪怕它们同样都是一个像素宽。同样，其他的浅色也是这样。就像这个模式一开始NIH里头使用的Teal那样。线条和背景之间的对比越不强烈，它就越显得越细，越淡。

另一种可以让发丝显得更淡的方式——并且可以为它增加另一种纹理——是让它成为虚线而不是视线。正在写这本书的时候，精细描画的虚线在Web上用得越来越多了，甚至在链接下划线上也是这样。

有一个在设计中增加张力和尖锐感的小技巧，那就是把发丝模式推到一行文本的底部。CSS禅意花园的第一个设计对它的标题和大标题就是这么处理的（见图9-8，在本章的介绍部分）。

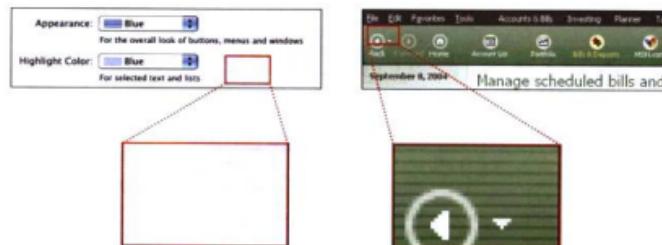
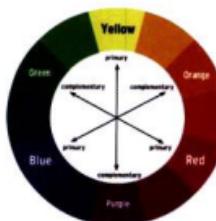


图9-30 微软和苹果公司都在它们的UI设计里使用了精细的发丝元素。这就是其中的两个例子——一个来自OS X的标准对话框，另一个来自微软的Money软件。两个设计都把小字体放在有底纹的背景上，这样有一点点冒险。但是这些底纹的颜色非常接近，其中的区别几乎感觉不到，因此这些发丝不会影响可读性。



图9-31 IDEO的特别ID卡Web特性使用了发丝作为主要的设计元素。和这个模式开始时的NIH例子一样。注意，发线划分了标题、主要的导航链接（“Digital Identity”）、次要链接（那些方形的图标），以及那些装饰性的方块。该设计在主要链接背后使用了水平的发丝底纹。这些发丝很好地协调了细细的Sans-Serif字体。参见<http://www.ideo.com/identity/>。

Color, Mood & Meaning



SEEING GREEN

Red is a primary color, along with yellow and blue.

Locate the positions of each on the color wheel.

Now look at the color opposite the primary. It is called the complement. Green is opposite red. Green is red's complement. You can start at red and then create its complement, green, on your own.

Purple is yellow's complement, and orange is blue's complement...therefore, if you started at a blue circle, you would end up at orange. If you started at yellow, the alternative would be purple. Note: A primary color's complement is the mix of the other two primaries.

Let's look at how colors react when surrounded by other colors.

Color, Contrast & Dimension in News Design

< >

REFRESH BY SECTION

图9-32 这个设计使用了发线来框定内容，并在标题上建立了强烈的焦点（“Color, Mood & Meaning”）。这次，很多细线是曲线而不是直线，但是应用了同样的原则。参见<http://poynterextra.org/cp/index.html>。

粗细字体对比

Contrasting Font Weights



图9-33 来自<http://eg2006.com>

它是什么

使用两种字体互相对照——一种细一些，颜色浅一些；另一种粗一点，颜色深一点——来区别不同级别的信息，并丰富视觉效果。

什么时候使用

文字是页面上非常重要的元素，而且希望页面的结构在第一眼看上去就很清楚。你希望页面看起来很生动。

为什么使用

当两种字体粗细不一的时候，它们就形成了强烈的视觉对比。在美学上，对比会让对象显得生动有趣，而且吸引眼球的排版对比，包括大小、底纹、颜色——不过尤其是粗细——能保证你的页面不会显得单调。

你可以使用这种对比来组织页面上的文本结构。例如，更深颜色的文字可以用在标题和大标题上，这样可以建立视觉层次结构。数字上方的粗体字会把视线引向它。因此，对比字体粗细在达到审美效果的同时

也有助于认知上的理解。

如何使用

这个模式有很多可能的应用方式。本书已经提到可以在大标题上应用粗体，但是还有以下一些应用方式：

- 用来区分双栏列表模式的数据标签
- 区分信息文字和导航链接
- 标识选中状态，例如选中的链接或列表元素
- 加重一个短语中的文字
- 在广告标识中区别其中的一些文字

如果你使用的字体比正文字体要大，例如下一页上的广告标识，要确保这种对比足够强，能被人觉察到。当某种字体使用了多种粗细时，例如Helvetica Neue字体那样，那么，选择那种字体之外的字体。如果对比不强烈，它看起来不是精心设计的（其他字体属性的区别也是这样）。如果你让两个文本元素有着不同的大小，那么它们看起来确实很不一样；如果你想把多个字体族混合起来使用，请确定它们看起来不要太相像！）



图9-34 不是一种，也不是两种，Adaptive Path的首页使用了三种字体粗细对照。首先是广告标识——最粗的字体，最深的颜色，着重突出短单词“Path”。它下面的图形则增加额外的重点。（注意，在本屏幕截图里还体现了深色背景和发线这两个模式）。

第二个地方是标语，这里用粗体重点突出了“Value”和“Experience”。这里，还使用了空间来区别这些文字，因此字体粗细对照的价值在于着重强调其中某些单词。

第三个地方在导航条上，当前页面（在本屏幕截图上，是“Home”）文字的字体比其他字体更粗。你应该把所有的链接当做一组对象，但是应该更突出当前页面？因此，它所使用的字体和其他链接字体一样，只有粗细不同。



图9-35 当你面对采用正文字体显示的数据时，可以用粗体来把标签和数据区分开来。在这个小窗口里，用户可能会对数据更感兴趣，因此它们得到了吸引眼球的粗字体。参见<http://lukew.com/portfolio/>。



图9-36 几种更随意的强调和区分的例子：Getty图片Logo，一个来自<http://microsoft.com>的标语，以及CHI 2003的Logo。

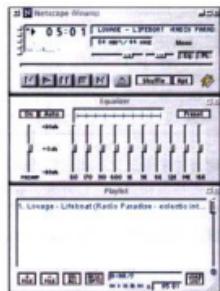


图9-37 三种Winamp皮肤

它是什么

为你的应用提供开放的外观和感觉架构，让用户可以设计他们自己的图形和风格。

什么时候使用

你的目标用户包括大量了解你应用界面的人群。对于这些人而言，界面在认知层次上要求不高——例如，它并不是用在高度压力的环境下——因此没有必要让所有的控件都看起来很标准。

还有，这些用户喜欢自己动手。他们看重风格，也喜欢对软件参数进行设置，以符合他们自己的品位。

为什么使用

当人们重新排序和自定义他们的个人空间时，不管是物理上的空间还是虚拟的空间，他们都产生了一种对该空间的拥有感。这是一种基本的人类需要（虽然并非所有的人都有这样的：很多人对软件的“出厂设置”已经心满意足了）。简单地改变颜色和字体参数是一种常见的自定义个人软件环境的方式，但皮肤模式走得

更远，它涉及颜色方案和字体。

Internet上有很多证据表明，用户非常喜欢皮肤。实际上，我们在讨论两组用户：那些下载和使用皮肤的用户，以及那些不但使用皮肤，而且喜欢设计皮肤的用户。那些设计皮肤的用户把它看做一种体现创意和将其作品发布给公众的机会。他们之中，很多是图形设计师。这些人可能会对你的UI设计非常非常了解。

不管怎么样，现在有数不清的应用——很多是开源应用，不过Windows XP也是其中之一——是可以更换皮肤的，而且，用户自己定义的皮肤也非常多。在这些皮肤上花费的人工说明了这种创意驱动的威力。对于皮肤设计师来说，可以更换皮肤的应用满足了另一种基本的人类需要：创造。

如何使用

准确地说，如何设计并实现一个可以更换皮肤的应用完全取决于你使用的UI技术。因此，这里很难再归纳出其他东西了。

首先，记住，任何Windows XP本地应用已经可以通过一个叫做WindowBlinds的产品（参见<http://www.stardock.com/products/windowblinds>）来对它们进行皮肤更换。本质上，这个产品可以改变本地控件的绘画方式。

其次，这里讲的是应用级的皮肤更换如何工作。部分可以更换皮肤的应用让皮肤设计者定义一些简单的位图碎片或“切片（Slices）”。它们可以放在应用框架具体的位置上。因此，一名皮肤设计师也许可以为角落标题、水平切片（它们在必要的时候沿着水平线不断重复）、垂直切片，以及按钮的不同状态设计图片：例如，选中状态、按下状态和鼠标滑过的时候。ICQ的皮肤类似于这种情况。参见图9-38。你的原始页面布局应该在这些情形下保持不变。

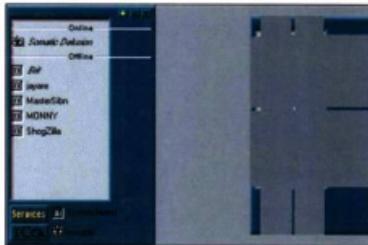


图9-38 静态位图皮肤实现，它的指南在 <http://www.geocities.com/bakerstr33t/>。

不过，有一些应用，包括Winamp 3，允许皮肤重新排

列控件的位置，重新定义应用的外观，甚至增加新的功能。有时候包括一些脚本，而这些皮肤也相对来说更难制作。而且它们会重新排列UI！你选择的功能和导航可以保留下，但其他的就很困难了。

皮肤引起的一个问题在于，它们让界面更难使用。对于很多设计不好的皮肤来说，确实是那样。不过，问问你自己：这有多重要？每个应用要做到认知上的完美吗？（默认的外观和感觉并不完美，但它们当然也比皮肤经受了更多的可用性测试。）如果对于某个应用，用户已经非常熟悉，没有很高的认知要求，那么有一种观点就是，它的基本可用性已经“足够好了”。现在个人偏好更重要。当有皮肤可用的时候，人们自己选择那些皮肤，不管他们是否在可用性方面了解了多少。

在某种程度上，你可以也应该。作为设计师职责的一部分，决定哪种更换皮肤的级别。你可能只允许更换图标。你可能允许位图级别的皮肤更换，那样会保持布局不会改变，例如ICQ那样。你也可以允许完全的自定义皮肤设计，像Winamp 3那样：完全取决于你自己，如果那样的自由会让界面非常难用。

我猜测，优秀的应用设计——例如，精心选择的功能，容易理解的组织模型，适当的导航，良好的页面布局，以及标准的控件——可以让一个界面更能承受糟糕的皮肤。尽量把界面设计好，然后把它拿出来，让人们在你觉得合适的级别上进行自定义设置。结果会怎么样呢？

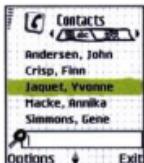


图9-39 很多高端手机会安装一些皮肤，还有铃声和其他一些自定义方式。这个图片显示了Nokia 6600的一些皮肤。要注意，所有的文字和按钮还在原来的位置；只有细节的地方，例如墙纸、图标和特定的控件图形是可以更换皮肤的。参见<http://mangothemes.com>。

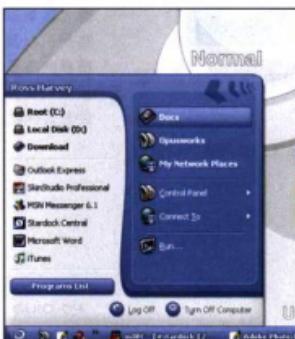


图9-40 四个WinXP皮肤的开始菜单，来自<http://browse.deviantart.com/skins/windows/windowblindsxp/>。

前面提到过，WindowBlind会让整个Windows XP的桌面，以及本地应用的皮肤可以自定义设置。这些屏幕截图显示了一些来自Windows XP的精致皮肤。注意，它们使用了本章讨论过的很多技术和模式：一到两种色调带来更多价值的模式，深色背景模式和拟3D的细节处理，动态的曲线和角度，以及有趣的角落处理。

施文泰
893433333

参考文献
REFERENCES



如果想要进一步学习本书提到的概念，可以从下面这些参考书目开始。当然，这并不是一个完备的清单，只是列出了一小部分非常有名的书籍和网站，但是还有很多很好的资源没有列在这里（极有可能是我不知道它们）。你也可能会发现同一本书列在几个不同的地方——毕竟，很多优秀的设计书籍涵盖了多个主题。

基本设计参考

- Krug, Steve. *Don't Make Me Think! A Common Sense Approach to Web Usability*, Second Edition. Berkeley, CA: New Riders, 2000.
- Cooper, Alan, Robert Riemann. *About face 2.0: The Essentials of Interaction Design*. Indianapolis: Wiley Publishing, Inc., 2003.
- Norman, Donald A. *The Design of Everyday Things*. New York: Basic Books, 1998.
- Shneiderman, Ben. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*, Fourth Edition. Reading, MA: Addison-Wesley, 2004.
- Williams, Robin. *The non-Designer's Design Book: Design and Typographic Principles for the Visual Novice*. Berkeley: Peachpit Press, 2004.

第1章：用户做些什么

- Krug, Steve. *Don't Make Me Think! A Common Sense Approach to Web Usability*, Second Edition. Berkeley, CA: New Riders, 2000.
- Hackos, Joann T. and Janice C. Redish. *User and Task Analysis for Interface Design*. New York: John Wiley & Sons, Inc., 1998.
- Holtzblatt, Karen, Jessamyn Burns Wendell, Shelley Wood. *Rapid contextual Design: A How-To Guide to Key Techniques for User-centered Design*. San Francisco: Morgan Kaufmann, 2005.
- Carroll, John M. *Making Use: Scenario-Based Design of Human-Computer Interactions*. Cambridge, MA: MIT Press, 2000.
- Cooper, Alan, Robert Riemann. *About face 2.0: The Essentials of Interaction Design*. Indianapolis: Wiley Publishing, Inc., 2003. (The first section, "Know Thy User," is especially relevant.)
- Csikszentmihalyi, Mihali. *Flow: The Psychology of Optimal Experience*. New York: Harper Perennial, 1991.
- Schacter, Daniel L. *The seven sins of Memory: How the Mind Forgets and Remembers*. Boston: Houghton Mifflin Company, 2001. (The chapter on absent-mindedness talks about prospective memory.)

第2章：组织内容

- McCloud, Scott. *Understanding Comics*. New York: HarperCollins, 1994. (Defines "idiom" for the comics and visual arts world.)
- Mccullough. *Digital Ground*. Cambridge, MA: MIT Press, 2004. (Defines "type" for architecture, and explains why the concept is important.)
- Rosenfeld, Louis, Peter Morville. *Information Architecture for the World Wide Web: Designing Large-Scale Web Sites*, Second Edition. Sebastopol, CA: O'Reilly Media, 2002.
- Wodtke, Christina. *Information Architecture: Blueprints for the Web*. Berkeley, CA: New Riders, 2002.
- Jacobson, Robert, ed. *Information Design*. Cambridge, MA: MIT Press, 1999.
- Lindholm, Christian, Turkka Keinonen, Harri Kiljander. *Mobile Usability: How Nokia Changed the Face of the Mobile Phone*. New York: McGraw-Hill, 2003. (Special considerations for the application design of mobile devices.)

第3章：到处走走

- Jacobson, Robert, ed. *Information Design*. Cambridge, MA: MIT Press, 1999. (See especially "sign-Posting Information Design," by romedi Passini.)
- Baxley, Bob. *Making the Web Work: Designing Effective Web Applications*. Indianapolis: Sams Publishing, 2002.
- van Duyne, Douglas, James A. Landay, Jason I. Hong. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Reading, MA: Addison-Wesley, 2002. (For this and all the remaining books listed here, find the chapters that deal with navigation and links.)
- Johnson, Jeff. *Web Bloopers: 60 Common Web Design Mistakes, and How To Avoid Them*. San Francisco: Morgan Kaufmann, 2003.
- Lindholm, Christian, Turkka Keinonen, Harri Kiljander. *Mobile Usability: How Nokia Changed the Face of the Mobile Phone*. New York: McGraw-Hill, 2003.
- cooper, Alan, Robert Reimann. *About Face 2.0: The Essentials of Interaction Design*. Indianapolis: Wiley Publishing, Inc., 2003.
- Nielsen, Jakob. *Designing Web Usability*. Berkeley, CA: New Riders, 2000.
- Krug, Steve. *Don't Make Me Think! A common Sense Approach to Web Usability*, Second Edition. Berkeley, CA: New Riders, 2000. (This book has a good explanation of visual hierarchy.)

第4章：组织页面

- Williams, Robin. *The Non-Designer's Design Book: Design and Typographic Principles for the Visual Novice*. Berkeley: Peachpit Press, 2004.
- Faimon, Peg, and John Weigand. *The Nature of Design: How the Principles of Design Shape Our World: From Graphics and Architecture to Interiors and Products*. Cincinnati: HOW Design Books, 2004.
- Hashimoto, Alan. *Visual Design Fundamentals: A Digital Approach*. Hingham, MA: Charles River Media, 2004.
- Mullet, Kevin, Darrell sano. *Designing Visual Interfaces: Communication Oriented Techniques*. Mountain View, CA: Sun Microsystems, Inc., 1995.
- Solso, Robert. *Cognition and the Visual Arts*. Cambridge, MA: MIT Press, 1994. (The chapters on color, Gestalt principles, and eye-tracking are particularly relevant.)
- Krug, steve. *Don't Make Me Think! A common Sense Approach to Web Usability*, Second Edition. Berkeley, CA: new riders, 2000. (This book has a good explanation of visual hierarchy.)
- Baxley, Bob. *Making the Web Work: Designing Effective Web Applications*. Indianapolis: Sams Publishing, 2002. (Chapter 10 discusses layout.)

第5章：完成任务

- Norman, Donald A. *The Design of Everyday Things*. New York: Basic Books, 1998. (This book discusses affordances, among other things.)
- Johnson, Jeff. *GUI Bloopers*. San Francisco: Morgan Kaufmann, 2000.
- Gamma, Erich, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: elements of reusable Object-Oriented software*. reading, MA: Addison-Wesley, 1995. (Defines the "Command" pattern, often used to implement three of Chapter 5's patterns.)
- cooper, Alan, Robert Reimann. *About face 2.0: The Essentials of Interaction Design*. Indianapolis: Wiley Publishing, Inc., 2003.

第6章：显示复杂数据

- Tufte, Edward. *The Visual Display of Quantitative Data*. Cheshire, CT: Graphics Press, 1983.
- Tufte, Edward. *Envisioning Information*. Cheshire, CT: Graphics Press, 1991.
- Tufte, Edward. *Visual Explanations: Images and Quantities, Evidence and Narrative*. Cheshire, CT: Graphics Press, 1997.
- Fowler, Susan, and Victor Stanwick. *Web Application Design Handbook: Best Practices for Web-Based Software*. San Francisco: Morgan Kaufmann, 2004.
- Cleveland, William S. *Visualizing Data*. Summit, NJ: Hobart Press, 1993.
- Wildbur, Peter, and Michael Burke. *Information Graphics: Innovative Solutions in Contemporary Design*. London: Thames and Hudson Ltd., 1998.
- Spence, Robert. *Information Visualization*. Reading, MA: Addison-Wesley, 2001.
- Monmonier, Mark. *How to Lie With Maps*, Second Edition. Chicago: University of Chicago Press, 1996.
- <http://www.cs.umd.edu/hcil/> The University of Maryland's Human-Computer Interaction Lab has done wonderful work on information graphics, and will no doubt continue to do so.
- <http://www.math.yorku.ca/scs/Gallery> A sampler of the best and worst information graphics out there.

第7章：从用户获得输入

- Johnson, Jeff. *GUI Bloopers*. San Francisco: Morgan Kaufmann, 2000.
- Johnson, Jeff. *Web Bloopers: 60 common Web Design Mistakes, and How To Avoid Them*. San Francisco: Morgan Kaufmann, 2003.
- Fowler, Susan, and Victor stanwick. *Web Application Design Handbook: Best Practices for Web-Based software*. San Francisco: Morgan Kaufmann, 2004. (chapter 3 talks about forms.)
- 37signals. *Defensive Design for the Web: How to Improve Error Messages, Help, Forms, and Other Crisis Points*. Berkeley, CA: New Riders, 2004. (Chapter 2, and Guidelines 9 and 10, discuss some of the same ideas used in this book's patterns.) Baxley, Bob. *Making the Web Work: Designing Effective Web Applications*. Indianapolis: Sams Publishing, 2002. (chapter 8 discusses forms.)

第8章：建造器和编辑器

- Cooper, Alan, Robert Riemann. *About face 2.0: The Essentials of Interaction Design*. Indianapolis: Wiley Publishing, Inc., 2003. (There really aren't many books that discuss this topic.)

第9章：修饰外观

- Norman, Donald A. *Emotional Design: Why We Love (or Hate) Everyday Things*. New York: Basic Books, 2004.
- Shea, David, and Molly Holzschlag. *The Zen of CSS Design: Visual Enlightenment for the Web*. Berkeley, CA: New Riders, 2005.
- Williams, Robin. *The Non-Designer's Design Book: Design and Typographic Principles for the Visual Novice*. Berkeley: Peachpit Press, 2004.
- Bringhurst, Robert. *The Elements of Typographic Style*, version 3.0. Vancouver: Hartley & Marks, 2004.
- Hashimoto, Alan. *Visual Design Fundamentals: A Digital Approach*. Hingham, MA: Charles River Media, 2004.
- McCloud, Scott. *Understanding Comics*. New York: Harpercollins, 1994.

其他模式

下面是一些有名的模式语言和书籍。其中只有一本，就是著名的*Design Patterns*（设计模式）一书是关于计算机的。但是也还有一些其他的计算机相关模式书籍供你查找。

Alexander, Christopher, Sara Ishikawa, Murray Silverstein, Max Jacobson, Ingrid Fiksdahl-King, Shlomo Angel. *A Pattern Language*. New York: Oxford University Press, 1977.

Alexander, Christopher. *The Timeless Way of Building*. New York: Oxford University Press, 1979.

Gamma, Erich, Richard Helm, Ralph Johnson, John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Reading, MA: Addison-Wesley, 1995.

Van Duyne, Douglas, James A. Landay, Jason I. Hong. *The Design of Sites: Patterns, Principles, and Processes for Crafting a Customer-Centered Web Experience*. Reading, MA: Addison-Wesley, 2002.

<http://www.welie.com/patterns/gui/index.html> Martijn van Welie的模式集合。

http://www.mit.edu/~jtldwell/common_ground.html 这本模式书籍的第一个早期版本。

<http://time-tripper.com/uipatterns/> 这本模式书籍的第二个早期版本。

<http://geography.uoregon.edu/datographics/patterns> 一小组数据可视化方面的模式。

<http://www.conservatoineconomy.org/> 这个大型的“pattern language for sustainability（可支持性模式语言）”和软件没有什么关系。但它涉及了本地和区域规划的方方面面。这是另一个领域正在运行的模式语言。很好的例子。

<http://iasummit.org/2005/finalpapers/52.Presentation.pdf> 这篇论文的标题为“Implementing a Pattern Library in the real World: A Yahoo! case study（在现实世界中实现一个模式库——Yahoo!案例研究）”，它描述的不只是是一组模式，而是开发交互设计模式的过程。

<http://usability.gov/guidelines/index.html> 这些基于研究的Web设计和可用性指导原则来自美国公共健康行政部门（Department of Health and Human services, DHHS）。它并不是一组模式，而是一些小规模的，定义良好的建议。就象本书中的模式一样，其中一些指导原则互相矛盾，你得动用自己的判断（和可用性测试）来决定哪些原则可以适用于你的实际情况中。

NUMBERS

80/20 rule, (80/20法则), 57, 65

A

accessibility (可访问性)

high-contrast color schemes and large fonts (高对比的配色方案和大字体), 288

JAWS, 288

styles (风格), 288-289
(也参见keyboards)

Action Panels (动作面板), 132, 136, 140-143

actions/commands (动作/命令)

Action Panels (动作面板), 132, 136, 140-143

affordances (启示), 135

Button Groups (按钮分组), 131, 136-141

buttons (按钮), 131, 136-141, 144-145

Cancelling (可取消性), 136, 149, 151-152

Command History (命令历史), 124, 136, 155-158

command line interfaces (命令行界面), 133

conventions (习惯用法/惯例), 135

creativity (创意), 133-135

double-clicking (双击), 132-133

drag-and-drop (拖拽并释放), 133

drop shadows (拖拽阴影), 135

dropdown menus (下拉菜单), 132

idioms (习惯用法), 133

invisible actions (不可见的动作), 132-133

interface architecture (界面架构)

lists of actions (动作列表), 24-26

lists of subject categories (主题类别列表), 26

lists of tools (工具列表), 27

keyboards (键盘), 132-133

links (链接), 132

Macros (宏), 17, 136, 156, 158-159

menu bars (菜单条), 131-132

mouse pointers (鼠标指针), 135

Multi-Level Undo (多级撤销), 136, 153-155

Alternative Views (可选视图), 40

Escape Hatch (逃生舱), 86

Macros (宏), 157-158

Property Sheet (属性表), 122

users (用户), 11

overview (总览), 131

patterns overview (模式总览), 136

pop-up menus (弹出菜单), 132

Preview (预览), 44, 136, 147-148, 233

Progress Indicator (进度提示), 65, 136, 149-151

Prominent Done Button (突出的“完成”按钮), 136, 144-145

Scrollbar (滚动条), 134, 139, 153

Smart Menu Items (智能菜单项), 136, 146, 152, 154-155

toolbars (工具条), 132

tooltip (工具提示), 135

typed commands (键入的命令), 133

visual impairment (视觉受损), 132

advice (user behavior) (建议, 用户行为), 18-19

affect (影响), 270

affordances (actions/commands) (启示), 135

alignment (layout) (对齐), 94-96, 99

closure (闭合性), 95-96

alternating row colors (交替的表格行颜色), 187

button groups (按钮分组), 137

styles (风格), 269

continuity (连续性), 95-96

alternating row colors (交替的表格行颜色), 187

local zooming (局部缩放), 184-185

preattentive variables (前摄变量), 163

right-left alignment (右对齐左对齐), 116

styles (风格), 269

Gestalt theory (格式塔理论), 94-96,

269

Prominent Done Button (突出的“完成”按钮), 144

Proximity (相邻性), 94-95, 97

Action Panels (动作面板), 141

alternating row colors (交替的表格行颜色), 187

button groups (按钮分组), 137

Datatips (数据提示), 172

multi-Y Graph (多Y轴图表), 198

responsive enabling (响应式允许), 126

right-left alignment (右对齐左对齐), 116

styles (风格), 269

similarity (相似性), 95, 97

button groups (按钮分组), 137

Multi-Y Graph (多Y轴图表), 198

preattentive variables (前摄变量), 163, 166

styles (风格), 269, 287

visual hierarchy (视觉层次结构), 90-91

Alternating row Colors (交替的表格行颜色)

(information graphics), 817-188

Alternative Views (可选视图)

(information architecture), 30, 39-41, 173

analyzing user goals (分析用户目标), 3-5

angles (角度) (styles), 279, 282-287

Borders that Echo Fonts (边界回应字体), 302

Corner Treatments (角落处理), 297-300

Skins (皮肤), 311

shapes (外形), 279

animation (动画)

Animated Transitions (动画转换), 63, 84-85, 173, 185

progress indicator (进度提示), 149

Annotated Scrollbar (注释滚动条) (navigation), 55, 63, 80-81, 173

appearance (外观), importance of (重要性), 269

arranging (排列)。参见 sorting
audience (受众)。参见 users
Autocompletion (controls), (自动完成) 127, 207-208, 227-229
axes (轴) (information graphics), 165, 171-172, 177, 198-201

B

Back button (后退按钮), 10, 12, 14, 43
backgrounds (背景) (styles), 280, 286-288, 290-293
Deep Background (深色背景), 288, 291-293, 307, 311
Few Hues, Many Values (少一点色彩, 多一些价值), 295
Hairlines (发丝), 303-304
Baxley, Bob, 69
behavioral patterns (行为模式)
changes in midstream (中途变卦), 12-13
deferred choices (延后选择), 13-14
habituation (习惯), 14-15
incremental construction (递增构建), 14, 243
instant gratification (即时满足), 11
keyboard only (只支持键盘), 17-18
other people's advice (旁人建议), 18-19
overview (总览), 10
prospective memory (前瞻记忆), 16-17
safe exploration (安全探索), 11
satisficing (满意即可), 11-12
spatial memory (空间记忆), 15-16, 166
streamlined repetition (简化重复工作), 17
blindness (盲人)。参见 visual impairment
borders (边界) (styles), 280, 283, 287-288
Borders that Echo Fonts (边界回应字体), 287-288, 300-302
Corner Treatments (角落处理), 298-299
Hairlines (发丝), 303
patterns overview (模式总览), 290

brand identity (品牌识别), 270
branding (品牌建立) (styles), 270, 279
Color-Coded Sections (颜色编码的栏目), 82-83
cultural references (文化含义), 286
Deep Background (深色背景), 291
Escape Hatch (逃生舱), 87
layout (布局), 92, 100
signposts (路标), 55
Breadcrumbs (面包屑层级结构) (navigation), 55, 78-79
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns overview (模式总览), 63
Sequence Maps (序列地图), 76
visual framework (视觉框架), 101
browsing (浏览) (information graphics), 166-168
drilling down (深入), 167-168, 196, 204
focus plus context (焦点加上下文), 166
opening/closing (打开/关闭), 167, 186
panning (平移), 166-168, 175, 181
scrollbars (滚动条)。参见 scrollbars
zooming (缩放), 161, 167-168
Data Brushing (数据刷), 181, 183
Datatips (数据提示), 176
Local Zooming (局部缩放), 184-185
patterns overview (模式总览), 173
Overview Plus Detail (总览加细节), 174-175
Treemap (树状地图), 204
builders。参见 editors
Button Groups (按钮分组), 131, 136-141
buttons (按钮)
actions/commands (动作/命令), 131, 136-141, 144-145
Button Groups (按钮分组), 131, 136-141
Cancel (取消), 149-152
Check (检查), 178-180

Close (关闭), 137-138
Done (完成), 18
OK (确认), 75, 93, 101
iconic (图标的), 35, 210
Prominent Done Button (突出的“完成”按钮), 136, 144-145
Radio (单选按钮), 179-180
Submit (提交), 18
toggle (切换按钮), 212

C

calendar (日历)
control (控件), 217
DateLens (日期透镜), 185
Cancelability (可取消性)
(actions/commands), 136, 149, 151-152
Canvas Plus Palette (画布加调色板工具条) (information architecture), 25, 29-30, 34-35, 243
Card Stack (卡片堆)
information architecture (信息架构), 33-34, 44
information graphics (信息图形), 196
layout (布局), 98-99, 101, 108-112
navigation (导航), 57
Cascading Lists (级联列表)
forms (表单), 212-213, 234
information architecture (信息架构), 33, 37
information graphics (信息图形), 162, 167, 195-197
layout (布局), 129
case studies (案例分析) (user research), 6
cell phone (手机), 5, 26-31, 36
Center Stage (中央舞台) (layout), 67, 91, 99, 103-106, 141
changes in midstream (中途变卦) (user behavior), 12-13
Checkbox (复选框), 178-179, 207, 210
Clear Entry Points (清楚的入口点) (navigation), 56, 63-65
Cleveland, William, 201
Closable Panels (可关闭面板)
actions/commands (动作/命令),

- 142
- information architecture (信息架构), 37, 45, 49-51
- layout (布局), 98-99, 108-109, 111-113
- navigation (导航), 84
- Closing (关闭) (information graphics), 167, 186
- closure (闭合性) (layout), 95-96
- alternating row colors (交替的表格行颜色), 187
- button groups (按钮分组), 137
- styles (风格), 269
(参见 also Gestalt principles)
- cognitive cost (认知代价) (navigation), 56-62
- color-blindness (色盲)。参见 visual impairment
- color chooser (颜色选择器), 233
- Color-Coded Sections (颜色编码的栏目) (navigation), 55, 63, 82-83
- colors (颜色)
color-blindness (色盲)。参见 visual impairment
- colour (continued)
- Color-Coded Sections (颜色编码的栏目) (navigation), 55, 63, 82-83
- combinations of hues (色调组合), 281
- dark vs. light background (深背景vs. 浅背景), 280
- and emotional response (以及情感上的反应), 270
- Few Hues, Many Values (少一点色彩。多一些价值), 281
- gradients (渐变), 292
- high contrast vs. low contrast (强对比vs. 弱对比), 280
- as preattentive variable (作为前摄变量), 163
- red vs. green (红色 vs. 绿色), 280
- saturated vs. unsaturated (饱和 vs. 不饱和), 280
- visual design (视觉设计) (styles), 279-288, 291-296, 300-308
- visual hierarchy (视觉层次) (layout), 92
- warm vs. cool (暖色调 vs. 冷色调), 280
- combo box (组合框), 18, 214, 222, 228, 230
- Command History (命令历史)
(actions/commands), 124, 136, 155-158
- command line interfaces (命令行界面), 133, 157
- commands (命令)。参见 actions/commands
- commuting (交通) (navigation), 55
- Composite selection (组合选择)
(editors), 248, 253-254
- computers/magazines layout
comparison (计算机/杂志布局对比), 98
- Confirmation dialog box (确认对话框), 15, 152
- Constantine, Larry, 72
- Constrained Resize (强制调整大小)
(editors), 248, 259-260
- content organization (内容组织)
idioms (习惯用法), 22-23
- information architecture (信息架构)
80/20 rule (80/20法则), 57, 65
- Alternative Views (可选视图), 30, 39-41, 173
- Canvas Plus Palette (画布加调色板工具), 25, 29-30, 34-35, 243
- Extras on Demand (需要时显示)。
参见 Extras on Demand
- Intriguing Branches (有趣的分支), 27, 30, 47-48
- Multi-Level Help (多级帮助), 9, 19, 30, 47-53
- One-Window Drilldown (单窗口深入)。参见 One-Window Drilldown
- Two-Panel selector (双面板选择器)。参见 Two-Panel selector
- Wizards (向导)。参见 Wizards
- lists of actions (动作列表), 24-26
- lists of objects (对象列表), 23-24
- lists of subject categories (主题类别列表), 26
- lists of tools (工具列表), 27
- mental models (心智模型), 22-23
- noun/verb model (名词/动词模型), 22-23
- overview (总览), 22-23
- context switches (上下文切换)
(navigation), 56-62, 100, 109, 123
- continuity (连续性)
alternating row colors (交替的表格行颜色), 187
- local zooming (局部缩放), 184-185
- preattentive variables (前摄变量), 163
- right-left alignment (右对齐左对齐), 116
- styles (风格), 269
(也参见 Gestalt principles)
- contrasting colors (颜色对比), 67, 91-93
- Contrasting Font Weights (字体大小对比) (styles), 306, 307
- controls (控件)
Autocompletion (自动完成), 127, 207-208, 227-229
- choice (选择), 121-122, 209
- dates/times, (日期/时间) 217
- Dropdown Chooser (下拉选择器), 132, 216-217, 230-232
- Fill-in-the-Blanks (填空), 216, 222-223
- Forgiving Format (容错格式), 215, 217, 219-220, 224, 239
- Good Defaults (良好的默认值), 208, 224-225, 237-238
- information architecture (信息架构), 43-44
- information graphics (信息图形), 193, 205
- users (用户), 13
- Illustrated Choices (图示选择), 208, 233-234
- actions/commands (动作/命令), 148
- information graphics (信息图形), 196
- styles (风格), 285
- Input Hints (输入提示), 208, 216, 224-225
- Forgiving Format (容错格式), 219-220
- information architecture (信息架构), 50
- layout (布局), 91
- Same-Page Error Messages (错误显示在同一页), 239
- Input Prompts (输入提醒), 216, 224-226
- Forgiving Format (容错格式), 219-220
- information architecture (信息架构), 50

- 构), 49-50
information graphics (信息图形), 193
Same-Page Error Messages (错误显示在同一页), 239
List Builder (列表建造器), 213, 235-236
lists of items (元素列表), 210-214
numbers (数字), 215-216
overview (总览), 207
patterns overview (模式总览), 218
principles (原则), 209-210
Same-Page Error Messages (错误显示在同一页), 239-241
Structured Format (结构化的格式), 208, 215, 217, 219-221, 224
text (文字), 214-215
conventions (习惯用法) (actions/commands), 135
Cooper, Alan, 153
copy (复制), 131, 245, 248, 266
Corner Treatments (角落处理) (styles), 287-288, 297-300, 302, 309, 311
cost (代价) (cognitive), 56-62
creativity (创意)
actions/commands (动作/命令), 133-135
 styles (风格), 287-289
Csikszentmihalyi, Mihaly, 313
Css Zen Garden (CSS禅意花园) (styles), 270-279
cultural references (文化含义) (styles), 279, 283, 286, 295
curves (曲线) (styles), 282-285, 287
 Borders that Echo Fonts (边界回应字体), 300
 Corner Treatments (角落处理), 297-298
 Hairlines (发丝), 305
 Skins (皮肤), 310
cut (剪切), 131, 154
- ## D
- Data Brushing (数据刷) (information graphics), 170, 172-173, 179, 181-183
data values (数据值) (information graphics), 171-172
database, 31, 120, 151, 153-154, 158, 209
Datatips (数据提示) (information graphics), 172-173, 176-177, 185, 203-204
dates/times (日期/时间)
 controls (控件), 217
 entering (输入), 209-210, 217
Deep Background (深色背景) (styles), 288, 291-293, 307, 311
defaults (默认值) (Good Defaults), 208, 224-225, 237-238
information architecture (信息架构), 43-44
information graphics (信息图形), 193, 205
 users (用户), 13
deferred choices (延后选择) (user behavior), 13-14
Detail View Direct Navigation (详细视图直接导航), 72
design process (设计过程)
 use goals (用户目标), 4
 user motivation (用户动机), 7
design models (设计模型), 7, 24, 32
dexterity (灵敏度)
 control choice and (控件选择及), 210, 217
 moving objects and (移动对象及), 261
Diagonal Balance (对角平衡) (layout), 99, 118-119, 145
dialog boxes (对话框)。参见 panes/windows
direct manipulation (直接操作)
 actions/commands (动作/命令), 131, 135
 editors (编辑器), 244-245, 248, 253, 259, 263
 forms (表单), 208
 information graphics (信息图形), 175, 179, 195
 layout (布局), 120-121
 precision and, (精确度) 120-121, 263
 direct observation of users (直接用户观察), 17
discoverability (可发现性), 133, 141
dividing stuff up (把内容分开)。参见 content organization (组织)
documentation (文档), 10
double-clicking (双击) (actions/commands), 132-133
drag-and-drop (拖拽和释放) (actions/commands), 133, 245, 253, 259
- drilling down (深入)
information graphics (信息图形), 167-168, 196, 204
One-Window Drilldown (单窗口深入) (information architecture), 24-25, 29-32, 36-38
breadcrumbs (面包屑层级结构), 78
Hub and spoke (中心和辐条), 68
modal Panel (模态面板), 75
Pyramid (金字塔), 71-72
drop shadows (actions/commands), 135
Dropdown Chooser (下拉选择器) (controls), 132, 216-217, 230-232
dropdown control, (下拉控件) 132, 140, 179, 191, 193, 208, 210, 218, 222(也参见 combo box)
Dynamic Queries (动态查询) (information graphics), 170, 178-181, 203-205
dynamics (动态) (layout), 98-99
- ## E
- Easter egg (复活节彩蛋), 46-47
Edit-In-Place editors (就地编辑编辑器), 248-250
editors (编辑器)
 Composite Selection (组合选择), 248, 253-254
 Constrained Resize (强制重定义大小), 248, 259-260
 direct manipulation (直接操作), 244-245, 248, 253, 259, 263
 actions/commands (动作/命令), 131, 135
 forms (表单), 208
 information graphics (信息图形), 175, 179, 195
 layout (布局), 120-121
 Edit-In-Place (就地编辑), 248-250
 Guides (对齐指示线), 135, 248, 261-265
 idioms (习惯用法), 243
 Magnetism (磁性吸附), 185, 248, 261-264
 modes (模式), 244-245
 Composite Selection (组合选择), 253
 Constrained Resize (强制调整大小), 259-260

- Guides (对齐指示线), 264
One-Off Mode (一次性模式), 35,
 245, 248, 255-258
patterns overview (模式总览),
 248
 Smart Selection (智能选择), 251
 Spring-Loaded Mode (弹性载入模
 式), 18, 35, 245, 248, 257-258
overview (总览), 243-244
 Paste Variations (粘贴变种), 248,
 266-267
patterns overview (模式总览), 247
 selection (选择), 244-248
 Smart Selection (智能选择),
 247-248, 251-252
 WYSIWYG editors (所见即所得编
 辑器), 244
 Edit-In-Place (就地编辑), 249
 forms (表单), 226
 Guides (对齐指示线), 264
 information architecture (信息架
 构), 41
 layout (布局), 120
 Paste Variations (粘贴变种), 267
 patterns (模式), 248
efficiency of use (使用效率), 14
email
 addresses (地址), 227
 clients (客户), 4, 9, 24
 messages (消息), 23, 31-32
emotional and visceral
 response (情感反应和内在反
 应) (参见 affect)
encoding data (数据编码), 165
environmental clues (环境线索), 56
error message (错误消息), 225, 237,
 239-240
Escape Hatch (逃生舱) (navigation),
 12, 56, 63, 86-87
examples (示例) (styles), 270-279
expectations (期望) (users), 7-9
Extras on Demand (需要时显示)
 80/20 rule (80/20法则), 57, 65
 information architecture (信息架
 构), 26, 30, 45-46, 57, 65
extras on Demand (需要时显示)
 (continued)
 information graphics (信息图形),
 173
 layout (布局), 108, 111
 styles (风格), 204-205
- users (用户), 13
- F**
- feedback (反馈), 14, 178-179, 245,
 260
Few Hues, Many Values (少一点色彩。
 多一些价值)
 (styles), 281, 294-296
file finder dialog (文件查找对话框),
 191
Fill-in-the-Blanks (填空) (controls),
 216, 222-223
filtering (过滤), 161, 166, 169-171,
 也参见 **searching**
 Dynamic Queries (动态查询),
 178-180
 Jump to Item (跳到目标对象), 191
 Treemap (树状视图), 204-205
fisheye lens (鱼眼透镜), 184
flow (流), 243-244
focal points (焦点) (visual flow),
 92-94, 99
 angles (角度), 284
 strength (力量), 292-293, 305
 style (风格), 269
focus plus context (焦点加上下文)
 (information graphics (信息图
 形)), 166
fonts (字体)。参见 **typography/fonts**
Forgiving Format (容错格式)
 (controls), 215, 217, 219-220, 224,
 239
forms (表单)
 controls (控件)
 Autocompletion (自动完成), 127,
 207-208, 227-229
 dates/times (日期/时间), 217
 Dropdown Chooser (下拉选择
 器), 132, 216-217, 230-232
 Fill-in-the-Blanks (填空), 216,
 222-223
 Forgiving Format (容错格式), 215,
 217,
 219-220, 224, 239
 Good Defaults (良好的默认值)。
 参见 **Good Defaults**
Illustrated Choices (图示选择)。
 参见 **Illustrated Choices**
Input Hints (输入提示)。参见
Input Hints
Input Prompts (输入提醒)。参见
- Input Prompts**
- List Builder** (列表建造器), 213,
 235-236
lists of items (元素列表), 210-214
numbers (数字), 215-216
overview (总览), 207
patterns overview (模式总览),
 218
principles (原则), 209-210
Same-Page Error Messages (错误显
 示在同一页), 239-241
Structured Format (结构化格式),
 208, 215, 217, 219-221, 224
text (文字), 214-215
overview (总览), 207
patterns overview (模式总览), 218
principles (原则), 207-209
- G**
- Gestalt principles** (格式塔原则)
 (layout), 94-96, 269 (参见
 also proximity; similarity;
 continuity; closure)
Global Navigation (全局导航), 55, 57,
 66-69
Color-Coded Sections (颜色编码的栏
 目), 83
 layout (布局), 101
patterns overview (模式总览), 63
 users (用户), 12
goals (目标) (users), 3-5
Good Defaults (良好的默认值), 208,
 224-225, 237-238
 information architecture (信息架
 构), 43-44
 information graphics (信息图形),
 193, 205
 users (用户), 13
good signage (好路标), 55
graphic design (图形设计), 22
graphic designer (图形设计师), 89,
 129, 170, 181, 233
graphics, 参见 **images**;
 information graphics (信息图
 形)
graphs (图形), 84-85, 162, 166, 168,
 199, 201
 (参见 also plots; Multi-Y Graph,
 information graphics (信息图
 形))
green/red colors (绿/红颜色), 240,

- 280
Group box (分组框), 92, 94, 253, 297
grouping (分组) (layout), 94-96, 99,
 144
 closure (闭合性), 95-96
 alternating row colors (交替的表格行颜色), 187
 button groups (按钮分组), 137
 styles (风格), 269
Continuity (连续性), 95-96
 alternating row colors (交替的表格行颜色), 187
 local zooming (局部缩放), 184-185
 preattentive variables (前摄变量), 163
right-left alignment (右对齐左对齐), 116
 styles (风格), 269
Gestalt theory (格式塔理论), 94-96,
 269
proximity (相邻性), 94-95, 97
 Action Panels (动作面板), 141
 alternating row colors (交替的表格行颜色), 187
 button groups (按钮分组), 137
 Datatips (数据提示), 172
 Multi-Y Graph (多Y轴图表), 198
 responsive enabling (响应式允许), 126
 right-left alignment (右对齐左对齐), 116
 styles (风格), 269
similarity (相似性), 95, 97
 button groups (按钮分组), 137
 Multi-Y Graph (多Y轴图表), 198
 preattentive variables (前摄变量), 163, 166
 styles (风格), 269, 287
GUI builder, 244, 246, 260, 263
GUI toolkit (GUI工具箱), 32, 210, 228
Guides (对齐指示线) (editors), 135,
 248, 261-265
- H**
- habituation (习惯) (user behavior),
 14-15
Hairlines (发丝) (styles), 284, 303-
 305, 307
Help (帮助)
 documentation (文档), 108
- Help text (帮助文本), 47, 49, 51,
 227
Multi-Level Help (多级帮助), 9, 19,
 30, 47-53
- Hierarchy (Cascading Lists)**
- forms (表单), 212-213, 234
 - information architecture (信息架构), 33, 37
 - information graphics (信息图形), 162, 167, 195-197
 - layout (布局), 129
(参见 also visual hierarchy; tree)
- history (历史) (Command History),
 124, 136, 155-158
- home page (首页), 67, 86
- HTML, 210, 267, 270
 - available controls (可用控件), 210-211
 - and WYSIWYG editing (及所见即所得编辑), 244-245
- Hub and Spoke navigation (中心和辐条导航)**, 27, 37, 68-70
- Clear Entry Points (清楚的入口点), 65
- Escape Hatch (逃生舱), 86
- Global Navigation (全局导航), 66-67
- patterns (模式), 63
- users (用户), 12
- I**
- Icons (图标) (styles)**, 270, 285-286,
 288
- Corner Treatments (角落处理), 298
- Deep Background (深色背景), 291
- Few Hues, Many Values (少一点色彩。多一些价值), 295
- Hairlines (发丝), 303-304
- Skins (皮肤), 309-310
- Idioms (习惯用法)**
- actions/commands (动作/命令), 133
 - defined, 22
 - editors (编辑器), 243
 - interface architecture (界面架构), 22-23
- Illustrated Choices (图示选择)**, 208,
 233-234
- actions/commands (动作/命令), 148
 - information graphics (信息图形), 196
- styles (风格), 285
- images (图片)
 参见 information graphics
- visual design (视觉设计) (styles),
 279-280, 285-288
- Borders that Echo Fonts (边界回应字体), 301
- Corner Treatments (角落处理), 297
- Deep Background (深色背景), 291-293
- Font Weights (字体重量), 307
- Hairlines (发丝), 303
- Skins (皮肤), 309
- Immediacy (直接)**, 215, 243
- incremental construction (增量构建)
(user behavior), 14, 243
- Information architecture (信息架构)**
- content organization (内容组织)
 80/20 rule (80/20法则), 57, 65
 - Alternative Views (可选视图), 30,
 39-41, 173
- Canvas Plus Palette (画布加调色板工具条), 25, 29-30, 34-35, 243
- Extras on Demand (需要时显示)**。参见 Extras on Demand
- idioms (习惯用法), 22-23
- Intriguing Branches (有趣的分支), 27, 30, 47-48
- lists of actions (动作列表), 24-26
- lists of objects (对象列表), 23-24
- lists of subject categories (主题类别列表), 26
- lists of tools (工具列表), 27
- mental models (心智模型), 22-23
- Multi-Level Help (多级帮助), 9,
 19, 30, 47-53
- noun/verb model (名词/动词模型), 22-23
- One-Window Draggable。参见 One-Window Draggable
- overview (总览), 22-23
- Two-Panel Selector (双面板选择器)。参见 Two-Panel Selector
- Wizards (向导)。参见 Wizards
- learnability, 30, 49-53
- navigation (导航), 63
- overview (总览), 21
- patterns overview (模式总览), 30
- physical structure (物理结构)

- Alternative Views (可选视图), 30, 39-41, 173
Canvas Plus Palette (画布加调色板工具条), 25, 29-30, 34-35, 243
multiple windows (多窗口), 28-29, 33, 36-37, 100
One-Window Drilldown. 参见 One-Window Drilldown
one-window paging (单窗口页面), 28-29, 37
overview (总览), 28
tiled panes (平铺面板). 参见 tiled panes
Two-Panel Selector (双面板选择器). 参见 Two-Panel Selector
visual hierarchy (视觉层次结构). 参见 visual hierarchy
- Information graphics (信息图形)**
- Alternating Row Colors (交替的表格行颜色), 187-188
 - axes (轴), 165, 171-172, 177, 198-201
 - browsing (浏览), 166-168
 - drilling down (深入), 167-168, 196, 204
 - focus plus context (焦点加上下文), 166
 - opening/closing (打开/关闭), 167, 186
 - panning (平移), 166-168, 175, 181
 - scrollbars (滚动条). 参见 scrollbars
 - zooming (缩放). 参见 zooming
- Cascading Lists (级联列表), 162, 167, 195-197
- information architecture (信息架构), 33, 37
- forms (表单), 212-213, 234
- layout (布局), 129
- Data Brushing (数据刷), 170, 172-173, 179, 181-183
- data values (数据值), 171-172
- Datatips (数据提示), 172-173, 176-177, 185, 203-204
- Dynamic Queries (动态查询), 170, 178-181, 203-205
- filtering (过滤), 161, 166, 169-171. 也参见 searching
- Dynamic Queries (动态查询), 178-180
- Jump to Item (跳到目标对象), 191
- Treemap (树状视图), 204-205
- Jump to Item (跳转到对象), 170, 191-193
- labels (标签), 167, 171-172
- Datatips (数据提示), 177
- Dynamic Queries (动态查询), 179
- Multi-Y Graph (多Y轴图表), 199
- Small Multiples (大量小对象), 201
- Treemap (树状地图), 203
- layers (图层), 166, 170, 176, 182, 185
- legends (图例), 171
- Local Zooming (局部缩放), 167, 176, 183-185
- Multi-Y Graph (多Y轴图表), 162, 198-199
- navigation (导航), 166-168
- Information graphics (信息图形) (continued)**
- drilling down (深入), 167-168, 196, 204
 - focus plus context (焦点加上下文), 166
 - opening/closing (打开/关闭), 167, 186
 - panning (平移), 166-168, 175, 181
 - scrollbars (滚动条). 参见 scrollbars
 - zooming (缩放). 参见 zooming
- New-Item Row (新对象行), 193-194, 213
- organization (组织), 162
- overview (总览), 161-162
- Overview Plus Detail (总览加细节), 167, 174-175, 185
- information architecture (信息架构), 33
- navigation (导航), 56, 63, 80
- patterns overview (模式总览), 173
- preattentive variables (前摄变量), 163-166, 182
- rulers (尺子), 172
- scales (比例), 172
- searching (搜索). 也参见 filtering
- information graphics (信息图形), 161, 166, 168-171
- Jump to Item (跳到目标对象), 192
- Local Zooming (局部缩放), 184
- Overview Plus Detail (总览加细节), 175
- Sortable Table (可排序表格), 189
- Small Multiples (大量小对象), 200-202, 287
- Sortable Table (可排序表格), 33, 40, 161-162, 168-169, 190
- sorting (排序), 162, 168-169
- Dynamic Queries (动态查询), 178
- information architecture (信息架构), 23-25, 27, 32-33, 39, 40
- navigation (导航), 73
- patterns overview (模式总览), 173
- sortable Table (可排序表格) (information graphics), 33, 40, 161-162, 168-169, 190
- Tree Table (树状表格), 197
- timelines (时间轴), 162, 172, 198
- Tree Tables (树状表格), 33, 162, 197
- Treemaps (树状视图), 162, 179, 203-205
- (参见 also visualization; plots; graphs)
- Input (输入). 参见 forms**
- Input Hints (输入提示), 208, 216, 224-225**
- Forgiving Format (容错格式), 219-220**
- information architecture (信息架构), 50
- layout (布局), 91
- Same-Page Error Messages (错误显示在同一页), 239
- Input Prompts (输入提醒), 216, 224-226**
- Forgiving Format (容错格式), 219-220**
- information architecture (信息架构), 49-50
- information graphics (信息图形), 193
- Same-Page Error Messages (错误显示在同一页), 239
- instant gratification (即时满足) (user behavior), 11
- instant messaging (即时消息), 133, 244
- interface idioms (界面习惯用法),

- 22-23, 246
- Interviews (访谈) (user research)**, 6
- Intriguing Branches (有趣的分支) (information architecture)**, 27, 30, 47-48
- Invisible actions, (不可见的动作)**, 132-133
- ## J-K
- Johnson, Jeff**, 124, 314
- Jump to Item (跳转到对象) (information graphics)**, 170, 191-193
- Jumps (跳转)**
- Jump to Item (跳转到对象) (information graphics)**, 170, 191-193
- number of (跳转次数), 56-62
- keyboards (键盘)**
- accelerators (加速键), 131
 - actions/commands (动作/命令), 132-133
 - keyboard only (只支持键盘) user behavior, 17-18
 - mnemonics (记忆法), 18, 257
 - shortcuts (快捷方式), 17-18, 133
- kiosk (信息亭)**, 8, 36, 128
- knowledge (知识) (users)**, 7-9
- "Knowledge in the world is better than knowledge in the head"**, 228
- Krug, Steve**, 313
- ## L
- labels (标签) (information graphics)**, 167, 171-172, 248-249
- controls (控件), 207-208, 215, 222, 225
- Datatypes (数据提示), 177
- Dynamic Queries (动态查询), 179
- information graphics (信息图形), 171-172
- Multi-Y Graph (多Y轴图表), 199
- Small Multiples (大量小对象), 201
- Treemap (树状视图), 203
- layers (图层) (information graphics)**, 166, 170, 176, 182, 185
- layout (布局)**
- alignment (对齐)。参见 alignment
 - Card Stack (卡片堆), 98-99, 101, 108-112
 - information architecture (信息架构), 33-34, 44
 - information graphics (信息图形), 196
 - navigation (导航), 57
 - Center Stage (中央舞台), 67, 91, 99, 103-106, 141
 - Closable Panels (可关闭面板), 98-99, 108-109, 111-113
 - actions/commands (动作/命令), 142
 - information architecture (信息架构), 37, 45, 49-51
 - navigation (导航), 84
- computer/magazine
- comparison (比较), 98
- Diagonal Balance (对角平衡), 99, 118-119, 145
- dynamics (动态), 98-99
- grid (网格), 101, 114
- grouping (分组), 94-96, 99, 144
- closure (闭合性), 95-96, 137, 187, 269
- continuity (连续性)。参见 continuity
- Gestalt theory (格式塔理论), 94-96, 269
- Proximity (相邻性)。参见 proximity
- similarity (相似性)。参见 similarity
- Liquid Layout (流式布局), 99, 128-129, 185
- Movable Panels (可移动的面板), 98-99, 108-109, 114-115, 122
- actions/commands (动作/命令), 141
- information architecture (信息架构), 37, 51
- layout (布局), 98-99, 108-109, 114-115, 122
- users (用户), 16
- navigation (导航), 63
- overview (总览), 89, 96-98
- Property Sheet (属性表)
- editors (编辑器), 249
 - forms (表单), 208-209
- information architecture (信息架构), 32
- layout (布局), 99, 109, 112, 120-122, 124
- Responsive Disclosure (响应式展开)**
- information architecture (信息架构), 43, 46
 - layout (布局), 98-99, 122-127
 - users (用户), 15
- Responsive Enabling (响应式允许)**, 43, 98-99, 123, 125-127
- Right/Left Alignment (右对齐/左对齐)**, 99, 116-117, 121-122
- Titled Sections (带标题的栏目)**。参见 Titled Sections
- visual flow (视觉流) (layout)**, 92-94, 97, 99
- focal points (焦点)。参见 focal points
- Prominent Done Button (突出的“完成”按钮), 144-145
- style (风格), 269
- Two-Panel Selector (双面板选择器), 32
- Visual Framework (视觉框架)**, 67, 82, 91, 99-102, 144
- visual hierarchy (视觉层次结构)**, 89-92, 98-99, 144, 205, 269, 306
- alignment (对齐), 90-91
- colors (颜色), 92
- information architecture (信息架构)**。参见 information architecture
- spatial relationships (空间关系), 91-92, 99
- typography/fonts (字体), 90-91
- visual weight (视觉重量), 91
- whitespace (空白), 90-91
- learnability (可学习性) (information architecture)**, 30, 49-53
- legends (图例) (information graphics)**, 171
- links (链接) (actions/commands)**, 132
- Linux users (Linux 用户)**, 4
- Liquid Layout (流式布局)**, 99, 128-129, 185
- List Builder (controls)**, 213, 235-236
- Lists (列表)**
- of actions (动作列表) (interface architecture), 24-26
 - of items (元素列表) (controls), 210-214
- list control (控件列表), 230
- multicolumn (多栏列表), 138, 141

- of objects (对象列表) (interface architecture), 22-23
architecture), 23-24
ordered (有序列表), 213-214
of subject categories (主题类别列表) (interface architecture), 26
unordered (无序列表), 213-214
- lists of tools (工具列表) (interface architecture), 27
- Local Zooming (局部缩放) (information graphics), 167, 176, 183-185
- Lockwood, Lucy, 72
- look and feel (外观和感觉), 75, 100-101
- ## M
- Macros (宏) (actions/commands), 17, 136, 156, 158-159
- magazines/computers layout comparison (比较), 98
- Magnetism (磁性吸附) (editors), 185, 248, 261-264
- maps (地图), 56, 63
- Annotated Scrollbar (注释滚动条), 55, 63, 80-81, 173
 - Breadcrumbs (面包屑层级结构), 55, 78-79
 - Escape Hatch (逃生舱), 86
 - One-Window Drilldown (单窗口深入), 37
 - patterns overview (模式总览), 63
 - Sequence Maps (序列地图), 76
 - visual framework (视觉框架), 101
- Color-Coded Sections (颜色编码的栏目), 55, 63, 82-83
- Sequence Maps (序列地图), 55-56, 76-77
- Breadcrumbs (面包屑层级结构), 78
- Escape Hatch (逃生舱), 86
 - One-Window Drilldown (单窗口深入), 37
- patterns overview (模式总览), 63
- Visual Framework (视觉框架), 101
- Wizards (向导), 44
- marketing, user research (市场调研。
用户研究)
- comparison (比较), 7
- media player (多媒体播放器), 21, 23, 133, 287
- mental models (心智模型) (interface architecture), 22-23
- menu bars (菜单条)
- actions/commands (动作/命令), 131-132
 - Smart Menu Items (智能菜单项), 136, 146, 152, 154-155
- menus (菜单)
- pop-up (弹出式), 211-212
 - pull-right (右拉式), 140, 267
- micro and macro readings (宏观阅读和微观阅读), 174
- mobile devices (移动设备), 68
- (参见 also cell phone; PDAs; palmtops)
- Modal Panel (模态面板) (navigation), 56, 63, 69, 74-75, 86
- actions/commands (动作/命令), 151, 154
- forms (表单), 231, 235, 239
- layout (布局), 123
- users (用户), 12-15
- modes (模态) (editors), 244-245
- Composite Selection (组合选择), 253
 - Constrained Resize (强制调整大小), 259-260
 - Guides (对齐指示线), 264
 - One-Off Mode (一次性模态), 35, 245, 248, 255-258
- patterns overview (模式总览), 248
- problems (问题), 244, 245
- Smart Selection (智能选择), 251
- Spring-Loaded Mode (弹性载入模态), 18, 35, 245, 248, 257-258
- motifs (主题) (visual design), 279, 286-287, 290, 297-298, 300
- mouse pointers (鼠标指针) (actions/commands), 135
- Movable Panels (可移动的面板)
- actions/commands (动作/命令), 141
 - information architecture (信息架构), 37, 51
- Movable Panels (可移动的面板) (continued)
- layout (布局), 98-99, 108-109, 114-115, 122
 - users (用户), 16
- Mullet, Kevin, 118
- Multi-Level Help (多级帮助) (information architecture), 9, 19, 30, 47-53
- Multi-Level Undo (多级撤销), 136, 153-155
- Alternative Views (可选视图), 40
- Escape Hatch (逃生舱), 86
- macros (宏), 157-158
- Property Sheet (属性表), 122
- users (用户), 11
- multidimensional data (多维数据), 165, 173
- multiple windows (多窗口) (information architecture), 28-29, 33, 36-37, 100
- Multi-Y Graph (多Y轴图表) (information graphics), 162, 198-199
- ## N
- navigation (导航)
- 80/20 rule (80/20法则), 57, 65
 - Animated Transition (动画转换), 63, 84-85, 173, 185
 - Annotated Scrollbar (注释滚动条), 55, 63, 80-81, 173
- Breadcrumbs (面包屑层级结构) (navigation), 55, 78-79
- Escape Hatch (逃生舱), 86
- One-Window Drilldown (单窗口深入), 37
- patterns overview (模式总览), 63
- Sequence Maps (序列地图), 76
- visual framework (视觉框架), 101
- Clear Entry Points (清楚的入口点), 56, 63-65
- cognitive cost (认知代价), 56-62
- Color-Coded Sections (颜色编码的栏目), 55, 63, 82-83
- commuting (交通), 55
- context switches (上下文切换), 56-62, 100, 109, 123
- Detail View Direct Navigation (详细视图直接导航), 72
- Escape Hatch (逃生舱), 12, 56, 63, 86-87
- Global Navigation (全局导航), 55, 57, 66-69
- Color-Coded Sections (颜色编码的栏目), 83
- layout (布局), 101
- patterns overview (模式总览), 63
- users (用户), 12
- Hub and Spoke (中心和辐条), 27,

37, 68-70
Clear Entry Points (清楚的入口点), 65
Escape Hatch (逃生舱), 86
Global Navigation (全局导航), 66-67
patterns (模式), 63
users (用户), 12
information architecture (信息架构), 63
information graphics (信息图形), 166-168
drilling down (深入), 167-168, 196, 204
focus plus context (焦点加上下文), 166
opening/closing (打开/关闭), 167, 186
panning (平移), 166-168, 175, 181
scrollbars (滚动条)。参见 scrollbars
zooming (缩放)。参见 zooming
layout (布局), 63
Modal Panel (模态面板)
(navigation), 56, 63, 69, 74-75, 86
actions/commands (动作/命令), 151, 154
forms (表单), 231, 235, 239
layout (布局), 123
users (用户), 12-15
number of jumps, 56-62
overview (总览), 55
patterns overview (模式总览), 63
Pyramid (金字塔), 57, 63, 71-73
restricting, 63
Sequence Maps (序列地图), 55-56, 76-77
Breadcrumbs (面包屑层级结构), 78
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns (模式) overview (模式总览), 63
Visual Framework (视觉框架), 101
Wizards (向导), 44
signposts (路标), 55, 63
wayfinding (找路), 55-56
environmental clues (环境线索),

56
good signage (好路标), 55
maps (地图), 56, 63
New-Item row (新对象行) (information graphics), 193-194, 213
noun/verb model (名词/动词模型) (interface architecture), 22-23
numbers (数字)
controls (控件), 215-216
entering (进入), 215
jumps (跳转), 56-62

O

objects, lists of (对象列表), 23-24
observation (观察) (user research), 6
One-Off Mode (一次性模式) (editors), 35, 245, 248, 255-258
One-Window Drilldown (单窗口深入) (information architecture), 24-25, 29-32, 36-38
Breadcrumbs (面包屑层级结构), 78
Hub and Spoke (中心和辐条), 68
Modal Panel (模态面板), 75
Pyramid (金字塔), 71-72
one-window paging (单窗口页面) (information architecture), 28-29, 37
open source (开源), 19, 205, 270, 308
opening (打开) (information graphics), 167, 186
organization (组织)
content (内容)。参见 content organization
information graphics (信息图形), 162
physical structure (物理结构) (information architecture)
Alternative Views (可选视图), 30, 39-41, 173
Canvas Plus Palette (画布加调色板工具条), 25, 29-30, 34-35, 243
multiple windows (多窗口), 28-29, 33, 36-37, 100
One-Window Drilldown (单窗口深入)。参见 One-Window Drilldown (单窗口深入)
one-window paging (单窗口页面), 28-29, 37
overview (总览), 28
tiled panes (平铺面板)。参见 tiled panes

Two-Panel Selector (双面板选择器)。参见 Two-Panel Selector
other people's advice (旁人建议) (user behavior), 18-19
Overview Plus Detail (总览加细节), 167, 174-175, 185
information architecture (信息架构), 33
navigation (导航), 56, 63, 80
P
Pages (页面)。参见 panes/patterns/windows
palette (调色板工具条) (Canvas Plus Palette), 25, 29-30, 34-35, 243
page layout (页面布局), 89-129
palmtops (Palm桌面), 21, 27
panels (面板)。参见 panes/patterns/windows
panes/patterns/windows
Action Panels (动作面板), 132, 136, 140-143
Card Stack (卡片堆)
information architecture (信息架构), 33-34, 44
information graphics (信息图形), 196
layout (布局), 98-99, 101, 108-112
navigation (导航), 57
Closable Panels (可关闭的面板)
actions/commands (动作/命令), 142
information architecture (信息架构), 37, 45, 49-51
layout (布局), 98-99, 108-109, 111-113
navigation (导航), 84
layout (布局)。参见 layout
Movable Panels (可移动的面板), 98-99, 108-109, 114-115, 122
actions/commands (动作/命令), 141
information architecture (信息架构), 37, 51
layout (布局), 98-99, 108-109, 114-115, 122
users (用户), 16
multiple windows (多窗口), 28-29, 33, 36-37, 100

- one-window paging (单窗口页面), 28-29, 37
task panes (任务面板)。参见 Action Panels
tiled panes (平铺面板) (information architecture), 28-29, 34, 36-38
actions/commands (动作/命令), 141
navigation (导航), 57, 64, 78
panning (平移) (information graphics), 166-168, 175, 181
Paste Variations (粘贴变种) (editors), 248, 266-267
patterns (模式)
 actions/commands (动作/命令)
 overview, 136
 controls overview (控件总览), 218
 definition (定义), 161
 editors overview (编辑器总览), 247
 forms overview (表单总览), 218
 how to use (如何使用), 63
 information architecture (信息架构)
 overview (总览), 30
 information graphics (信息图形)
 overview (总览), 173
 languages (语言), 315
 navigation overview (导航总览), 63
 styles overview (风格总览), 290
 user behavior (用户行为)
 changes in midstream (中途变卦), 12-13
 deferred choices (延后选择), 13-14
 habituation (习惯), 14-15
 incremental construction (递增构建), 14, 243
 instant gratification (即时满足), 11
 keyboard only (只支持键盘), 17-18
 other people's advice (旁人建议), 18-19
 overview (总览), 10
 prospective memory (前瞻记忆), 16-17
 safe exploration (安全探索), 11
 satisficing (满意即可), 11-12
 spatial memory (空间记忆), 15-16, 166
 streamlined repetition (简化重复工作), 17
- people's advice (人们的建议) (user behavior), 18-19
personas (人物角色) (user research), 7
PDAs, 25, 39, 40, 65, 67, 98
physical structure (物理结构) (information architecture)
 Alternative Views (可选视图), 30, 39-41, 173
 Canvas Plus Palette (画布加调色板工具条), 25, 29-30, 34-35, 243
 multiple windows (多窗口), 28-29, 33, 36-37, 100
 One-Window Drilldown (单窗口深入)。参见 One-Window Drilldown
 one-window paging (单窗口页面), 28-29, 37
 overview (总览), 28
 tiled panes (平铺面板)。参见 tiled panes
 Two-Panel Selector (双面板选择器)。参见 Two-Panel selector
Plots (图), 162, 170, 172, 198
Trellis plot (格子图形), 201
(参见 also graphs; information graphics)
pop-up menus (弹出菜单) (actions/commands), 132
postures (姿态) (programs), 9
preattentive variables (前摄变量) (information graphics), 163-166, 182
Preview (预览) (actions/commands), 44, 136, 147-148, 233
Principles (原则)
 controls (控件), 209-210
 forms (表单), 207-209
Print preview (打印预览), 147
program structure (程序结构), 9
Progress Indicator (进度提示) (actions/commands), 65, 136, 149-151
progressive disclosure (逐步展开), 57, 124
Prominent Done Button (突出的“完成”按钮) (actions/commands), 136, 144-145
Property sheet (属性表)
 editors (编辑器), 249
 forms (表单), 208-209
- information architecture (信息架构), 32
layout (布局), 99, 109, 112, 120-122, 124
prospective memory (前瞻记忆) (user behavior), 16-17
Proximity (相邻性), 94-95, 97
Action Panels (动作面板), 141
alternating row colors (交替的表格行颜色), 187
button groups (按钮分组), 137
Proximity (相邻性) (continued)
Datatips (数据提示), 172
Multi-Y Graph (多Y轴图表), 198
responsive enabling (响应式允许), 126
right-left alignment (右对齐左对齐), 116
styles (风格), 269
Pyramid (金字塔) (navigation), 57, 63, 71-73

Q-R

- querying (查询)。参见 searching
Radio buttons (单选按钮), 179, 209-211
Raymond, Eric s., 4
rearranging (重新排列)。参见 sorting
red/green colors (红/绿颜色), 240, 280
Reimann, Robert, 153
repeated visual motifs (重复的视觉主题), 286-287
repetition (重复) (user behavior), 17
repetitive strain injury (重复性的韧带损伤), 245
research (研究), users (用户), 5-7
resizing objects on canvas (在画布上重定义对象大小), 248, 253, 256, 259
Responsive Disclosure (响应式展开)
 information architecture (信息架构), 43, 46
 layout (布局), 98-99, 122-127
 users (用户), 15
Responsive Enabling (响应式允许)
 (layout), 43, 98-99, 123, 125-127
restricting navigation (限制导航), 63
Revert to Default (恢复默认设置), 115
rhythm (旋律) (visual design), 284, 287
Right/Left Alignment (右对齐/左对齐), 99, 116-117, 121-122

- root cause analysis (根本原因分析),
(users), 4
rulers (尺子) (information graphics),
172
- ## S
- safe exploration (安全探索) (user behavior), 11
Same-Page Error Messages (错误显示在同一一页), 239-241
Sano, Darrell, 118
sans-serif fonts (无衬线字体), 281-283, 300, 303
satisficing (满意即可) (user behavior), 11-12
scales (比例) (information graphics), 172
screens (屏幕)。参见 panes/panels/windows
scrollbars (滚动条)
actions/commands (动作/命令), 134, 139, 153
Annotated Scrollbar (注释滚动条), 55, 63, 80-81, 173
information graphics (信息图形), 166-167, 170, 173-175
Cascading Lists (级联列表), 195
Datatips (数据提示), 177
Jump to Item (跳转到对象), 191-192
layout (布局)
dynamic displays (动态显示), 98
Liquid Layout (流式布局), 129
Searching (搜索)。也参见 filtering
information graphics (信息图形), 161, 166, 168-171
Jump to Item (跳转到对象), 192
Local Zooming (局部缩放), 184
Overview Plus Detail (总览加细节), 175
Sortable Table (可排序表格), 189
security (安全), 159, 208
selection (选择)
builders and, 244
composites (组合选择), 248-253
editors and (编辑器), 244-248
lasso (套索选择), 35, 182, 253, 255
of list items (列表对象选择), 245
multiple (多选), 236, 258
rubber band (橡皮框选择), 180, 245, 246
- single (单选), 258
of text (文字选择), 245
self-describing (自述), 137, 147
Sequence Maps (序列地图), 55-56, 76-77
Breadcrumbs (面包屑层级结构), 78
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns overview (模式总览), 63
Visual Framework (视觉框架), 101
Wizards (向导), 44
shadows (阴影) (actions/commands), 135
Shneiderman, Ben, 205
signposts (路标) (navigation), 55, 63
Annotated Scrollbar (注释滚动条), 55, 63, 80-81, 173
Breadcrumbs (面包屑层级结构), 55, 78-79
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns overview (模式总览), 63
Sequence Maps (序列地图), 76
visual framework (视觉框架), 101
Color-Coded Sections (颜色编码的栏目), 55, 63, 82-83
Sequence Maps (序列地图), 55-56, 76-77
Breadcrumbs (面包屑层级结构), 78
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns overview (模式总览), 63
Visual Framework (视觉框架), 101
Wizards (向导), 44
similarity (相似性), 95, 97
button groups (按钮分组), 137
Multi-Y Graph (多Y轴图表), 198
preattentive variables (前摄变量), 163, 166
styles (风格), 269, 287
skins (皮肤) (styles), 286-287, 308-311
sliders (滑块), 138, 176, 179
Small Multiples (大量小对象) (information graphics), 200-202, 287
Smart Menu Items (智能菜单项)
- (actions/commands), 136, 146, 152, 154-155
Smart Selection (智能选择) (editors), 247-248, 251-252
snap to grid (贴近表格), 260
softkeys (软键), 25, 65
Sortable Table (可排序表格) (information graphics), 33, 40, 161-162, 168-169, 190
sorting (排序) (information graphics), 162, 168-169
Dynamic Queries (动态查询), 178
information architecture (信息架构), 23-25, 27, 32-33, 39, 40
navigation (导航), 73
patterns overview (模式总览), 173
Sortable Table (可排序表格) (information graphics), 33, 40, 161-162, 168-169, 190
Tree Table (树状表格), 197
sovereign structure programs (独占结构程序), 9
spaciousness and crowding (空间和拥挤), 283-284
spatial memory (空间记忆) (user behavior), 15-16, 166
spatial relationships (空间关系)
visual design (视觉设计) (styles), 279, 283-285, 300, 307-308
visual hierarchy (视觉层次结构) (layout), 91-92, 99
spin box (微调输入框), 209, 215
spinners (微调按钮), 211, 216
spreadsheets, 81, 84, 103
Spring-Loaded Mode (弹性加载模式) (editors), 18, 35, 245, 248, 257-258
statistics (统计学) (color-blindness), 280
streamlined repetition (简化重复工作) (user behavior), 17
strong titles (大标题) (Titled sections)
actions/commands (动作/命令), 148
information architecture (信息架构), 43-44
information graphics (信息图形), 187
layout (布局), 98-99, 107-108
Card Stack (卡片堆), 109
Center Stage (中央舞台), 104

- Closable Panels (可关闭的面板), 111, 113
Visual Framework (视觉框架), 101-102
styles (风格), 287-288, 303
- structure**
content organization (内容组织)。
· 参见 content organization
physical structure (物理结构)
(information architecture)
Alternative Views (可选视图), 30,
39-41, 173
Canvas Plus Palette (画布加调色板
工具条), 25, 29-30, 34-35, 243
multiple windows (多窗口),
28-29, 33, 36-37, 100
One-Window Drilldown (单窗
口深入)。参见 One-Window
Drilldown
one-window paging (单窗口页
面), 28-29, 37
overview (总览), 28
tilted panes (平铺窗格)。参见
tilted panes
Two-Panel Selector (双面板选择
器)。参见 Two-Panel Selector
Structured Format (结构化格式)
(controls), 208, 215, 217, 219-221,
224
- Structured Format (结构化格式)**
(controls), 208, 215, 217, 219-221,
224
- styles (风格)**
accessibility (可访问性), 288-289
angles (角度), 279, 282-287
Borders that Echo Fonts (边界回应
字体), 302
Corner Treatments (角落处理),
297-300
Skins (皮肤), 311
backgrounds (背景), 280, 286-288,
290-293
Deep Background (深色背景),
288, 291-293, 307, 311
Few Hues, Many Values (少一点色
彩, 多一些价值), 295
Hairlines (发丝), 303-304
borders (边界), 280, 283, 287-288
Borders that Echo Fonts (边界回应
字体), 287-288, 300-302
Corner Treatments (角落处理),
298-299
Hairlines (发丝) 303
patterns overview (模式总览),
290
- Borders that Echo Fonts (边界回应
字体), 287-288, 300-302
branding (品牌建立), 270, 279
Color-Coded Sections (颜色编码的
栏目), 82-83
cultural references (文化含义),
286
Deep Background (深色背景),
291
Escape Hatch (逃生船), 87
layout (布局), 92, 100
signposts (路标), 55
Contrasting Font Weights (粗细字体
大小), 306, 307
Corner Treatments (角落处理),
287-288, 297-300, 302, 309, 311
creativity (创意), 287-289
CSS Zen Garden (CSS禅意花园),
270-279
Deep Background (深色背景), 288,
291-293, 307, 311
examples (示例), 270-279
Few Hues, Many Values (少一点色
彩, 多一些价值), 281, 294-296
Hairlines (发丝), 284, 303-305, 307
overview (总览), 269-270
patterns overview (模式总览), 290
Skins (皮肤), 286-287, 308-311
user trust (用户信任), 269
visual design (视觉设计), 279-287
angles (角度), 279, 282-287,
297-300, 302, 311
color (颜色), 279-288, 291-296,
300-308
cultural references (文化含义),
279, 283, 286, 295
curves (曲线)。参见 curves
icons (图标)。参见 icons
images (图片)。参见 images
motifs (主题), 279, 286-287, 290,
297-298, 300
rhythm (旋律), 284, 287
spatial relationships (空间关系),
279, 283-285, 300, 307-308
textures (底纹)。参见 textures
typography/fonts (字体)。参见
typography/fonts
- summary (小结) (Preview), 44, 136,
147-148, 233
surveys (调查) (user research), 6-7
switching contexts (上下文切换)
(navigation), 56-62, 100, 109, 123

T

- tabs, 34, 55, 66-67, 297
vertical tabs (竖直tab), 109
(参见 also Card stack)
task panes (任务面板) (Action Panels),
132, 136, 140-143
text (文字)。参见 typography/fonts
multiline text area (多行文本输入
框), 214
text field (文本输入框), 117, 120-121,
125
text line length (文字行长度), 129
textures (底纹) (styles), 279, 281-282,
284-286, 288
Borders that Echo Fonts (边界回应字
体), 300
Contrasting Font Weights (字体重量
对比), 306
textures (底纹) (styles) (continued)
Deep Background (深色背景), 292
Hairlines (发丝), 303-304
patterns overview (模式总览), 290
tilted panes (平铺窗格) (information
architecture), 28-29, 34, 36-38
actions/commands (动作/命令),
141
navigation (导航), 57, 64, 78
timelines (时间轴) (information
graphics), 162, 172, 198
Titled Sections (带标题的栏目)
actions/commands (动作/命令),
148
information architecture (信息架
构), 43-44
information graphics (信息图形),
187
layout (布局), 98-99, 107-108
Card Stack (卡片堆), 109
Center Stage (中央舞台), 104
Closable Panels (可关闭面板),
111, 113
Visual Framework (视觉框架),
101-102
styles (风格), 287-288, 303
Tognazzini, bruce, 126

- toolbars (工具条) (actions / commands)**, 132
 tooltips (工具提示)
 actions / commands (动作 / 命令), 135
 Datatips (数据提示) (information graphics), 172-173, 176-177, 185, 203-204
transient structure programs (暂时结构程序), 9
tree (树), 21, 24, 33, 91, 161
tree control (树形控件), 195
Tree Tables (树状表格) (information graphics), 33, 162, 197
Treemap (树状视图) (information graphics), 162, 179, 203-205
trust, users (信任, 用户), 269
Tufte, Edward, 174
Two-Panel Selector (双面板选择器) (information architecture), 24, 31-33
 Canvas Plus Palette (画布加调色板工具条), 35
 layout (布局), 108-109
 One-Window Drilldown (单窗口深入), 36-38
 patterns overview (模式总览), 30
 tiled panes (平铺面板), 29
 Wizards (向导), 43
typed commands (键入的命令), 133
typographic errors (印刷错误), 179, 227, 239
typography/fonts (字体)
 controls (控件), 214-215
 styles (风格), 279, 281-283, 286-288
 Borders that Echo Fonts (边界回应字体), 287-288, 300-302
 Corner Treatments (角落处理), 297
 Contrasting Font Weights (字体重量对比), 306-307
 Hairlines (发丝), 303-304
 patterns overview (模式总览), 290
 Skins (皮肤), 308
 visual hierarchy (视觉层次结构) (layout), 90-91
- U**
- UML sequence diagram (UML序列图)**, 61-62
- V**
- van Welie, Martijn**, 191
vertical tabs (竖直tab) (参见 tabs)
visual design (视觉设计) (styles), 279-287
 angles (角度), 279, 282-287
 Borders that Echo Fonts (边界回应字体), 302
- undo (撤销) (Multi-Level Undo)**, 136, 153-155
Alternative Views (可选视图), 40
Escape Hatch (逃生舱), 86
Macros (宏), 157-158
Property Sheet (属性表), 122
users (用户), 11
usability testing (可用性测试), 111, 126, 138, 142, 154
users (用户)
 behavioral patterns (行为模式)
 changes in midstream (中途变卦), 12-13
 deferred choices (延后选择), 13-14
 habituation (习惯), 14-15
 incremental construction (递增构建), 14, 243
 instant gratification (即时满足), 11
 keyboard only (只支持键盘), 17-18
 other people's advice (旁人建议), 18-19
 overview (总览), 10
 prospective memory (前瞻记忆), 16-17
 safe exploration (安全探索), 11
 satisficing (满意即可), 11-12
 spatial memory (空间记忆), 15-16, 166
 streamlined repetition (简化重复工作), 17
 expectations (期望), 7-9
 goals, analyzing (目标, 分析), 3-5
 input (输入)。参见 forms
 knowledge (知识), 7-9
 Linux, 4
 research (研究), 5-7
 root cause analysis (根本原因分析), 4
 trust (信任), 269
- Corner Treatments (角落处理)**, 297-300
Skins (皮肤), 311
color (颜色), 279-288, 291-296, 300-308
cultural references (文化含义), 279, 283, 286, 295
curves (曲线)。参见 curves
icons (图标) (styles), 270, 285-286, 288
Corner Treatments (角落处理), 298
Deep Background (深色背景), 291
Few Hues, Many Values (少一点色彩, 多一些价值), 295
Hairlines (发丝), 303-304
Skins (皮肤), 309-310
images (图片) (styles), 279-280, 285-288
Borders that Echo Fonts (边界回应字体), 301
Corner Treatments (角落处理), 297
Deep Background (深色背景), 291-293
Font Weights (字体重量), 307
Hairlines (发丝), 303
Skins (皮肤), 309
motif (主题), 279, 286-287, 290, 297-298, 300
rhythm (旋律), 284, 287
spatial relationships (空间关系), 279, 283-285, 300, 307-308
textures (底纹) (styles), 279, 281-282, 284-286, 288
Borders that Echo Fonts (边界回应字体), 300
Contrasting Font Weights (粗细字体对比), 306
Deep Background (深色背景), 292
Hairlines (发丝), 303-304
typography/fonts (字体)。参见 typography/fonts
visual flow (视觉流) (layout), 92-94, 97, 99
 focal points (焦点), 92-94, 99
 angles (角度), 284
 strength (力量), 292-293, 305
 style (风格), 269

Prominent Done Button (突出的“完成”按钮), 144-145
style (风格), 269
Two-Panel Selector (双面板选择器), 32
Visual Framework (视觉框架)
(layout), 67, 82, 91, 99-102, 144
visual hierarchy (视觉层次结构)
information architecture (信息架构)。参见 information architecture
layout (布局), 89-92, 98-99, 144, 205, 269, 306
alignment (对齐), 90-91
color (颜色), 92
spatial relationships (空间关系), 91-92, 99
typography/fonts (字体), 90-91
visual weight (视觉重量), 91
whitespace (空白), 90-91
visual impairment (视觉受损)
Action Panels (动作面板), 141
actions/commands (动作/命令), 132
Color-Coded Sections (颜色编码的栏目), 83
red/green (红/绿), 240, 280
Same-Page Error Messages (错误显示在同一页), 240
statistics (统计学), 280
visual vocabulary (视觉词汇表), 233
visual weight (视觉重量) (visual hierarchy), 91
visualization (可视化), 162, 166, 174, 185, 200
(参见 also information graphics)

W-Z

wayfinding (找路) (navigation), 55-56
Annotated Scrollbar (注释滚动条), 55, 63, 80-81, 173
Breadcrumbs (面包屑层级结构), 55, 78-79
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns overview (模式总览), 63
Sequence Maps (序列地图), 76
visual framework (视觉框架),

101
Color-Coded Sections (颜色编码的栏目), 55, 63, 82-83
environmental clues (环境线索), 56
good signage (好路标), 55
maps (地图), 56, 63
Sequence Maps (序列地图), 55-56, 76-77
Breadcrumbs (面包屑层级结构), 78
Escape Hatch (逃生舱), 86
One-Window Drilldown (单窗口深入), 37
patterns overview (模式总览), 63
Visual Framework (视觉框架), 101
Wizards (向导), 44
weight (重量) (visual hierarchy), 91
whitespace (空白) (visual hierarchy), 90-91
windows (窗口)。参见 panes/panels/windows
Wizards (向导), 42-44
actions/commands (动作/命令), 147
forms (表单), 234
layout (布局), 93, 123-125
navigation (导航), 57, 69, 71, 76, 86
patterns overview (模式总览), 30
users (用户), 12
WYSIWYG editors (所见即所得编辑器), 244
Edit-In-Place (就地编辑), 249
forms (表单), 226
Guides (对齐指示线), 264
HTML, 244
information architecture (信息架构), 41
layout (布局), 120
Paste Variations (粘贴变种), 267
patterns (模式), 248
You are Here (你在这里), 32, 63
zooming (缩放) (information graphics), 161, 167-168
Data Brushing (数据刷), 181, 183
Datatips (数据提示), 176
Local Zooming (局部缩放), 184-185
Overview Plus Detail (总览加细节), 174-175
patterns overview (模式总览), 173
Treemap (树状视图), 204